QorIQ LS1012A Reference Manual

Document Number: LS1012ARM
Rev. 1, 01/2018

Contents

Section number **Title** **Page**

Chapter 1
Overview

Chapter 2
Memory Map

Chapter 3
Signal Descriptions

Chapter 4
Reset, Clocking, and Initialization

Chapter 5
Interrupt Assignments

Chapter 6
Arm Modules

Chapter 7
CSU, OCRAM, and MSCM

Chapter 8
System Counter

Chapter 9
Cache Coherent Interconnect (CCI-400)

Chapter 10
Arm CoreLink™ TrustZone Address Space Controller TZC-380

Chapter 11
Supplement Configuration Unit

Chapter 12
Device Configuration and Pin Control

Chapter 13
Run Control and Power Management (RCPM)

Chapter 14
Packet Forwarding Engine (PFE)

Chapter 15
DUART

Chapter 16
Direct Memory Access Multiplexer (DMAMUX)

Chapter 17
Enhanced Direct Memory Access Controller (eDMA)

Chapter 18
Enhanced Secured Digital Host Controller (eSDHC)

Chapter 19
FlexTimer Module (FTM)

Chapter 20
General Purpose I/O (GPIO)

Chapter 21
Inter-Integrated Circuit (I2C)

Chapter 22
Integrated Interchip Sound (I2S) / Synchronous Audio Interface (SAI)

Chapter 23
Multi Mode DDR Controller (MMDC)

Chapter 24
Pre-Boot Loader (PBL)

Chapter 25
PCI Express Interface Controller

Chapter 26
Quad Serial Peripheral Interface (QuadSPI)

Chapter 27
SATA 3.0

Chapter 28
SerDes Module

Chapter 29
Secure Boot and Trust Architecture 2.1

Chapter 30
Serial Peripheral Interface (SPI)

Chapter 31
Thermal Monitoring Unit (TMU)

Chapter 32
Universal Serial Bus Interface 2.0

Chapter 33
Universal Serial Bus Interface 3.0

Chapter 34
Watchdog Timer (WDOG)

Chapter 1
Overview

1.1 Introduction

This chapter provides an overview of the features and functionalities of the LS1012A integrated processor. LS1012A is targeted at the consumer NAS, IoT gateway, broadband Ethernet gateway, and industrial automation markets, and delivers an unmatched level of features and performance. LS1012A features an advanced 64-bit Arm® v8 Cortex®-A53 processor, with 32 KB of parity protected L1 I-cache and L1 D-cache, as well as 256 KB of ECC protected L2 cache. Hardware coherency is supported by the Arm CCI-400 interconnect. LS1012A offers an optimized blend of system acceleration logic, extremely small package size, and the ultra-low power utilization required for fan-less IoT and broadband gateways, consumer NAS, industrial automation and general embedded applications.

The LS1012A processor is a cost-effective, power-efficient, and highly integrated system-on-chip that extends the reach of NXP's QorIQ communications processors into new low-power applications without compromising on performance. The device delivers greater packet processing performance than any prior sub 1 W microprocessor with extremely power-efficient 64-bit Cortex v8 core running at up to 1000 MHz . Also, it includes an integrated hardware packet acceleration engine, together with high-speed interfaces and support for DDR3L running at 1000 MHz. High-speed peripherals include PCI Express Gen2, super speed USB 3.0, SATA 3.0, USB 2.0 and dual gigabit Ethernet controllers, with support for RGMII, and SGMII that can accommodate 2.5G Ethernet PHYs. To deliver outstanding security performance, the device features a hardware-based acceleration engine to support secure boot, and networking with both Arm TrustZone and NXP's QorIQ Trust Architecture.

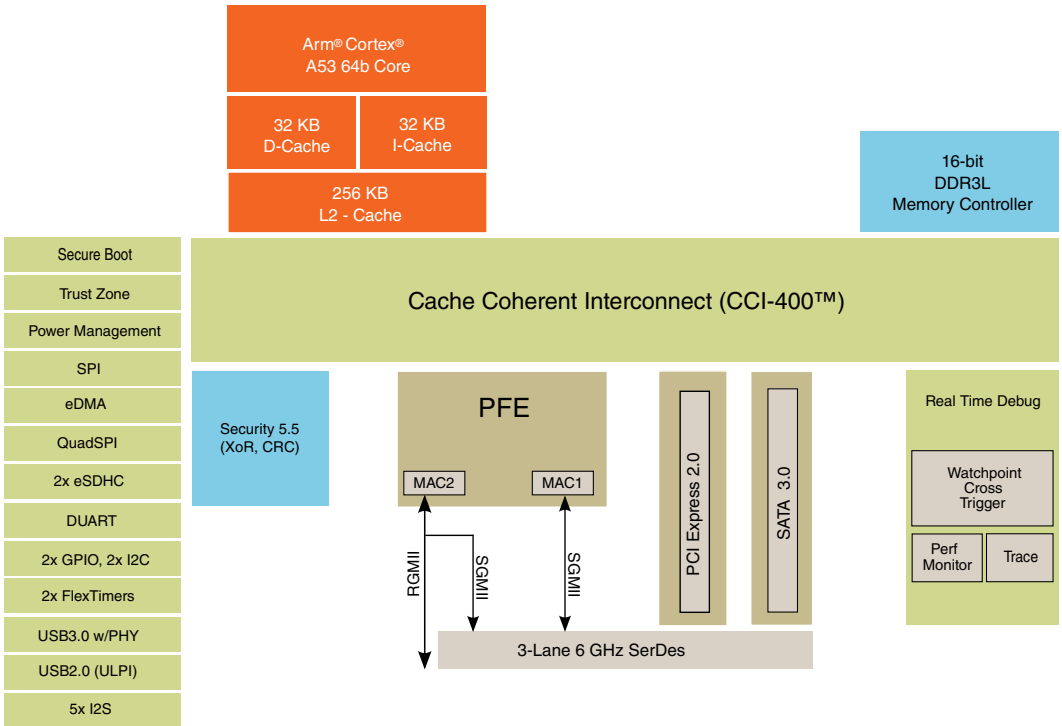


Figure 1-1. Block diagram

1.2 Features summary

The chip includes the following distinctive functions and features:

- One 64-bit Arm® v8 Cortex®-A53 core with the following capabilities:
  - Arranged as a cluster of one core supporting a 256 KB L2 cache with ECC protection
  - Speed up to 1000 MHz
  - Parity-protected 32 KB L1 instruction cache and 32 KB L1 data cache
  - Neon SIMD engine
  - Arm v8 cryptography extensions
- One 16-bit DDR3L SDRAM memory controller:
  - Up to 1.0 GT/s
  - Supports 16-/8-bit operation (no ECC support)
  - Supports for x16 devices (also supports two x8 devices totaling to 16-bit data)
- Arm core-link CCI-400 cache coherent interconnect
- Packet Forwarding Engine (PFE)
- Cryptography acceleration (SEC)
- Ethernet interfaces supported by PFE:
  - Two quad-speed Ethernet MACs supporting 2.5G, 1G, 100M, 10M
  - Supports RGMII, SGMII 1G, SGMII 2.5G

    - Up to 2 x SGMII supporting 1000 or 2500 Mbps. No limitation for 1G and 2.5G interfaces.
    - Energy efficient Ethernet support (802.3az)
- One Configurable x3 SerDes:
    - Two Serdes PLLs supported for usage by any SerDes data lane
    - Supports up to 6 Gbps operation
- High-speed peripheral interfaces:
    - One PCI Express Gen2 controller, supporting x1 operation
    - One serial ATA (SATA Gen 3.0) controller
    - One USB 3.0/2.0 controller with integrated PHY
    - One USB 2.0 controller with ULPI interface. Supports operation as a stand-alone USB host or a stand-alone device.
- Additional peripheral interfaces:
    - One quad serial peripheral interface (QuadSPI) controller for serial flash
    - One serial peripheral interface (SPI) controller
    - Two enhanced secure digital host controllers supporting SD 3.0, eMMC 4.4 and eMMC 4.5 modes
    - Two I^2C controllers
    - One 16550 compliant DUART (two UART interfaces)
    - Two general purpose IOs (GPIO)
    - Two FlexTimers
    - One eDMA controller
    - Five synchronous audio interfaces (SAI) supporting I^2S
    - QorIQ platform's trust architecture
    - Debug supporting run control, data acquisition, high-speed trace, and performance/event monitoring
    - Pre-boot loader (PBL) provides pre-boot initialization and RCW loading capabilities
    - Single-source clocking solution enabling generation of core, platform, DDR, SerDes, and USB clocks from a single external crystal and internal crystal oscillator
    - Thermal monitor unit (TMU) with +/- 3°C accuracy
    - Two WatchDog timers
    - Arm generic timer

1.3 Application examples

LS1012A is a very small footprint, highly versatile and ultra-power-efficient microprocessor that can be configured to support consumer NAS, IoT aggregation, entry-level broadband gateway applications and many other low-end communication applications. The following sections provide several use case examples supported by LS1012A.

1.3.1 Entry-level broadband Ethernet gateway

The figure below shows LS1012A supporting an entry-level broadband Ethernet gateway. Point-to-point broadband gateways can be used for copper, wireless, or fiber-to-the-home applications, which use different optical connectors for Fiber WAN interface transceivers. With this technology, ISPs and Telcos (operators and carriers) can offer greater bandwidth over greater distances to deliver IPTV or home entertainment services to end customers.

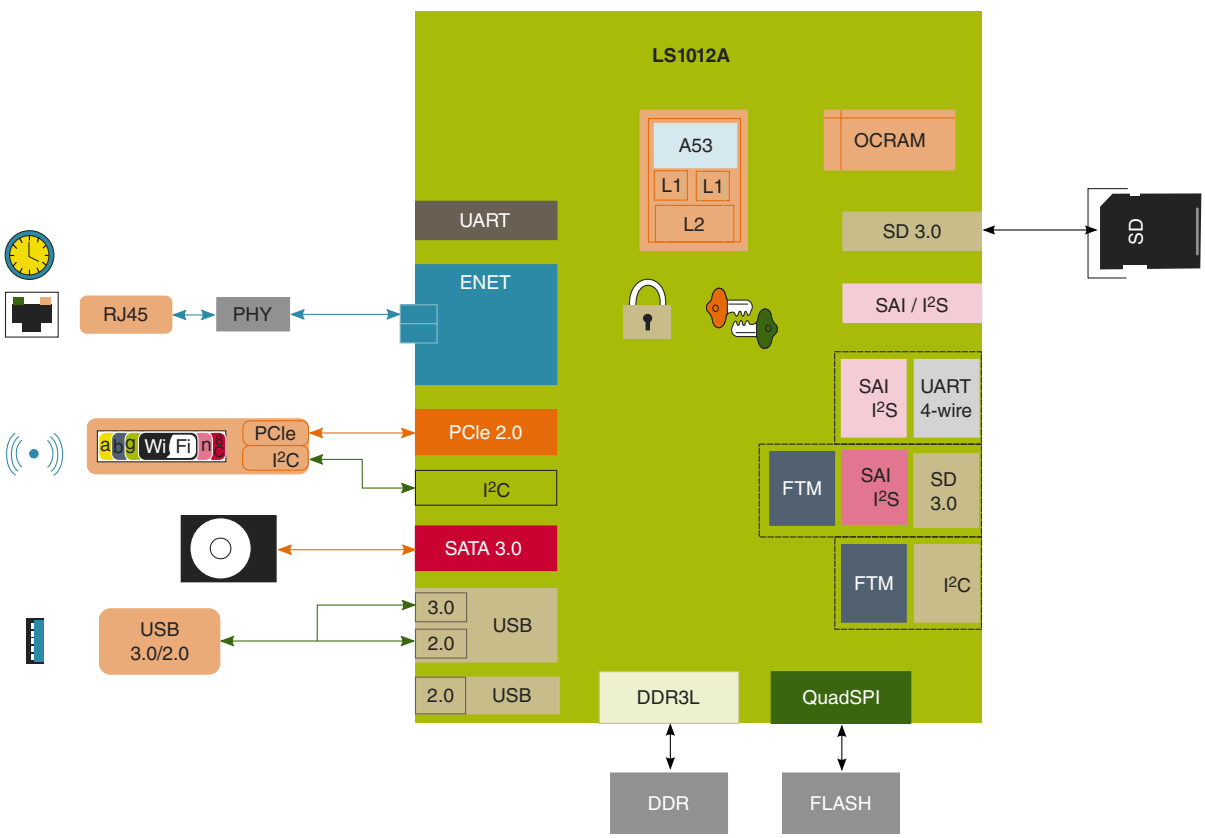


Figure 1-2. Entry-level broadband Ethernet gateway

1.3.2 Consumer NAS/DAS and battery-powered NAS applications

The increasing adoption of digital cameras and Wi-Fi connectivity in practically all mobile devices, including laptop PCs, tablets, smart phones, and digital cameras has led to an ever increasing demand for storage, over both wired and wireless networking links. As shown in the figures below, LS1012A has been engineered to support both wired and wireless storage needs with a high-speed USB 3.0 controller for direct attached modes, or through a Wi-Fi radio connected to its PCI Express port, or one of the two SDIO ports.

Following are the advantages of LS1012A when used to support consumer and battery-powered NAS application:

- Scalable 64-bit Arm Cortex-A53 core that supports Wi-Fi devices using the latest generation of 802.11ac Wi-Fi radios.
- Half-speed battery mode support for extending wireless NAS performance.
- PCI Express Gen 2 controller to support dual-band 802.11ac WLAN networks.
- USB 3.0 to support LTE/4G wireless broadband backhaul or super-fast direct attachment.
- SATA 3.0 to support high-speed data streaming to both disk-based or solid-state storage media devices.
- Dual SD/MMC enables configuration for small media storage or low-bandwidth wireless connectivity.

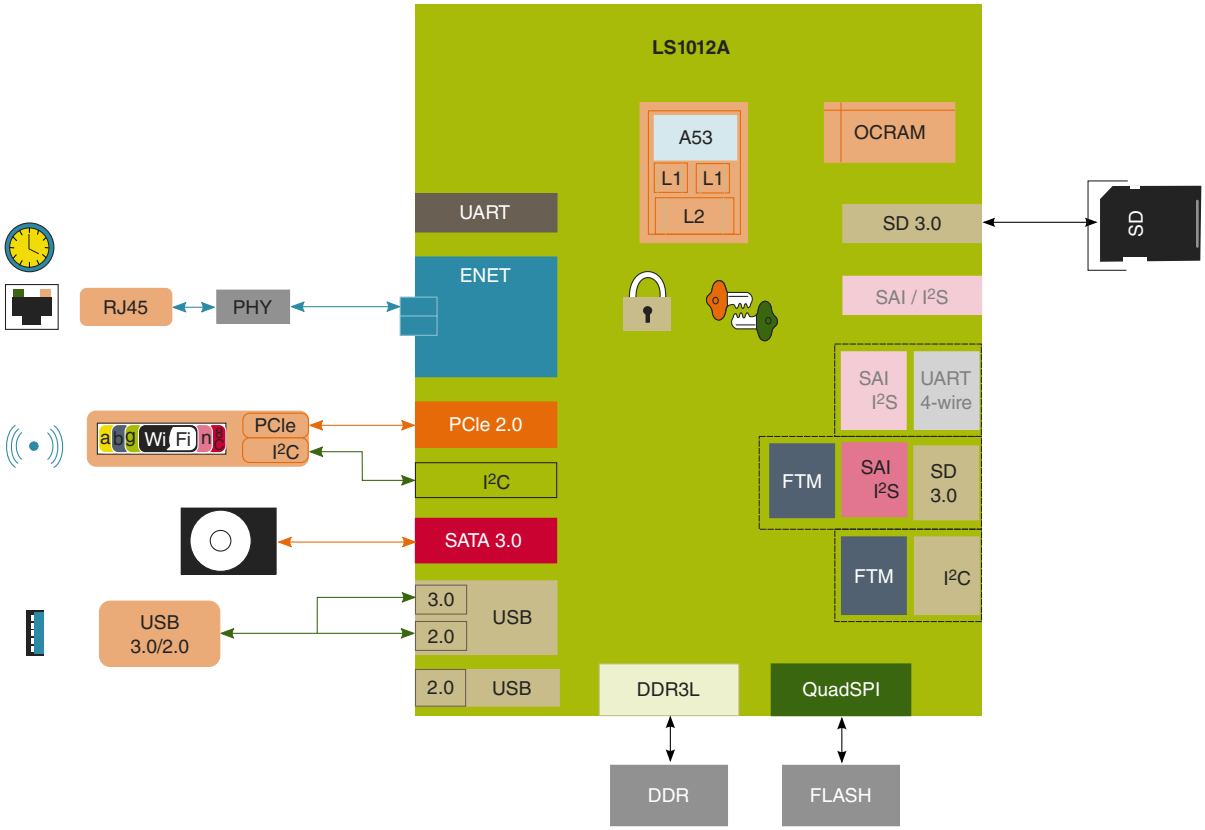


Figure 1-3. Consumer NAS/DAS with optional Wi-Fi

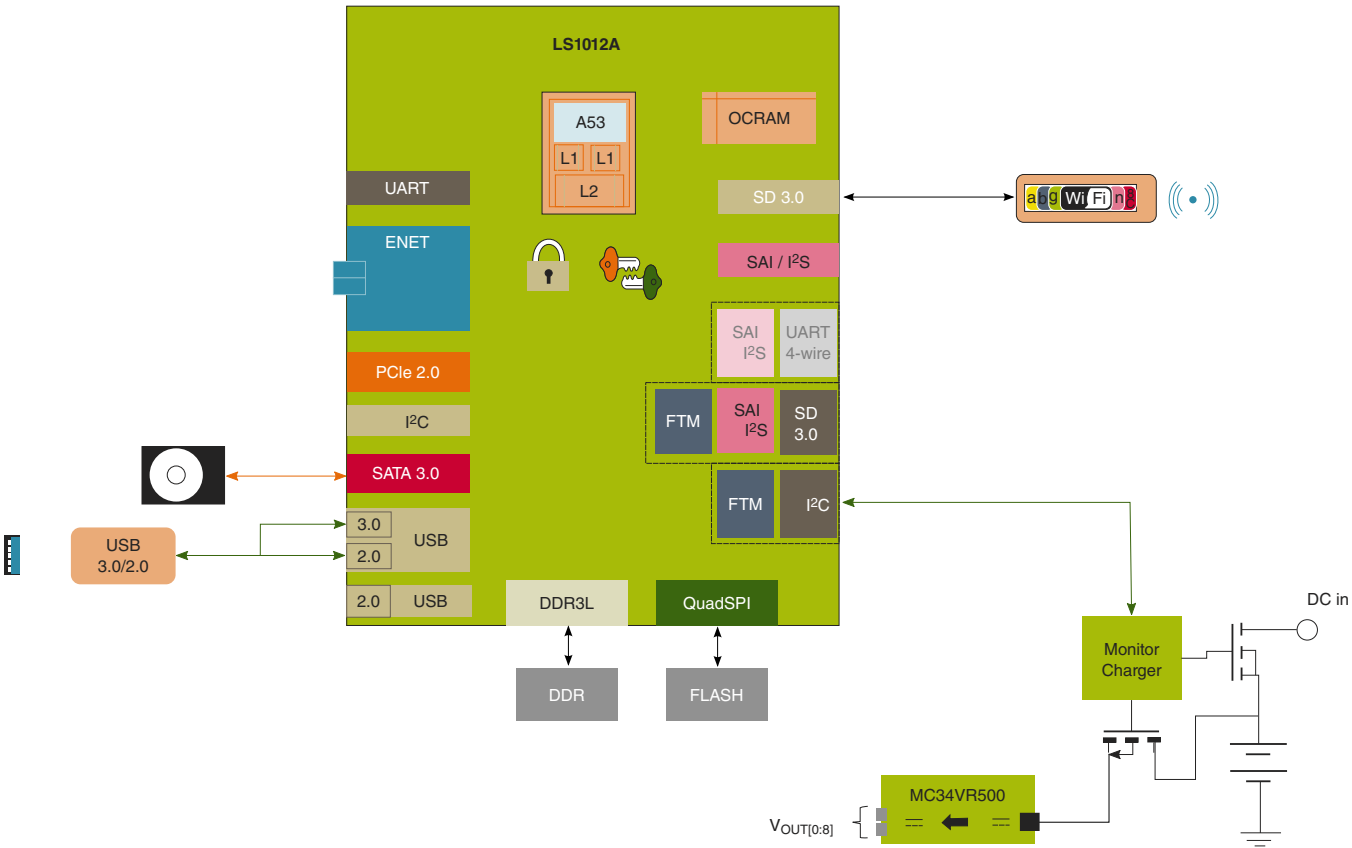


Figure 1-4. Battery-powered consumer NAS

With the ever increasing demand for scalable, cost-efficient storage growing day by day, there has been a significant move toward modular, Ethernet to SATA storage solutions. As shown in the figure below, LS1012A addresses this use case with a very low cost system design, where a HDD or SSD can be attached through SATA to a board that supports single or dual Ethernet ports. To differentiate, LS1012A supports the new 2.5G Ethernet PHY's, which significantly increase data transfer speeds.

Following are the advantages of LS1012A when used to support Ethernet NAS drive:

- Scalable 64-bit Arm Cortex-A53 core that supports high-performance NAS.
- Dual Ethernet controllers, which can be configured as the first controller supporting 2.5G SGMII PHY and the second controller supporting 1G RGMII PHY.
- SATA 3.0 to support high-speed data streaming to both disk-based or solid-state storage media devices.

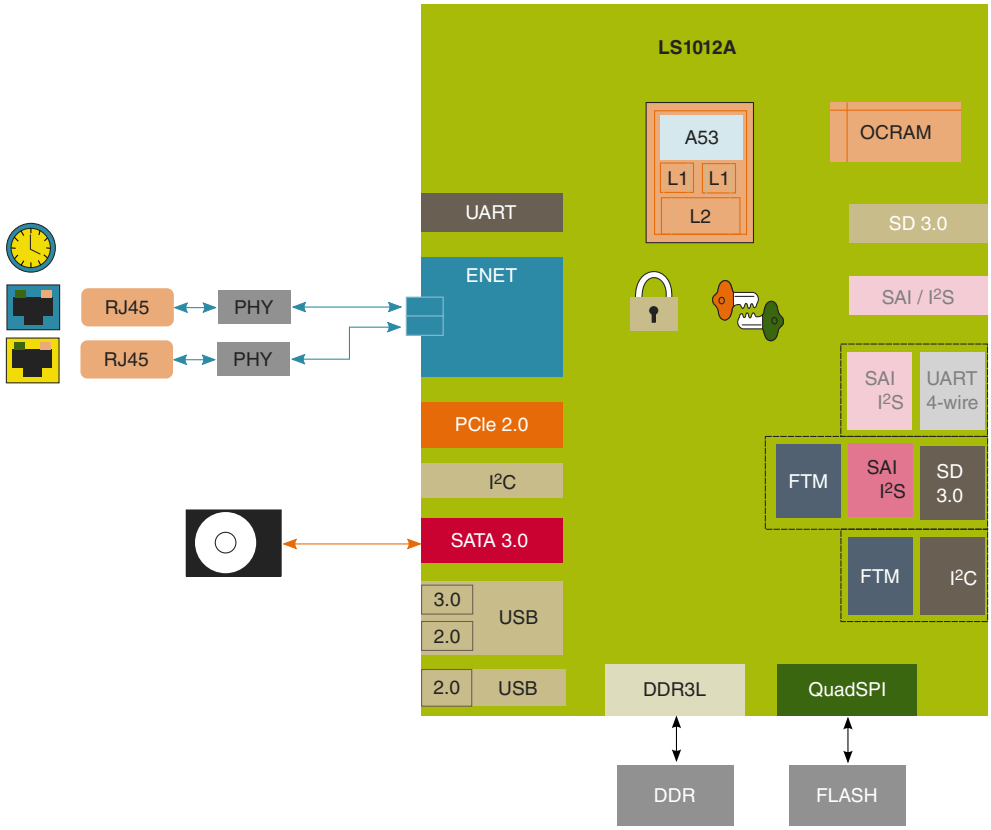


Figure 1-5. Ethernet to SATA HDD

1.3.3 IoT gateways

The figures below illustrate the flexibility of LS1012A showing different use cases for IoT aggregators and gateways, including support for next generation 802.11ac wireless LAN networking and a variety of protocols used for IoT node connectivity.

Following are the advantages of using LS1012A for IoT gateways:

- Scalable 64-bit Arm Cortex-A53 core that supports Wi-Fi devices using the latest generation of 802.11ac Wi-Fi radios.
- PCI Express Gen 2 controller to support dual-band 802.11ac WLAN networks.
- USB 3.0 to support for LTE/4G wireless broadband backhaul or super-fast data transfers.
- SATA 3.0 to support for high-speed data streaming to both disk-based or solid-state storage media devices.
- Dual SD/MMC enables configuration for small media storage or low-bandwidth wireless audio streaming.

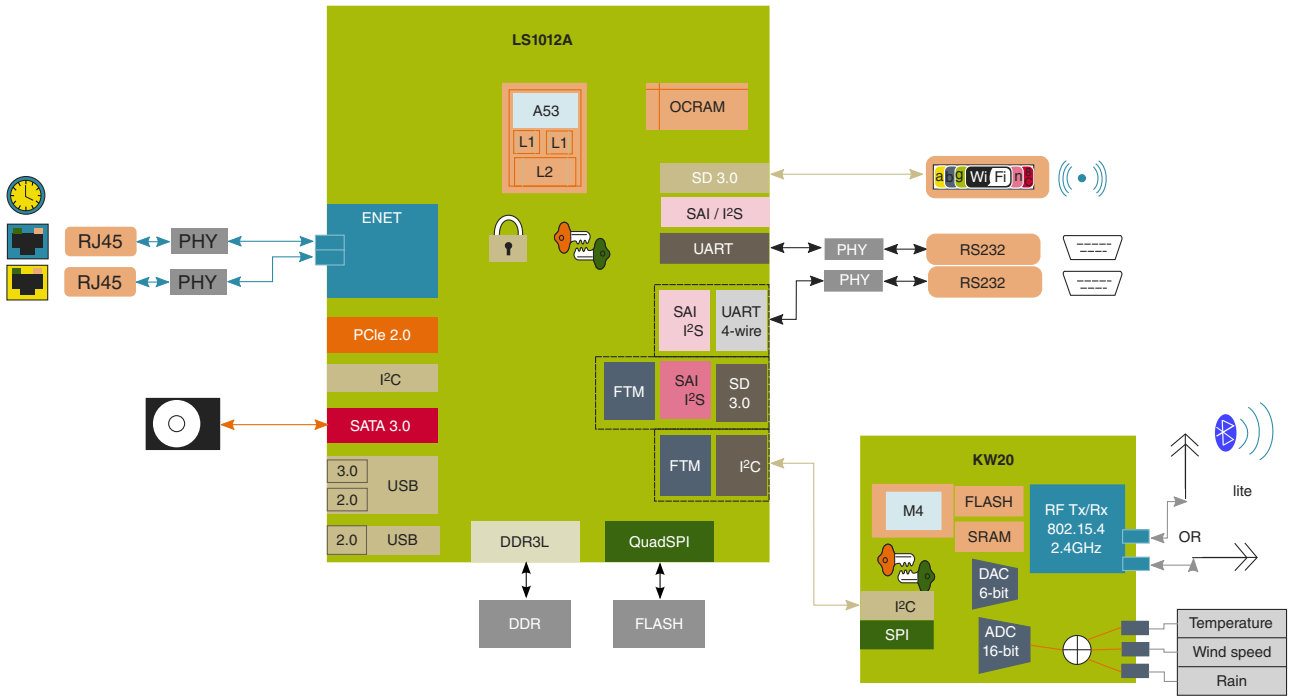


Figure 1-6. IoT gateway with LS1012A

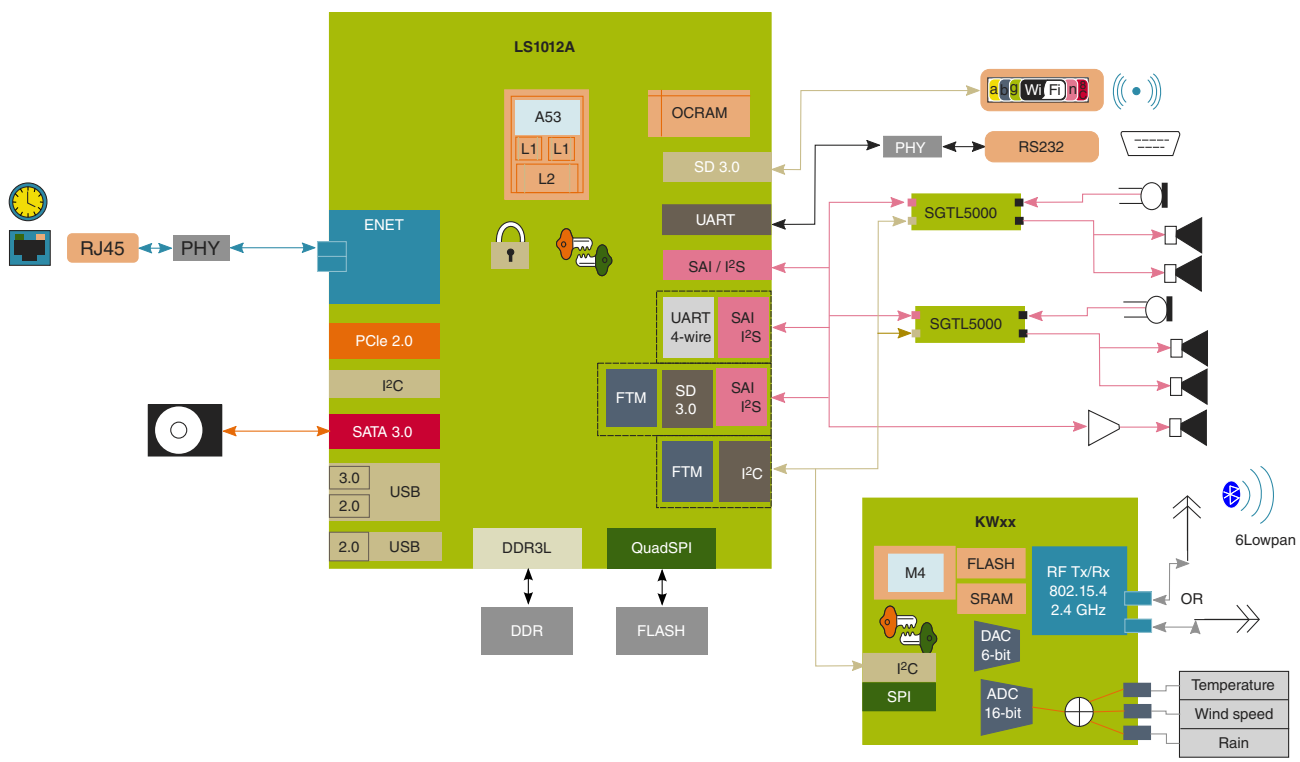


Figure 1-7. IoT gateway with audio networking

1.4 Chip features

This section contains a high level view of the chip architecture with the key features and functionalities of the chip.

1.4.1 Arm Cortex-A53 core

The Arm® Cortex®-A53 processor is an extremely power efficient Arm v8 processor capable of supporting 64-bit code seamlessly. It makes use of a highly efficient 8-stage in-order pipeline balanced with advanced fetch and data access techniques for performance.

The chip features one high-performance Cortex-A53 core:

- 64-bit execution state for scalable high performance
- 32 KB Instruction Cache, 32 KB Data Cache with parity protection
- NEON technology - Accelerates multimedia and signal processing algorithms such as video encode/decode, 2D/3D graphics, gaming, audio and speech processing, image processing, telephony, and sound synthesis. Also useful in accelerating floating point code with SIMD execution
- Floating point unit - Hardware support for floating point operations in half-, single- and double-precision floating point arithmetic. This also includes IEEE754-2008 enhancements
- Program Trace Macrocell and CoreSight Design Kit for unobtrusive tracing of instruction execution
- Integrated low-latency high bandwidth level-2 cache controller, supporting 256 KB coherent L2 cache with ECC (correct single-bit errors and detect double-bit errors)
  - 16 way cache with sequential TAG and Data RAM access
  - Automatic data prefetching into L2 cache for load streaming
- NEON SIMD extensions onboard

1.4.2 Arm CoreLink CCI-400 cache coherent interconnect

CCI-400 combines interconnect and coherency functions into a single module. The CCI-400 cache coherent interconnect is an infrastructure component that supports:

- Data coherency between the Cortex-A53 core and all I/O masters with three independent Points-of-Serialization (PoS) and full barrier support high-bandwidth, interconnect functionality between the masters and up to three slaves

- Quality-of-Service (QoS) regulation for shaping traffic profiles
- Performance monitoring unit (PMU) to count performance-related events
- Programmers view (PV) to control the coherency and interconnect functionality

1.4.3 PreBoot loader and non-volatile memory interfaces

The PBL performs the following functions:

- Simplifies boot operations, replacing pin strapping resistors with configuration data loaded from non-volatile memory.
- Uses the configuration data to initialize other system logic and to copy data from QuadSPI into fully initialized DDR or OCRAMs.
- Releases CPU from reset, allowing the boot processes to begin from fast system memory.

The non-volatile memory interfaces accessible by the PBL are described in the subsequent sections. Note that these interfaces may be accessed by software running on the CPU following boot; they are not dedicated to the PBL.

1.4.4 Multi-mode DDR controller (MMDC)

The MMDC is a configurable high performance and optimized DDR controller that supports DDR3L. The key features are as follows:

- Supports data rate up to 1.0 GT/s
- One 16-bit DDR3L SDRAM memory controller
- Supports 16-bit operation (no ECC support)
- Supports one x16 or two x8 devices totaling to 16-bit data
- Supports one chip-select

1.4.5 Enhanced direct memory access (eDMA) and direct memory access multiplexer (DMAMUX)

The eDMA module has the following general features:

- 32 channels support independent 8-, 16- or 32-bit single value or block transfers
- Supports variable sized queues and circular queues
- Source and destination address registers are independently configured to post increment or remain constant

- Each transfer is initiated by a peripheral, CPU, periodic timer interrupt or eDMA channel request
- Each DMA channel can optionally send an interrupt request to the CPU on completion of a single value or block transfer
- DMA transfers possible between system memories, General Purpose I/Os (GPIOs) and Slave Peripherals that support DMA
- Programmable DMAMUX allows assignment of any DMA source to any available DMA channel with up to a total of 64 potential request sources

The DMA channel request can be initiated by the following peripherals:

- 2x I^2C
- 1x SPI
- QuadSPI

The DMA requests from these peripherals are connected to eDMA through DMAMUX and the implementation details can be found in LS1012A DMAMUX module special consideration.

The DMAMUX selects from many DMA requests down to 16 for the DMA controller. There are 2 DMAMUXs associated with each 32-channel DMA.

1.4.6 DUART

DUART supports full-duplex operation and is compatible with the PC16450 and PC16550 programming models. All the transmitter and receiver support 16-byte FIFOs. It also supports auto flow for clear to send (CTS_B) and ready to send (RTS_B) modem control functions.

1.4.7 FlexTimer module (FTM)

The key features of the FTM are as follows:

- Selectable FTM source clock and programmable prescaler
- 16-bit counter supporting free-running or initial/final value and counting is up or up-down
- Input capture, output compare, edge aligned and center aligned PWM modes
- Operation of FTM channels as pairs with equal outputs, pairs with complimentary outputs, or independent channels with independent outputs
- Deadtime insertion is available for each complementary pair
- Generation of hardware triggers

- Software control of PWM outputs
- Up to four fault inputs for global fault control
- Configurable channel polarity
- Programmable interrupt on input capture, reference compare, overflowed counter or detected fault condition
- Quadrature decoder with input filters, relative position counting and interrupt on position count or capture of position count on external event
- DMA support for FTM event

1.4.8 Universal serial bus (USB) 3.0 controller and PHY

The universal serial bus (USB) 3.0 controller with integrated PHY provides point-to-point connectivity that complies with the Universal Serial Bus Revision 3.0 Specification. The USB controller and integrated PHY can be configured to operate as a stand-alone host, a stand-alone device, or OTG.

The host and device functions are configured to support the following types of USB transfers:

- Bulk
- Control
- Interrupt
- Isochronous

Key features of the USB 3.0 controller include the following:

- Supports OTG 2.0
- USB dual-role operation and can be configured as host or device
- Supports operation as a stand-alone USB device
  - Supports one upstream facing port
  - Supports six programmable USB endpoints
- Supports operation as a stand-alone USB host controller
  - Supports USB root hub with one downstream-facing port
  - Extensible host controller interface (xHCI) compatible
- Super-speed (5 GT/s), High-speed (480 Mbps), and full-speed (12 Mbps) operations.

1.4.9 USB 2.0 controller with ULPI interface

The USB 2.0 controller with ULPI interface provides point-to-point connectivity that complies with the USB specification, Rev. 2.0. The device doesn't have an integrated PHY for this USB 2.0 controller and instead supports external PHY with ULPI interface.

The USB 2.0 controller with ULPI interface can be configured to operate as a stand-alone host controller or a stand-alone device.

Key features of the USB 2.0 controller include the following:

- Complies with USB specification, Rev. 2.0
- Supports high-speed (480 Mbps), full-speed (12 Mbps), and low-speed (1.5 Mbps) operations.
- USB dual-role operation and can be configured as host or device
- Supports operation as a stand-alone USB device
  - One upstream-facing port
  - Six programmable USB Endpoints
- Supports operation as a stand-alone USB host controller
  - USB root hub with one downstream-facing port
  - Enhanced host controller interface (EHCI) compatible

1.4.10 High-speed I/O interfaces

The chip supports the SGMII, PCI Express 2.0, USB 3.0, and SATA3 high-speed I/O interface standards.

1.4.10.1 Serial advanced technology attachment (SATA) controller

The serial advanced technology attachment (SATA) controller is compliant with the Serial ATA 3.0 Specification and the AHCI 1.3.1 specification. The SATA controller includes the following features:

- Supports SATA Gen1, Gen2, and Gen3 data rates: 1.5 Gbps (first-generation SATA), 3 Gbps (second-generation SATA), and 6 Gbps (third-generation SATA)
- Single SATA 3.0 controller with chip-level interface
- Supports asynchronous notification and hot plug
  - Asynchronous signal recovery
  - Link power management
  - Native command queuing
  - Staggered spin-up
  - Port multiplier support
- Standard ATA master-only emulation
- Supports ATA shadow registers
- SATA superset registers
- SError, SControl, SStatus
- Interrupt driven

- Supports power management
- Supports error handling and diagnostic features
- Far-end/near-end loopback
- Failed CRC error reporting
- Increased ALIGN insertion rates
- Scrambling and CONT override

1.4.10.2 PCI Express 2.0 interface

The PCI Express interface is compatible with the PCI Express Base Specification Revision 3.0. Key features of the PCI Express interface include the following:

- Power-on reset configuration options allow Root Complex or Endpoint functionality
- The physical layer operates at 2.5 or 5 GT/s data rate
- Both 32- and 40-bit addressing and 256-byte maximum payload size
- Full 64-bit decode with 36-bit wide windows
- Inbound INTx transactions
- Message Signaled Interrupt (MSI) transactions

1.4.10.3 SGMII

The serial gigabit media independent interface (SGMII) is a high-speed interface linking the Ethernet controller with an Ethernet PHY. SGMII uses differential signaling for electrical robustness. Only four signals are required: receive data and its inverse, and send data and its inverse; no clock signals are required. SGMII operates up to 2.5G or 1G depending on the maximum rate selection. The chip does not support SGMII for backplane/PHY-less use cases.

1.4.11 Integrated interchip sound (I^2S) / synchronous audio interface (SAI)

LS1012A integrates five I^2S/SAI modules, out of which two are full duplex and three are simplex. The I^2S module provides a synchronous audio interface (SAI) that supports full-duplex serial interfaces with frame synchronization such as I^2S, AC97, TDM, and codec/DSP interfaces.

The I^2S/SAI module provides the following features:

- Transmitter with independent bit clock and frame sync supporting a data line
- Receiver with independent bit clock and frame sync supporting a data line

- Maximum frame size of 32 words
- Word size between 8 bits and 32 bits
- Word size configured separately for the first word and the remaining words in a frame
- Asynchronous 32 x 32-bit FIFO for each transmit and receive channel
- Supports graceful restart after FIFO error

1.4.12 Enhanced secure digital host controller and SDIO

The chip supports two eSDHC interfaces.

- eSDHC1: Supported primarily for SD cards. Card initialization happens at 3.3V, but can dynamically switch to 1.8V.
- eSDHC2: Supported for 1.8V embedded SDIO and 1.8V eMMC.

Key features of the SD/eSDHC/eMMC controller include the following:

- Conforms to the SD host controller standard specification version 3.0
- Compatible with the eMMC system specification version 4.5
- Compatible with the SD Memory Card Physical Layer specification version 3.01
- Compatible with the SD - SDIO card specification version 2.0
- Designed to work with eMMC devices as well as SD Memory, SDIO, and SD combo cards and their variants
- Supports SD UHS-1 speed modes

Table 1-1. eSDHC parameters supported on chip

eSDHC parameter	eSDHC1 4 bit	eSDHC2 4 bit
SD (SD cards/SDIO cards/embedded SDIO)		
SD memory card	Y	N
SDIO card	Y	N
eSDIO	N	Y
MMC/ eMMC		
eMMC DDR	Y	Y
eMMC FS/HS/HS200	Y	Y

NOTE

For maximum speed supported on each interface, refer to Device data sheet and AN5192.

1.4.13 Integrated security engine (SEC)

The security engine (SEC 5.5) is designed to support Secure Boot, Arm TrustZone as well as Trust Architecture. The modular and scalable SEC 5.5 supports booting to a known good state (secure boot) with untamperable boot code, key storage, IO protection, and secure debug.

The SEC 5.5 security engine can process all algorithms associated with IPSec, IKE, SSL/TLS, iSCSI, SRTP, IEEE Std 802.11i™, IEEE Std 802.16™ (WiMAX), and IEEE Std 802.1AE™ (MACSec).

Also, the SEC 5.5 is optimized to perform multi-algorithmic operations (for example, 3DES-HMAC-SHA-1) in a single pass of the data.

The SEC 5.5 security engine supports the following key features and functions:

- XOR engine for parity checking in RAID storage applications
- Four crypto-channels, each supporting multi-command descriptor chains
- Cryptographic execution units:
  - PKHA - Public Key Hardware Accelerator
  - DESA - Data Encryption Standard Accelerator
  - AESA - Advanced Encryption Standard Accelerator
  - MDHA - Message Digest Hardware Accelerator
  - CRCA - Cyclical Redundancy Check Accelerator
  - RNG - Random Number Generator

NOTE

For read transactions to be coherent from Security module, both of the following registers must be configured:

- MCFGR[NSP] (which controls the coherency)
- MCFGR[ARCACHE] (which controls the AxCache port-connectivity)

1.4.14 Inter-integrated circuit (I^2C) controller

The I^2C is integrated with eDMA and allows communication between a number of devices. LS1012A supports two I^2C controllers.

1.4.15 Quad serial peripheral interface (QuadSPI)

The QuadSPI is integrated with eDMA and has the following general features:

- Boot from QuadSPI flash
- Interface for external quad serial flash memory for code/data storage and code execution
- Supports industry standard: Single, dual and quad mode serial flashes

1.4.16 Serial peripheral interface (SPI)

The chip supports the synchronous serial bus for communication to an external device.

The SPI module is integrated with eDMA and supports the following features:

- Full-duplex, three-wire synchronous transfers
- Buffered transmit operation using the transmit first in first out (TX FIFO) with depth of 16 entries
- Support for 8/16-bit access to the PUSH TX FIFO register data field
- Buffered receive operation using the receive FIFO (RX FIFO) with depth of 16 entries
- Asynchronous clocking scheme for register and protocol interfaces

1.4.17 Watchdog timer (WDOG)

The WDOG monitors internal system operation and forces a reset in case of failure. It operates on RTC 32 KHz clock and module clock (platform clock). LS1012A supports two WDOG timers. Out of two WDOGs, one is dedicated for TrustZone support and another is for A53 core.

Chapter 2
Memory Map

2.1 Overview

There are several address domains within the chip, including the following:

- Logical, virtual, and physical (real) address spaces within the Arm architecture core(s)
- Internal configuration, control, and status register (CCSR) address space, which is a special-purpose subset of the internal local address space

NOTE

SCFG_ALTCBAR refers to the higher address bits of the complete CCSR address space, it refers to 01 as the higher address.

- Internal debug control and status register (DCSR) address space, which is another special-purpose set of registers mapped in the internal local address space
- External memory, I/O, and configuration address spaces of the PCI Express links

2.2 System memory map

This table shows the system memory map of the chip.

Table 2-1. System memory map

Start address (hex)	Module name	Size	Accessible with x-bit addressing		
			32	36	40
0_0000	Secure boot ROM	1 MB	Y	Y	Y
10_0000	Extended Boot ROM	15 MB	Y	Y	Y
100_0000	CCSR register space	240 MB	Y	Y	Y

Table continues on the next page...

Table 2-1. System memory map (continued)

Start address (hex)	Module name	Size	Accessible with x-bit addressing		
			32	36	40
1000_0000	OCRAM1	64 KB	Y	Y	Y
1001_0000	OCRAM2	64 KB	Y	Y	Y
1004_00 00	Reserved	65408 KB	Y	Y	Y
1100_0000	Reserved	16 MB	Y	Y	Y
1200_0000	STM	16 MB	Y	Y	Y
1300_0000	Reserved	208 MB	Y	Y	Y
2000_0000	DCSR	64 MB	Y	Y	Y
4000_0000	QuadSPI	128 MB	Y	Y	Y
5000_0000	Reserved	256 MB	Y	Y	Y
6000_0000	QuadSPI ARDB ADDR (RX Buffer)	64 KB	Y	Y	Y
8000_0000	DRAM1	2 GB	Y	Y	Y
1_0000_0000	Reserved	640 MB	N	Y	Y
1_0400_0000	Reserved	4032 MB	N	Y	Y
2_0000_0000	Reserved	1 GB	N	Y	Y
2_4000_0000	Reserved	7 GB	N	Y	Y
4_0000_0000	Reserved	256 MB	N	Y	Y
4_1000_0000	Reserved	256 MB	N	Y	Y
4_2000_0000	Reserved	256 MB	N	Y	Y
4_3000_0000	Reserved	1280 MB	N	Y	Y
4_8000_0000	Reserved	2 GB	N	Y	Y
5_0000_0000	Reserved	128 MB	N	Y	Y
5_0800_0000	Reserved	128 MB	N	Y	Y
5_1000_0000	Reserved	4 GB	N	Y	Y
6_0000_0000	Reserved	512 MB	N	Y	Y
6_2000_0000	Reserved	3584 MB	N	Y	Y
7_0000_0000	Reserved	4 GB	N	Y	Y
8_0000_0000	Reserved	2 GB	N	Y	Y
8_8000_0000	Reserved	30 GB	N	Y	Y
10_0000_0000	Reserved	64 GB	N	Y	Y
20_0000_0000	Reserved	128 GB	N	N	Y
40_0000_0000	PCI Express 1 (I/O, config and memory space)	32 GB	N	N	Y
48_0000_0000	Reserved	32 GB	N	N	Y
50_0000_0000	Reserved	192 GB	N	N	Y
80_0000_0000	Reserved	32 GB	N	N	Y
88_0000_0000	Reserved	480 GB	N	N	Y

2.3 CCSR address map

The following table lists the block base address assigned to each block from the base of the 240 MB CCSR space along with their corresponding endianness.

NOTE

Arm® Cortex®-A53 endian type can either be little endian or big endian.

Table 2-2. CCSR block base address map

Block base address (hex)	Block	CCSR configuration bus endianness	Comments
100_0000-107_FFFF	Reserved		
108_0000-108_FFFF	DDR memory controller	Big-endian (byte swapping required)	-
109_0000-117_FFFF	Reserved		
118_0000-118_FFFF	Cache coherent interconnect (CCI-400)	Little-endian (byte swapping not required)	-
119_0000-13F_FFFF	Reserved		
140_0000-14F_FFFF	Generic interrupt controller (GIC-400)	Little-endian (byte swapping not required)	-
150_0000-150_FFFF	TZASC	Little-endian (byte swapping not required)	-
151_0000-151_FFFF	CSU	Little-endian (byte swapping required)	-
152_0000-152_FFFF	Platform control	Big-endian (byte swapping required)	-
153_0000-154_FFFF	Reserved		
155_0000-155_FFFF	QuadSPI	Big-endian (byte swapping required)	-
156_0000-156_FFFF	eSDHC controller 1	Big-endian (byte swapping required)	-
157_0000-157_FFFF	SCFG	Big-endian (byte swapping required)	Some bit fields are swapped bitwise when connected to other modules.
158_0000-158_FFFF	eSDHC controller 2	Big-endian (byte swapping required)	-
159_0000-160_FFFF	Reserved		
161_0000-161_FFFF	PBL	Little-endian (byte swapping not required)	-
162_0000-16F_FFFF	Reserved		
170_0000-17F_FFFF	SEC[1]	Big-endian (byte swapping required)	Refer chip security reference manual for details.
180_0000-1E7_FFFF	Reserved		

Table continues on the next page...

Table 2-2. CCSR block base address map (continued)

Block base address (hex)	Block	CCSR configuration bus endianness	Comments
1E8_0000-1E8_FFFF	SFP	Big-endian (byte swapping required)	-
1E9_0000-1E9_FFFF	Security monitor	Little-endian	-
1EA_0000-1EA_FFFF	SerDes	Big-endian (byte swapping required)	-
1EB_0000-1ED_FFFF	Reserved		
1EE_0000-1EE_0FFF	Device configuration and pin control (DCFG)	Big-endian (byte swapping required)	-
1EE_1000-1EE_1FFF	Clocking	Big-endian (byte swapping required)	-
1EE_2000-1EE_2FFF	Run control/power management (RCPM)	Big-endian (byte swapping required)	-
1EE_3000-1EF_FFFF	Reserved		
1F0_0000-1F0_FFFF	Thermal monitoring unit (TMU)	Big-endian (byte swapping required)	-
1F1_0000-20F_FFFF	Reserved		
210_0000-210_FFFF	Serial peripheral interface (SPI)	Big-endian (byte swapping required)	-
211_0000-217_FFFF	Reserved		
218_0000-218_FFFF	I^2C controller 1	Byte accessible	-
219_0000-219_FFFF	I^2C controller 2	Byte accessible	-
21A_0000-21B_FFFF	Reserved		
21C_0000-21C_FFFF	DUART	Byte accessible	
21D_0000-22F_FFFF	Reserved		
230_0000-230_FFFF	GPIO1	Big-endian (byte swapping required)	-
231_0000-231_FFFF	GPIO2	Big-endian (byte swapping required)	-
232_0000-29C_FFFF	Reserved		
29D_0000-29D_FFFF	FlexTimer Module 1 (FTM1)	Big-endian (byte swapping required)	-
29E_0000-29E_FFFF	FlexTimer Module 2 (FTM2)	Big-endian (byte swapping required)	-
29F_0000-2AC_FFFF	Reserved		
2AD_0000-2AD_FFFF	WDOG1	Big-endian (byte swapping required)	-
2AE_0000-2AE_FFFF	WDOG2	Big-endian (byte swapping required)	-
2AF_0000-2AF_FFFF	Reserved		
2B0_0000-2B0_FFFF	System counter (secure)	Big-endian (byte swapping required)	-
2B1_0000-2B1_FFFF	System counter (non-secure)	Big-endian (byte swapping required)	-

Table continues on the next page...

Table 2-2. CCSR block base address map (continued)

Block base address (hex)	Block	CCSR configuration bus endianness	Comments
2B2_0000-2B4_FFFF	Reserved		
2B5_0000-2B5_FFFF	Serial audio interface 1 (SAI1)	Little endian (byte swapping not required)	-
2B6_0000-2B6_FFFF	Serial audio interface 2 (SAI2)	Little endian (byte swapping not required)	
2B7_0000-2B7_FFFF	Serial audio interface 3 (SAI3)	Little endian (byte swapping not required)	
2B8_0000-2B8_FFFF	Serial audio interface 4 (SAI4)	Little endian (byte swapping not required)	
2B9_0000-2B9_FFFF	Serial audio interface 5 (SAI5)	Little endian (byte swapping not required)	
2BA_0000-2BF_FFFF	Reserved		
2C0_0000-2C0_FFFF	eDMA	Big-endian (byte swapping required)	-
2C1_0000-2C1_FFFF	DMACHMUX1	Byte accessible	-
2C2_0000-2C2_FFFF	DMACHMUX2	Byte accessible	-
2C3_0000-2EF_FFFF	Reserved		
2F0_0000-2FF_FFFF	USB3 controller	Little-endian (byte swapping not required)	-
300_0000-31F_FFFF	Reserved		
320_0000-320_FFFF	SATA	Little-endian (byte swapping not required)	-
321_0000-33F_FFFF	Reserved		
340_0000-340_FFFF	PCI Express controller 1 PF0/ shared	Little-endian (byte swapping not required)	Applicable to PCI Express controller register configuration.
341_0000-347_FFFF	Reserved		
348_0000-348_FFFF	PCI Express controller 1 Lookup table	Big-endian (byte swapping required)	-
349_0000-34B_FFFF	Reserved		
34C_0000-34C_FFFF	PCI Express controller 1 PF0 controls	Big-endian (byte swapping required)	-
34D_0000-3FF_FFFF	Reserved		
400_0000-4FF_FFFF	PFE[1]	Big-endian (byte swapping required)	-
500_0000-84E_FFFF	Reserved		
84F_0000-84F_FFFF	USB3 PHY controller	Little-endian (byte swapping not required)	-
850_0000-85F_FFFF	Reserved		
860_0000-860_FFFF	USB2 controller	Big-endian (byte swapping required)	-
861_0000-FFF_FFFF	Reserved		

1. Datapath components should be in mixed endianness. PFE and Security module data structures can be in the following locations:

- CCSR registers
- Frame descriptors such as, data structures shared between software/hardware (in DDR):
    - Arm A53, little-endian mode: Datapath software (at A53) should perform endianness-related byte-swap (for write access, little endian should be swapped to big endian; for read access, big endian should be swapped to little endian) for accessing the datapath components.
    - Arm A53, big-endian mode: Datapath software (at A53) does not perform endianness-related byte-swap for accessing the datapath components.

Chapter 3
Signal Descriptions

3.1 Signals Introduction

This chapter describes the external signals and is organized into the following sections:

- Overview of signals and cross-references for signals that serve multiple functions
- List of reset configuration signals
- Signal multiplexing details
- List of output signal states at reset

3.2 Signals Overview

The signals are grouped as follows:

- DDR SDRAM memory interface signals
- DUART interface signals
- I^2C interface signals
- Enhanced SDHC interface signals
- Serial Peripheral interface (SPI) signals
- Enhanced DMA (eDMA) interface signals
- Ethernet controller RGMII interface signals
- Ethernet management interface (EMI) signals
- HSSI/SerDes signals
- General-purpose input/output, security monitor, system control, power management, and debug signals
- Clock and JTAG signals
- Power-on-Reset configuration signals
- QuadSPI interface signals
- FlexTimer module interface signals
- SAI/I2S module interface signals
- USB controller interface (USB) signals

Note that individual chapters of this document provide details for each signal, describing each signal's behavior when the signal is asserted or negated and when the signal is an input or an output.

The following tables provides a summary of the signals grouped by function. This table details the signal name, interface, alternate functions, number of signals, and whether the signal is an input, output, or bidirectional. The direction of the multiplexed signals applies for the primary signal function listed in the left-most column of the table for that row (and does not apply for the state of the reset configuration signals). Finally, the tables provide a pointer to the table where the signal function is described.

Table 3-1. LS1012A signal reference by functional block

Name	Description	Alternate Function(s)	Pin type
DDR SDRAM Memory Interface (See DDR External Signal Descriptions for more details.)			
D1_MA00	Address	-	O
D1_MA01	Address	-	O
D1_MA02	Address	-	O
D1_MA03	Address	-	O
D1_MA04	Address	-	O
D1_MA05	Address	-	O
D1_MA06	Address	-	O
D1_MA07	Address	-	O
D1_MA08	Address	-	O
D1_MA09	Address	-	O
D1_MA10	Address	-	O
D1_MA11	Address	-	O
D1_MA12	Address	-	O
D1_MA13	Address	-	O
D1_MA14	Address	-	O
D1_MA15	Address	-	O
D1_MBA0	Bank Select	-	O
D1_MBA1	Bank Select	-	O
D1_MBA2	Bank Select	-	O
D1_MCAS_B	Column Address Strobe	-	O
D1_MCK	Clock	-	O
D1_MCKE	Clock Enable	-	O
D1_MCK_B	Clock Complement	-	O
D1_MCS_B	Chip Select	-	O
D1_MDIC	Driver Impedence Calibration	-	IO
D1_MDM0	Data Mask	-	O
D1_MDM1	Data Mask	-	O

Table continues on the next page...

Table 3-1. LS1012A signal reference by functional block (continued)

Name	Description	Alternate Function(s)	Pin type
D1_MDQ00	Data	-	IO
D1_MDQ01	Data	-	IO
D1_MDQ02	Data	-	IO
D1_MDQ03	Data	-	IO
D1_MDQ04	Data	-	IO
D1_MDQ05	Data	-	IO
D1_MDQ06	Data	-	IO
D1_MDQ07	Data	-	IO
D1_MDQ08	Data	-	IO
D1_MDQ09	Data	-	IO
D1_MDQ10	Data	-	IO
D1_MDQ11	Data	-	IO
D1_MDQ12	Data	-	IO
D1_MDQ13	Data	-	IO
D1_MDQ14	Data	-	IO
D1_MDQ15	Data	-	IO
D1_MDQS0	Data Strobe	-	IO
D1_MDQS0_B	Data Strobe	-	IO
D1_MDQS1	Data Strobe	-	IO
D1_MDQS1_B	Data Strobe	-	IO
D1_MODT	On Die Termination	-	O
D1_MRAS_B	Row Address Strobe	-	O
D1_MWE_B	Write Enable	-	O
DUART (See DUART External Signal Descriptions for more details.)			
UART1_SIN	Receive Data	GPIO1_01	I
UART1_SOUT	Transmit Data	GPIO1_00 cfg_eng_use	O
UART2_CTS_B	Clear to send	**TMS** GPIO1_09 SAI5_TX_SYNC SAI5_RX_SYNC	I
UART2_RTS_B	Request to send	**TDI** GPIO1_07 SAI5_TX_DATA SAI5_RX_DATA	O
UART2_SIN	Receive Data	**TCK** GPIO1_06	I
UART2_SOUT	Transmit Data	**TDO** GPIO1_08	O
IIC1 (See IIC External Signal Descriptions for more details.)			
IIC1_SCL	Serial Clock	GPIO1_02 FTM1_CH0	IO

Table continues on the next page...

Table 3-1. LS1012A signal reference by functional block (continued)

Name	Description	Alternate Function(s)	Pin type
IIC1_SDA	Serial Data	GPIO1_03 FTM2_CH0	IO
IIC2 (See IIC External Signal Descriptions for more details.)			
IIC2_SCL	Serial Clock	**QSPI_A_DATA2** GPIO1_13	IO
IIC2_SDA	Serial Data	**QSPI_A_DATA3** GPIO1_14 RESET_REQ_B	IO
QSPI (See QSPI External Signal Descriptions for more details.)			
QSPI_A_CS0	QSPI Chip Select	GPIO1_05 cfg_sysclk_sel	O
QSPI_A_DATA0	QSPI DATA 0	GPIO1_11 cfg_eng_use2	IO
QSPI_A_DATA1	QSPI DATA 1	GPIO1_12 cfg_func_backup	IO
QSPI_A_DATA2	QSPI DATA 2	GPIO1_13 IIC2_SCL	IO
QSPI_A_DATA3	QSPI DATA 3	GPIO1_14 IIC2_SDA RESET_REQ_B	IO
QSPI_A_SCK	QSPI Clock	GPIO1_04	O
Serial Peripheral Interface (See SPI External Signal Descriptions for more details.)			
SPI_CLK	SPI Clock	**SDHC2_CLK** GPIO1_29 FTM2_CH3 SAI1_TX_DATA	O
SPI_CS0_B	Chip Select 0	**SDHC2_DAT0** GPIO1_25 FTM1_CH3 SAI1_RX_SYNC	O
SPI_CS1_B	Chip Select 1	**SDHC2_DAT1** GPIO1_26 FTM2_CH2 SAI1_TX_BCLK	O
SPI_CS2_B	Chip Select 2	**SDHC2_DAT2** GPIO1_27 FTM1_CH2 SAI1_TX_SYNC	O
SPI_MISO	Master in slave out	**SDHC2_DAT3** GPIO1_28 FTM2_CH1 SAI1_RX_BCLK	I
SPI_MOSI	Master out slave in	**SDHC2_CMD** GPIO1_24 FTM1_CH1 SAI1_RX_DATA	O
eSDHC 1 (See eSDHC External Signal Descriptions for more details.)			

Table continues on the next page...

Table 3-1. LS1012A signal reference by functional block (continued)

Name	Description	Alternate Function(s)	Pin type
SDHC1_CLK	Host to Card Clock	GPIO1_20	O
SDHC1_CMD	Command/Response	GPIO1_15	O
SDHC1_DAT0	Data	GPIO1_16	IO
SDHC1_DAT1	Data	GPIO1_17	IO
SDHC1_DAT2	Data	GPIO1_18	IO
SDHC1_DAT3	Data	GPIO1_19	IO
SDHC1_VSEL	SDHC Voltage Select	GPIO1_23	O
SDHC1_CD_B	SDHC Card Detect	GPIO1_21	O
SDHC1_WP	SDHC Write Protect	GPIO1_22	I
eSDHC 2 (See eSDHC External Signal Descriptions for more details.)			
SDHC2_CLK	Host to Card Clock	GPIO1_29 FTM2_CH3 SAI1_TX_DATA SPI_CLK	O
SDHC2_CMD	Command/Response	GPIO1_24 FTM1_CH1 SAI1_RX_DATA SPI_MOSI	IO
SDHC2_DAT0	Data	GPIO1_25 FTM1_CH3 SAI1_RX_SYNC SPI_CS0_B	IO
SDHC2_DAT1	Data	GPIO1_26 FTM2_CH2 SAI1_TX_BCLK SPI_CS1_B	IO
SDHC2_DAT2	Data	GPIO1_27 FTM1_CH2 SAI1_TX_SYNC SPI_CS2_B	IO
SDHC2_DAT3	Data	GPIO1_28 FTM2_CH1 SAI1_RX_BCLK SPI_MISO	IO
System Control (See System Control Signal Descriptions for more details.)			
ASLEEP	ASLEEP	**USB1_PWRFAULT** GPIO2_01	O
PORESET_B	Power On Reset	-	I
RESET_REQ_B	Reset Request	**QSPI_A_DATA3** GPIO1_14 IIC2_SDA	O
RESET_REQ_B	Reset Request	GPIO1_31 cfg_rcw_src **CLK_OUT**	O
TA_TMP_DETECT_B	Tamper Detect	GPIO2_17	I

Table continues on the next page...

Table 3-1. LS1012A signal reference by functional block (continued)

Name	Description	Alternate Function(s)	Pin type
Clocking (See External Clock Signal Descriptions for more details.)			
CLK_OUT	Output clock	GPIO1_31 cfg_rcw_src RESET_REQ_B	O
EXTAL	25Mhz Crystal/Clock Input	-	I
XTAL	Crystal Osc output	-	O
JTAG			
TBSCAN_EN_B	An IEEE 1149.1 JTAG compliance enable pin. 0: To be compliant to the 1149.1 specification for boundary scan functions. The JTAG compliant state is documented in the BSDL. 1: JTAG connects to DAP controller for the Arm core debug.	GPIO1_30 FTM_EXTCLK	I
TCK	Test Clock	GPIO1_06 UART2_SIN	I
TDI	Test Data In	GPIO1_07 UART2_RTS_B SAI5_TX_DATA SAI5_RX_DATA	I
TDO	Test Data Out	GPIO1_08 UART2_SOUT	O
TJTAG_EN	Selection for JTAG IOs	-	I
TMS	Test Mode Select	GPIO1_09 UART2_CTS_B SAI5_TX_SYNC SAI5_RX_SYNC	I
TRST_B	Test Reset	GPIO1_10 SAI5_TX_BCLK SAI5_RX_BCLK	I
SerDes (See SerDes External Signal Descriptions for more details.)			
SD1_IMP_CAL_RX	SerDes Receive Impedence Calibration	-	I
SD1_IMP_CAL_TX	SerDes Transmit Impedance Calibration	-	I
SD1_REF_CLK1_N	SerDes PLL 1 Reference Clock Complement	-	I
SD1_REF_CLK1_P	SerDes PLL 1 Reference Clock	-	I
SD1_RX0_N	SerDes Receive Data (negative)	-	I
SD1_RX0_P	SerDes Receive Data (positive)	-	I
SD1_RX1_N	SerDes Receive Data (negative)	-	I
SD1_RX1_P	SerDes Receive Data (positive)	-	I
SD1_RX2_N	SerDes Receive Data (negative)	-	I
SD1_RX2_P	SerDes Receive Data (positive)	-	I
SD1_TX0_N	SerDes Transmit Data (negative)	-	O
SD1_TX0_P	SerDes Transmit Data (positive)	-	O
SD1_TX1_N	SerDes Transmit Data (negative)	-	O

Table continues on the next page...

Table 3-1. LS1012A signal reference by functional block (continued)

Name	Description	Alternate Function(s)	Pin type
SD1_TX1_P	SerDes Transmit Data (positive)	-	O
SD1_TX2_N	SerDes Transmit Data (negative)	-	O
SD1_TX2_P	SerDes Transmit Data (positive)	-	O
USB PHY (See USB 3.0 External Signal Descriptions for more details.)			
USB1_DRVVBUS	USB PHY Digital signal - Drive VBUS	GPIO2_00	O
USB1_VBUS	USB PHY VBUS	-	O
USB1_D_M	USB PHY Data Minus	-	IO
USB1_D_P	USB PHY Data Plus	-	IO
USB1_ID	USB PHY ID Detect	-	I
USB1_PWRFAULT	USB PHY Digital signal - Power Fault	GPIO2_01 ASLEEP	I
USB1_RX_P	USB PHY SS Receive Data (positive)	-	I
USB1_TX_P	USB PHY SS Transmit Data (positive)	-	O
USB1_RX_M	USB PHY SS Receive Data (negative)	-	I
USB1_TX_M	USB PHY SS Transmit Data (negative)	-	O
USB1_RESREF	USB PHY Impedance Calibration	-	IO
USB 2.0 ULPI (See USB 2.0 External Signal Descriptions for more details.)			
USB2_CLK	USB Clock	**EC1_TX_EN** GPIO2_06 SAI2_TX_SYNC	I
USB2_D0	USB Data	**EC1_TXD3** GPIO2_02 SAI4_TX_DATA SAI4_RX_DATA	IO
USB2_D1	USB Data	**EC1_TXD2** GPIO2_03 SAI3_TX_DATA SAI3_RX_DATA	IO
USB2_D2	USB Data	**EC1_TXD1** GPIO2_04 SAI2_TX_DATA	IO
USB2_D3	USB Data	**EC1_TXD0** GPIO2_05 SAI2_RX_DATA	IO
USB2_D4	USB Data	**EC1_GTX_CLK** GPIO2_07 SAI2_TX_BCLK	IO
USB2_D5	USB Data	**EC1_RX_CLK** GPIO2_13 SAI4_TX_SYNC SAI4_RX_SYNC	IO
USB2_D6	USB Data	**EC1_RXD3** GPIO2_09 SAI2_RX_SYNC	IO

Table continues on the next page...

Table 3-1. LS1012A signal reference by functional block (continued)

Name	Description	Alternate Function(s)	Pin type
USB2_D7	USB Data	**EC1_RXD2** GPIO2_10 SAI2_RX_BCLK	IO
USB2_DIR	USB Direction	**EC1_RX_DV** GPIO2_14 SAI4_TX_BCLK SAI4_RX_BCLK	I
USB2_NXT	USB ULPI next data	**EC1_RXD1** GPIO2_11 SAI3_TX_SYNC SAI3_RX_SYNC	I
USB2_STP	USB ULPI stop	**EC1_RXD0** GPIO2_12 SAI3_TX_BCLK SAI3_RX_BCLK	O
Ethernet Management Interface 1			
EMI1_MDC	Management Data Clock	GPIO2_15	O
EMI1_MDIO	Management Data In/Out	GPIO2_16	IO
Ethernet Controller 1			
EC1_GTX_CLK	Transmit Clock Out	GPIO2_07 SAI2_TX_BCLK USB2_D4	O
EC1_RXD0	Receive Data	GPIO2_12 SAI3_TX_BCLK SAI3_RX_BCLK USB2_STP	I
EC1_RXD1	Receive Data	GPIO2_11 SAI3_TX_SYNC SAI3_RX_SYNC USB2_NXT	I
EC1_RXD2	Receive Data	GPIO2_10 SAI2_RX_BCLK USB2_D7	I
EC1_RXD3	Receive Data	GPIO2_09 SAI2_RX_SYNC USB2_D6	I
EC1_RX_CLK	Receive Clock	GPIO2_13 SAI4_TX_SYNC SAI4_RX_SYNC USB2_D5	I
EC1_RX_DV	Receive Data Valid	GPIO2_14 SAI4_TX_BCLK SAI4_RX_BCLK USB2_DIR	I
EC1_TXD0	Transmit Data	GPIO2_05 SAI2_RX_DATA USB2_D3	O
EC1_TXD1	Transmit Data	GPIO2_04	O

Table continues on the next page...

Table 3-1. LS1012A signal reference by functional block (continued)

Name	Description	Alternate Function(s)	Pin type
		SAI2_TX_DATA USB2_D2	
EC1_TXD2	Transmit Data	GPIO2_03 SAI3_TX_DATA SAI3_RX_DATA USB2_D1	O
EC1_TXD3	Transmit Data	GPIO2_02 SAI4_TX_DATA SAI4_RX_DATA USB2_D0	O
EC1_TX_EN	Transmit Enable	GPIO2_06 SAI2_TX_SYNC USB2_CLK	O
Analog Signals			
D1_MVREF	SSTL Reference Voltage	-	O
TD1_ANODE	Thermal diode anode	-	IO
TD1_CATHODE	Thermal diode cathode	-	IO
General Purpose Input/Output (See GPIO External Signal Descriptions for more details.)			
GPIO1_00	General Purpose Input/Output	**UART1_SOUT** cfg_eng_use	O
GPIO1_01	General Purpose Input/Output	**UART1_SIN**	IO
GPIO1_02	General Purpose Input/Output	**IIC1_SCL** FTM1_CH0	IO
GPIO1_03	General Purpose Input/Output	**IIC1_SDA** FTM2_CH0	IO
GPIO1_04	General Purpose Input/Output	**QSPI_A_SCK**	O
GPIO1_05	General Purpose Input/Output	**QSPI_A_CS0** cfg_sysclk_sel	O
GPIO1_06	General Purpose Input/Output	**TCK** UART2_SIN	IO
GPIO1_07	General Purpose Input/Output	**TDI** UART2_RTS_B SAI5_TX_DATA SAI5_RX_DATA	IO
GPIO1_08	General Purpose Input/Output	**TDO** UART2_SOUT	IO
GPIO1_09	General Purpose Input/Output	**TMS** UART2_CTS_B SAI5_TX_SYNC SAI5_RX_SYNC	IO
GPIO1_10	General Purpose Input/Output	**TRST_B** SAI5_TX_BCLK SAI5_RX_BCLK	IO
GPIO1_11	General Purpose Input/Output	**QSPI_A_DATA0** cfg_eng_use2	O
GPIO1_12	General Purpose Input/Output	**QSPI_A_DATA1**	O

Table continues on the next page...

Table 3-1. LS1012A signal reference by functional block (continued)

Name	Description	Alternate Function(s)	Pin type
		cfg_func_backup	
GPIO1_13	General Purpose Input/Output	**QSPI_A_DATA2** IIC2_SCL	IO
GPIO1_14	General Purpose Input/Output	**QSPI_A_DATA3** IIC2_SDA RESET_REQ_B	IO
GPIO1_15	General Purpose Input/Output	**SDHC1_CMD**	IO
GPIO1_16	General Purpose Input/Output	**SDHC1_DAT0**	IO
GPIO1_17	General Purpose Input/Output	**SDHC1_DAT1**	IO
GPIO1_18	General Purpose Input/Output	**SDHC1_DAT2**	IO
GPIO1_19	General Purpose Input/Output	**SDHC1_DAT3**	IO
GPIO1_20	General Purpose Input/Output	**SDHC1_CLK**	O
GPIO1_21	General Purpose Input/Output	**SDHC1_CD_B**	IO
GPIO1_22	General Purpose Input/Output	**SDHC1_WP**	IO
GPIO1_23	General Purpose Input/Output	**SDHC1_VSEL**	O
GPIO1_24	General Purpose Input/Output	**SDHC2_CMD** FTM1_CH1 SAI1_RX_DATA SPI_MOSI	IO
GPIO1_25	General Purpose Input/Output	**SDHC2_DAT0** FTM1_CH3 SAI1_RX_SYNC SPI_CS0_B	IO
GPIO1_26	General Purpose Input/Output	**SDHC2_DAT1** FTM2_CH2 SAI1_TX_BCLK SPI_CS1_B	IO
GPIO1_27	General Purpose Input/Output	**SDHC2_DAT2** FTM1_CH2 SAI1_TX_SYNC SPI_CS2_B	IO
GPIO1_28	General Purpose Input/Output	**SDHC2_DAT3** FTM2_CH1 SAI1_RX_BCLK SPI_MISO	IO
GPIO1_29	General Purpose Input/Output	**SDHC2_CLK** FTM2_CH3 SAI1_TX_DATA SPI_CLK	O
GPIO1_30	General Purpose Input/Output	**TBSCAN_EN_B** FTM_EXTCLK	IO
GPIO1_31	General Purpose Input/Output	**CLK_OUT** cfg_rcw_src RESET_REQ_B	O
GPIO2_00	General Purpose Input/Output	**USB1_DRVVBUS**	O
GPIO2_01	General Purpose Input/Output	**USB1_PWRFAULT**	IO

Table continues on the next page...

Table 3-1. LS1012A signal reference by functional block (continued)

Name	Description	Alternate Function(s)	Pin type
		ASLEEP	
GPIO2_02	General Purpose Input/Output	**EC1_TXD3** SAI4_TX_DATA SAI4_RX_DATA USB2_D0	IO
GPIO2_03	General Purpose Input/Output	**EC1_TXD2** SAI3_TX_DATA SAI3_RX_DATA USB2_D1	IO
GPIO2_04	General Purpose Input/Output	**EC1_TXD1** SAI2_TX_DATA USB2_D2	IO
GPIO2_05	General Purpose Input/Output	**EC1_TXD0** SAI2_RX_DATA USB2_D3	IO
GPIO2_06	General Purpose Input/Output	**EC1_TX_EN** SAI2_TX_SYNC USB2_CLK	IO
GPIO2_07	General Purpose Input/Output	**EC1_GTX_CLK** SAI2_TX_BCLK USB2_D4	IO
GPIO2_09	General Purpose Input/Output	**EC1_RXD3** SAI2_RX_SYNC USB2_D6	IO
GPIO2_10	General Purpose Input/Output	**EC1_RXD2** SAI2_RX_BCLK USB2_D7	IO
GPIO2_11	General Purpose Input/Output	**EC1_RXD1** SAI3_TX_SYNC SAI3_RX_SYNC USB2_NXT	IO
GPIO2_12	General Purpose Input/Output	**EC1_RXD0** SAI3_TX_BCLK SAI3_RX_BCLK USB2_STP	IO
GPIO2_13	General Purpose Input/Output	**EC1_RX_CLK** SAI4_TX_SYNC SAI4_RX_SYNC USB2_D5	IO
GPIO2_14	General Purpose Input/Output	**EC1_RX_DV** SAI4_TX_BCLK SAI4_RX_BCLK USB2_DIR	IO
GPIO2_15	General Purpose Input/Output	**EMI1_MDC**	O
GPIO2_16	General Purpose Input/Output	**EMI1_MDIO**	IO
GPIO2_17	General Purpose Input/Output	**TA_TMP_DETECT_B**	I
Power-On-Reset Configuration (See Reset Configuration Signal Descriptions for more details.)			

Table continues on the next page...

Table 3-1. LS1012A signal reference by functional block (continued)

Name	Description	Alternate Function(s)	Pin type
cfg_eng_use	Power-on-Reset Configuration	**UART1_SOUT** GPIO1_00	I
cfg_eng_use2	Power-on-Reset Configuration	**QSPI_A_DATA0** GPIO1_11	I
cfg_func_backup	backup	**QSPI_A_DATA1** GPIO1_12	I
cfg_rcw_src	Power-on-Reset Configuration	**CLK_OUT** GPIO1_31 RESET_REQ_B	I
cfg_sysclk_sel	Power-on-Reset Configuration	**QSPI_A_CS0** GPIO1_05	I
Frequency Timer Module 1 (See FTM External Signal Descriptions for more details.)			
FTM1_CH0	Channel 0	**IIC1_SCL** GPIO1_02	IO
FTM1_CH1	Channel 1	**SDHC2_CMD** GPIO1_24 SAI1_RX_DATA SPI_MOSI	IO
FTM1_CH2	Channel 2	**SDHC2_DAT2** GPIO1_27 SAI1_TX_SYNC SPI_CS2_B	IO
FTM1_CH3	Channel 3	**SDHC2_DAT0** GPIO1_25 SAI1_RX_SYNC SPI_CS0_B	IO
FTM_EXTCLK	External Clock	**TBSCAN_EN_B** GPIO1_30	I
Frequency Timer Module 2 (See FTM External Signal Descriptions for more details.)			
FTM2_CH0	Channel 0	**IIC1_SDA** GPIO1_03	IO
FTM2_CH1	Channel 1	**SDHC2_DAT3** GPIO1_28 SAI1_RX_BCLK SPI_MISO	IO
FTM2_CH2	Channel 2	**SDHC2_DAT1** GPIO1_26 SAI1_TX_BCLK SPI_CS1_B	IO
FTM2_CH3	Channel 3	**SDHC2_CLK** GPIO1_29 SAI1_TX_DATA SPI_CLK	O
Synchronous Audio Interfaces (See SAI External Signal Descriptions for more details.)			
SAI1_RX_BCLK	SAI Receive Clock	**SDHC2_DAT3** GPIO1_28 FTM2_CH1	I

Table continues on the next page...

Table 3-1. LS1012A signal reference by functional block (continued)

Name	Description	Alternate Function(s)	Pin type
		SPI_MISO	
SAI1_RX_DATA	SAI Receive Data	**SDHC2_CMD** GPIO1_24 FTM1_CH1 SPI_MOSI	I
SAI1_RX_SYNC	SAI Receive Sync	**SDHC2_DAT0** GPIO1_25 FTM1_CH3 SPI_CS0_B	IO
SAI1_TX_BCLK	SAI Transmit Clock	**SDHC2_DAT1** GPIO1_26 FTM2_CH2 SPI_CS1_B	I
SAI1_TX_DATA	SAI Transmit Data	**SDHC2_CLK** GPIO1_29 FTM2_CH3 SPI_CLK	O
SAI1_TX_SYNC	SAI Transmit Sync	**SDHC2_DAT2** GPIO1_27 FTM1_CH2 SPI_CS2_B	IO
SAI2_RX_BCLK	SAI Receive Clock	**EC1_RXD2** GPIO2_10 USB2_D7	I
SAI2_RX_DATA	SAI Receive Data	**EC1_TXD0** GPIO2_05 USB2_D3	I
SAI2_RX_SYNC	SAI Receive Sync	**EC1_RXD3** GPIO2_09 USB2_D6	IO
SAI2_TX_BCLK	SAI Transmit Clock	**EC1_GTX_CLK** GPIO2_07 USB2_D4	I
SAI2_TX_DATA	SAI Transmit Data	**EC1_TXD1** GPIO2_04 USB2_D2	O
SAI2_TX_SYNC	SAI Transmit Sync	**EC1_TX_EN** GPIO2_06 USB2_CLK	IO
SAI3_RX_BCLK / SAI3_TX_BCLK	SAI Receive Clock	**EC1_RXD0** GPIO2_12 USB2_STP	I
SAI3_RX_DATA / SAI3_TX_DATA	SAI Receive Data	**EC1_TXD2** GPIO2_03 USB2_D1	I
SAI3_RX_SYNC / SAI3_TX_SYNC	SAI Receive Sync	**EC1_RXD1** GPIO2_11 USB2_NXT	IO

Table continues on the next page...

Table 3-1. LS1012A signal reference by functional block (continued)

Name	Description	Alternate Function(s)	Pin type
SAI3_TX_BCLK / SAI3_RX_BCLK	SAI Transmit Clock	**EC1_RXD0** GPIO2_12 USB2_STP	I
SAI3_TX_DATA / SAI3_RX_DATA	SAI Transmit Data	**EC1_TXD2** GPIO2_03 USB2_D1	O
SAI3_TX_SYNC / SAI3_RX_SYNC	SAI Transmit Sync	**EC1_RXD1** GPIO2_11 USB2_NXT	IO
SAI4_RX_BCLK / SAI4_TX_BCLK	SAI Receive Clock	**EC1_RX_DV** GPIO2_14 USB2_DIR	I
SAI4_RX_DATA / SAI4_TX_DATA	SAI Receive Data	**EC1_TXD3** GPIO2_02 USB2_D0	I
SAI4_RX_SYNC / SAI4_TX_SYNC	SAI Receive Sync	**EC1_RX_CLK** GPIO2_13 USB2_D5	IO
SAI4_TX_BCLK / SAI4_RX_BCLK	SAI Transmit Clock	**EC1_RX_DV** GPIO2_14 USB2_DIR	I
SAI4_TX_DATA / SAI4_RX_DATA	SAI Transmit Data	**EC1_TXD3** GPIO2_02 USB2_D0	O
SAI4_TX_SYNC / SAI4_RX_SYNC	SAI Transmit Sync	**EC1_RX_CLK** GPIO2_13 USB2_D5	IO
SAI5_RX_BCLK / SAI5_TX_BCLK	SAI Receive Clock	**TRST_B** GPIO1_10	I
SAI5_RX_DATA / SAI5_TX_DATA	SAI Receive Data	**TDI** GPIO1_07 UART2_RTS_B	I
SAI5_RX_SYNC / SAI5_TX_SYNC	SAI Receive Sync	**TMS** GPIO1_09 UART2_CTS_B	IO
SAI5_TX_BCLK / SAI5_RX_BCLK	SAI Transmit Clock	**TRST_B** GPIO1_10	I
SAI5_TX_DATA / SAI5_RX_DATA	SAI Transmit Data	**TDI** GPIO1_07 UART2_RTS_B	O
SAI5_TX_SYNC / SAI5_RX_SYNC	SAI Transmit Sync	**TMS** GPIO1_09 UART2_CTS_B	IO

NOTE

Refer Differences between silicon revisions 1.0 and 2.0 for the signal multiplexing changes in silicon revision 1.0.

3.3 Configuration signals sampled at reset

The signals that serve alternate functions as configuration input signals during system reset are summarized in this table.

Note that throughout this document, the reset configuration signals are described as being sampled at the negation of PORESET_B. However, there is a setup and hold time for these signals relative to the rising edge of PORESET_B, as described in the chip's data sheet document.

The reset configuration signals are multiplexed with other functional signals. The values on these signals during reset are interpreted to be logic one or zero, regardless of whether the functional signal name is defined as active-low. The reset configuration signals have internal pull-up resistors so that if the signals are not driven, the default value is high (a one), as shown in the table. Some signals must be driven high or low during the reset period. For details about all the signals that require external pull-up resistors, see the data sheet document. Refer POR status register 1 (PORSR1) and POR Status Register 2 (PORSR2) for the definition of reset configuration signals.

Table 3-2. Reset configuration signals

Reset configuration name	Functional interface	Functional Signal Name	Default	Reference
cfg_rcw_src	Clocking	CLK_OUT	1	Refer Reset configuration word (RCW) source for detailed description.
cfg_eng_use	DUART	UART1_SOUT	1	Refer Transconductance selection control for detailed description.
cfg_eng_use2	QuadSPI	QSPI_A_DATA0	1	
cfg_func_backup	QuadSPI	QSPI_A_DATA1	1	Refer Clocking mode for detailed description.
cfg_sysclk_sel	QuadSPI	QSPI_A_CS0	1	

3.4 Signal multiplexing details

Due to the extensive functionality present on the chip and the limited number of external signals available, several functional blocks share signal resources through multiplexing. These signals are designated in the Alternate function(s) column of Table 3-1. In this case when there is alternate functionality between multiple functional blocks, the signal's function is determined at the device level (rather than at the block level) typically by a reset configuration word (RCW) option. For example, the signal QSPI_A_DATA[2:3]/

IIC2_SCL, IIC2_SDA/GPIO1[13:14] can be utilized by either the QuadSPI block, the I^2C block, or by GPIO1. Because these signals alternatively service three different functional blocks, their function is determined at the device level by RCW[QSPI_IIC2]. The following sections describe external signal functional selection using the RCW.

3.4.1 Ethernet controller 1, SAI, USB 2.0, and GPIO2 signal multiplexing

The Ethernet controller 1 (EC1) RGMII interface shares signals with SAI, USB 2.0, and GPIO2. The functionality of these signals is determined by the RCW[EC1_BASE], RCW[EC1_EXT_SAI2_RX], RCW[EC1_EXT_SAI2_TX], RCW[EC1_EXT_SAI3], and RCW[EC1_EXT_SAI4] fields.

Table 3-3. EC1 RGMII signal configuration

Signal name	Signal function	RCW[EC1_EXT_*]	RCW[EC1_BASE]
		RCW[EC1_EXT_SAI2_RX]	RCW[EC1_BASE]
EC1_TXD0	GPIO2[5]	0	00
	SAI2_RX_DATA	1	
	EC1_TXD0	-	01
	USB2_D3		10
	Reserved		11
EC1_RXD3	GPIO2[9]	0	00
	SAI2_RX_SYNC	1	
	EC1_RXD3	-	01
	USB2_D6		10
	Reserved		11
EC1_RXD2	GPIO2[10]	0	00
	SAI2_RX_BCLK	1	
	EC1_RXD2	-	01
	USB2_D7		10
	Reserved		11
		RCW[EC1_EXT_SAI2_TX]	RCW[EC1_BASE]
EC1_TXD1	GPIO2[4]	0	00
	SAI2_TX_DATA	1	
	EC1_TXD1	-	01
	USB2_D2		10
	Reserved		11
EC1_TX_EN	GPIO2[6]	0	00
	SAI2_TX_SYNC	1	
	EC1_TX_EN	-	01

Table continues on the next page...

Table 3-3. EC1 RGMII signal configuration (continued)

Signal name	Signal function	RCW[EC1_EXT_*]	RCW[EC1_BASE]
	USB2_CLK		10
	Reserved		11
EC1_GTX_CLK	GPIO2[7]	0	00
	SAI2_TX_BCLK	1	
	EC1_GTX_CLK	-	01
	USB2_D4		10
	Reserved		11
		RCW[EC1_EXT_SAI3]	RCW[EC1_BASE]
EC1_TXD2	GPIO2[3]	00	00
	SAI3_TX_DATA	01	
	SAI3_RX_DATA	10	
	Reserved	11	
	EC1_TXD2	-	01
	USB2_D1		10
	Reserved		11
EC1_RXD1	GPIO2[11]	00	00
	SAI3_TX_SYNC	01	
	SAI3_RX_SYNC	10	
	Reserved	11	
	EC1_RXD1	-	01
	USB2_NXT		10
	Reserved		11
EC1_RXD0	GPIO2[12]	00	00
	SAI3_TX_BCLK	01	
	SAI3_RX_BCLK	10	
	Reserved	11	
	EC1_RXD0	-	01
	USB2_STP		10
	Reserved		11
		RCW[EC1_EXT_SAI4]	RCW[EC1_BASE]
EC1_TXD3	GPIO2[2]	00	00
	SAI4_TX_DATA	01	
	SAI4_RX_DATA	10	
	Reserved	11	
	EC1_TXD3	-	01
	USB2_D0		10
	Reserved		11
EC1_RX_CLK	GPIO2[13]	00	00
	SAI4_TX_SYNC	01	

Table continues on the next page...

Table 3-3. EC1 RGMII signal configuration (continued)

Signal name	Signal function	RCW[EC1_EXT_*]	RCW[EC1_BASE]
	SAI4_RX_SYNC	10	
	Reserved	11	
	EC1_RX_CLK	-	01
	USB2_D5		10
	Reserved		11
EC1_RX_DV	GPIO2[14]	00	00
	SAI4_TX_BCLK	01	
	SAI4_RX_BCLK	10	
	Reserved	11	
	EC1_RX_DV	-	01
	USB2_DIR		10
	Reserved		11

3.4.2 Ethernet management interface 1 and GPIO2 signal multiplexing

The Ethernet management interface 1 (EMI1) shares signals with GPIO2. The functionality of these signals is determined by the RCW[EMI1_BASE] field.

Table 3-4. EMI1 signal configuration

Signal name	Signal function	RCW[EMI1_BASE]
EMI1_MDC	GPIO2[15]	0
	EMI1_MDC	1
EMI1_MDIO	GPIO2[16]	0
	EMI1_MDIO	1

3.4.3 eSDHC1 and GPIO1 signal multiplexing

The eSDHC1 interface shares signals with GPIO1. The functionality of these signals are determined by the RCW[SDHC1_BASE], RCW[SDHC1_CD], RCW[SDHC1_WP], and RCW[SDHC1_VSEL] fields.

Table 3-5. SDHC_BASE signal configuration

Signal name	Signal function	RCW[SDHC1_BASE] / RCW[SDHC1_CD] / RCW[SDHC1_WP] / RCW[SDHC1_VSEL]
		RCW[SDHC1_BASE]
SDHC1_CMD	GPIO1[15]	00
	SDHC1_CMD	01
	SDHC1_CMD	10
	Reserved	11
SDHC1_DAT0	GPIO1[16]	00
	SDHC1_DAT0	01
	SDHC1_DAT0	10
	Reserved	11
SDHC1_DAT1	GPIO1[17]	00
	SDHC1_DAT1	01
	SDHC1_CLK	10
	Reserved	11
SDHC1_DAT2	GPIO1[18]	00
	SDHC1_DAT2	01
	GPIO1[18]	10
	Reserved	11
SDHC1_DAT3	GPIO1[19]	00
	SDHC1_DAT3	01
	GPIO1[19]	10
	Reserved	11
SDHC1_CLK	GPIO1[20]	00
	SDHC1_CLK	01
	GPIO1[20]	10
	Reserved	11
		RCW[SDHC1_CD]
SDHC1_CD_B	GPIO1[21]	0
	SDHC1_CD_B	1
		RCW[SDHC1_WP]
SDHC1_WP	GPIO1[22]	0
	SDHC1_WP	1
		RCW[SDHC1_VSEL]
SDHC_VSEL	GPIO1[23]	0
	SDHC1_VSEL	1

3.4.4 eSDHC2, GPIO1, FTM, SAI, and SPI signal multiplexing

The eSDHC2 interface shares signals with GPIO1, FTM, SAI, and SPI. The functionality of these signals are determined by the RCW[SDHC2_BASE_DAT321], RCW[SDHC2_BASE_BASE], and RCW[SDHC2_EXT_*] fields.

Table 3-6. SDHC2_BASE signal configuration

Signal name	Signal function	RCW[SDHC2_EXT_*]	RCW[SDHC2_BASE_DAT321] / RCW[SDHC2_BASE_BASE]
		RCW[SDHC2_EXT_DAT3]	RCW[SDHC2_BASE_DAT321]
SDHC2_DAT3	GPIO1[28]	0	00
	FTM2_CH1	1	
	SDHC2_DAT3	-	01
	SAI1_RX_BCLK		10
	SPI_MISO		11
		RCW[SDHC2_EXT_DAT2]	RCW[SDHC2_BASE_DAT321]
SDHC2_DAT2	GPIO1[27]	0	00
	FTM1_CH2	1	
	SDHC2_DAT2	-	01
	SAI1_TX_BCLK		10
	SPI_CS2_B		11
		RCW[SDHC2_EXT_DAT1]	RCW[SDHC2_BASE_DAT321]
SDHC2_DAT1	GPIO1[26]	0	00
	FTM2_CH2	1	
	SDHC2_DAT1		01
	SAI1_TX_SYNC		10
	SPI_CS1_B		11
		RCW[SDHC2_EXT_CMD]	RCW[SDHC2_BASE_BASE]
SDHC2_CMD	GPIO1[24]	0	00
	FTM1_CH1	1	
	SDHC2_CMD		01
	SAI1_RX_DATA		10
	SPI_MOSI		11
		RCW[SDHC2_EXT_DAT0]	RCW[SDHC2_BASE_BASE]
SDHC2_DAT0	GPIO1[25]	0	00
	FTM1_CH3	1	
	SDHC2_DAT0		01

Table continues on the next page...

Table 3-6. SDHC2_BASE signal configuration (continued)

Signal name	Signal function	RCW[SDHC2_EXT_*]	RCW[SDHC2_BASE_DAT321] / RCW[SDHC2_BASE_BASE]
	SAI1_RX_SYNC		10
	SPI_CS0_B		11
		RCW[SDHC2_EXT_CLK]	RCW[SDHC2_BASE_BASE]
SDHC2_CLK	GPIO1[29]	0	00
	FTM2_CH3	1	
	SDHC2_CLK		01
	SAI1_TX_DATA		10
	SPI_CLK		11

3.4.5 GPIO1 and FTM signal multiplexing

The GPIO1 interface shares signals with FTM. The functionality of these signals is determined by the RCW[GPIO_FTM_EXTCLK_BASE].

Table 3-7. I²C1 signal configuration

Signal name	Signal function	RCW[GPIO_FTM_EXTCLK_BASE]
TBSCAN_EN_B[1]	GPIO1[30]	00
	Reserved	01
	FTM_EXTCLK	10
	Reserved	11

1. If TJTAG_EN=1, this pin is used as TBSCAN_EN_B. If JTAG_EN_B=0, the pins are used as defined in the bit settings.

3.4.6 I²C1, GPIO1, and FTM signal multiplexing

The I²C1 interface shares signals with GPIO1 and FTM. The functionality of these signals is determined by the RCW[IIC1_BASE].

Table 3-8. I²C1 signal configuration

Signal name	Signal function	RCW[IIC1_BASE]
IIC1_SCL	GPIO1[2]	00
	IIC1_SCL	01
	FTM1_CH0	10

Table continues on the next page...

Table 3-8. I²C1 signal configuration (continued)

Signal name	Signal function	RCW[IIC1_BASE]
	Reserved	11
IIC1_SDA	GPIO1[3]	00
	IIC1_SDA	01
	FTM2_CH0	10
	Reserved	11

3.4.7 QuadSPI, I2C, and GPIO1 signal multiplexing

The QuadSPI controller shares signals with I2C and GPIO2. The functionality of these signals is determined by the RCW[QSPI_DATA0_GPIO], RCW[QSPI_DATA1_GPIO], and RCW[QSPI_IIC2] fields as described in the following table.

NOTE

The QuadSPI multiplexing can be changed after boot by the software to alternate function by programming Pinmux control register (PMUXCR0).

Table 3-9. QuadSPI signal configuration

Signal name	Signal function	RCW[QSPI_DATA0_GPIO] / RCW[QSPI_DATA1_GPIO] / RCW[QSPI_IIC2]
		RCW[QSPI_DATA0_GPIO]
QSPI_A_DATA0	QSPI_A_DATA0	0
	GPIO1[11]	1
QSPI_A_SCK	QSPI_A_SCK	0
	GPIO1[4]	1
QSPI_A_CS0	QSPI_A_CS0	0
	GPIO1[5]	1
		RCW[QSPI_DATA1_GPIO]
QSPI_A_DATA1	QSPI_A_DATA1	00
	GPIO1[12]	01
	Reserved	10
	Reserved	11
		RCW[QSPI_IIC2]
QSPI_A_DATA2	GPIO1[13]	00
	IIC2_SCL	01
	QSPI_A_DATA2	10
	GPIO1[13]	11

Table continues on the next page...

Table 3-9. QuadSPI signal configuration (continued)

QSPI_A_DATA3	GPIO1[14]	00
	IIC2_SDA	01
	QSPI_A_DATA3	10
	RESET_REQ_B	11

3.4.8 UART1 and GPIO signal multiplexing

The functionality of these signals is determined by the RCW[UART1_BASE]).

Table 3-10. UART signal configuration

Signal name	Signal function	RCW[UART1_BASE]
UART1_SOUT	GPIO1[0]	00
	UART1_SOUT	01
	Reserved	10
	Reserved	11
UART1_SIN	GPIO1[1]	00
	UART1_SIN	01
	Reserved	10
	Reserved	11

3.4.9 UART2, GPIO, and SAI signal multiplexing

The functionality of these signals is determined by the RCW[UART2_BASE_DATA] and RCW[UART2_BASE_FLOW]).

Table 3-11. UART2 signal configuration

Signal name	Signal function	RCW[UART2_BASE_DATA] / RCW[UART2_BASE_FLOW]
		RCW[UART2_BASE_DATA][1]
TDO	GPIO1[8]	0
	UART2_SOUT	1
TCK	GPIO1[6]	0
	UART2_SIN	1
		RCW[UART2_BASE_FLOW][1]
TDI	GPIO1[7]	00
	UART2_RTS_B	01

Table continues on the next page...

Table 3-11. UART2 signal configuration (continued)

Signal name	Signal function	RCW[UART2_BASE_DATA] / RCW[UART2_BASE_FLOW]
	SAI5_TX_DATA	10
	SAI5_RX_DATA	11
TMS	GPIO1[9]	01
	UART2_CTS_B	01
	SAI5_TX_SYNC	10
	SAI5_RX_SYNC	11
TRST_B	GPIO1[10]	00
	GPIO1[10]	01
	SAI5_TX_BCLK	10
	SAI5_RX_BCLK	11

1. This field is ignored if TJTAG_EN=1, in which case these pins are used as JTAG.

3.4.10 USB, ASLEEP, and GPIO1 signal multiplexing

The USB controller 1 shares signals with the power management signal (ASLEEP) and GPIO2. The functionality of these signals are determined by the RCW[USB1_DRVVBUS_ BASE] and RCW[USB1_PWRFAULT _BASE] fields.

Table 3-12. USB signal configuration

Signal name	Signal function	RCW[USB1_DRVVBUS_ BASE] / RCW[USB1_PWRFAULT _BASE]
		RCW[USB1_DRVVBUS_ BASE]
USB1_DRVVBUS	GPIO2[0]	00
	USB1_DRVVBUS	01
	Reserved	10
	Reserved	11
		RCW[USB1_PWRFAULT _BASE]
USB1_PWRFAULT	GPIO2[1]	00
	USB1_PWRFAULT	01
	ASLEEP	10
	Reserved	11

3.4.11 TA_TMP_DETECT_B and GPIO2 signal multiplexing

The TA_TMP_DETECT_B shares signals with GPIO2. The functionality of these signals is determined by the ITS fuse.

Table 3-13. TA_TMP_DETECT_B signal configuration

Signal name	Signal function	ITS fuse
TA_TMP_DETECT_B	GPIO2[17]	0
	TA_TMP_DETECT_B	1

3.5 Output Signal States During Reset

When a system reset is initiated (PORESET_B sampled asserted by the chip), the chip aborts all current internal and external transactions and releases the bidirectional I/O signals to a high-impedance state. However, some signals get stable values at power-on reset.

While the chip is in reset, it ignores most input signals (except for the reset configuration signals) and drives output-only signals D1_MCKE and D1_MODT to an inactive state.

Note that signals associated with all potential RCW source interfaces are enabled and active while the chip is in reset (that is, after PORESET_B is deasserted). This is necessary to allow the interfaces to be used for fetching configuration information from non-volatile memory devices.

Chapter 4
Reset, Clocking, and Initialization

4.1 Reset, clocking, and initialization overview

This content describes the reset, clocking, and initialization, including a definition of the reset configuration signals and the options they select.Note that other chapters in this book may describe specific aspects of initialization for individual blocks.

The reset, clocking, and control signals provide many options for operation. Additionally, many modes are selected with reset configuration signals during a power on reset (assertion of PORESET_B) and by using the reset configuration word (RCW) functionality.

4.2 External Signal Descriptions

The table below summarizes the external signals described in this chapter.

Table 4-3 and Table 4-4 have detailed signal descriptions, but this table contains references to additional sections that contain more information.

Table 4-1. Reset and Control Signals Summary

Signal	I/O	Description
PORESET_B	I	Power on reset input.
RESET_REQ_B (if selected)	O	Reset request output. An internal block requests that PORESET_B be asserted. **NOTE:** The following scenarios require RESET_REQ_B signal under which the system may not be recoverable: • Core watchdog expiry • Uncorrectable ECC errors • Boot and pre-boot errors

Table continues on the next page...

Table 4-1. Reset and Control Signals Summary (continued)

Signal	I/O	Description
		Since PORESET is the only reset supported in the chip, this eventually triggers PORESET and hence the source of violation is not detectable.
ASLEEP (if selected)	O	Asleep
TA_TMP_DETECT_B	I	Tamper Detect.

Table 4-2. Clock Signals Summary

Signal	I/O	Description
SD_REF_CLK*n*_P/SD_REF_CLK*n*_N (if selected)	I	SerDes high-speed interface reference clock *n*.
CLK_OUT	O	Diagnostic clock output.

The following sections describe the reset and clock signals in detail.

4.2.1 System control signals

The table below describes some of the system control signals. Power-on reset configuration describes the signals that also function as reset configuration signals.

Table 4-3. System control signals: Detailed signal descriptions

Signal	I/O	Description	
PORESET_B	I	Power on reset. Causes the chip to abort all current internal and external transactions and set all registers to their default values. PORESET_B may be asserted completely asynchronously with respect to all other signals.	
		State Meaning	Asserted/Negated-See the Signal Descriptions chapter and the Power-on reset configuration section in the Reset chapter for more information on the interpretation of the other signals during reset.
		Timing	Assertion/Negation-The chip data sheet gives specific timing information for this signal and the reset configuration signals.
RESET_REQ_B	O	Reset request. It is not a primary pin but is recommended to select this pin for trust architecture. There are some restrictions as RESET_REQ_B is selected through RCW. Refer to RESET_REQ_B behavior for details.. This signal indicates to the board (system in which the chip is embedded) that a condition requiring the assertion of PORESET_B has been detected. **NOTE:** Refer Differences between silicon revisions 1.0 and 2.0 for the implementation in silicon revision 1.0.	
		State Meaning	Asserted-An event has triggered a request for a power on reset. See the DCFG_RSTRQSR1 register in the DCFG chapter for more information. Negated-Indicates no reset request.

Table continues on the next page...

Table 4-3. System control signals: Detailed signal descriptions (continued)

Signal	I/O	Description	
		Timing	Assertion/Negation-May occur any time. Once asserted, RESET_REQ_B does not negate until PORESET_B is asserted.
ASLEEP	O	Power Management Signal (optional). See external signal description section in the RCPM chapter for details.	
TA_TMP_DETECT_B	I	Tamper Detect.	

4.2.1.1 RESET_REQ_B behavior

The following scenarios, in which the system is not recoverable, assert the RESET_REQ_B signal:

- Security violation
- Core watchdog expiry
- Uncorrectable ECC errors
- Boot and pre-boot errors

The RESET_REQ_B is multiplexed on secondary function. For trust architecture, it is a requirement to implement the RESET_REQ_B. The RESET_REQ_B can be selected through RCW on:

- CLK_OUT or
- the multiplexed RESET_REQ_B/QSPI/I2C/GPIO pin

Typically, RESET_REQ_B is used to trigger a reset using PORESET_B, note that this erases the source of the violation.

NOTE

Since RESET_REQ_B is selected through RCW, if there is any issue during or before RCW loading, the chip is not capable to handle it as the RESET_REQ_B is not yet selected.

NOTE

Refer Differences between silicon revisions 1.0 and 2.0 for the implementation in silicon revision 1.0.

4.2.2 External Clock Signals

The table below describes some of the external clock signals of the chip.

Note that some clock signals are specific to modules within the chip, and although some of their functionality is described here, they are defined in detail in their respective chapters.

Table 4-4. Clock External Signals-Detailed Signal Descriptions

Signal	I/O	Description
SD_REF_CLK*n*_P, SD_REF_CLK*n*_N	I	SerDes high-speed interface differential reference clocks. These differential clock inputs are used to independently clock the banks/ports of high-speed differential signal lanes available on the chip. The SerDes reference clock timing specifications are given in the chip data sheet . See the Reference Clocks for SerDes Protocols section in the SerDes Module chapter. **NOTE:** The SerDes clock is internally generated and the external clock is an alternate source. **Timing** — Assertion/Negation-See the chip data sheet for specific timing information for these signals.
CLK_OUT	O	Diagnostic clock output. This output may be configured to offer one of a variety of internal system clocks to external hardware for diagnostic or debug purposes. See CLK_OUT configuration.

4.3 Clocking register descriptions

The following table summarizes the memory mapped registers which are used to configure clocking features.

4.3.1 Clocking Memory map

Clocking base address: 1EE_1000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Core cluster n clock control/status register (CLKC1CSR)	32	RW	0000_0000h
800h	PLL cluster n general status register (PLLC1GSR)	32	RW	0000_0000h
A00h	Platform clock domain control/status register (CLKPCSR)	32	RW	0000_0000h
C00h	Platform PLL general status register (PLLPGSR)	32	RO	0000_0000h

4.3.2 Core cluster n clock control/status register (CLKC1CSR)

4.3.2.1 Offset

Register	Offset
CLKC1CSR	0h

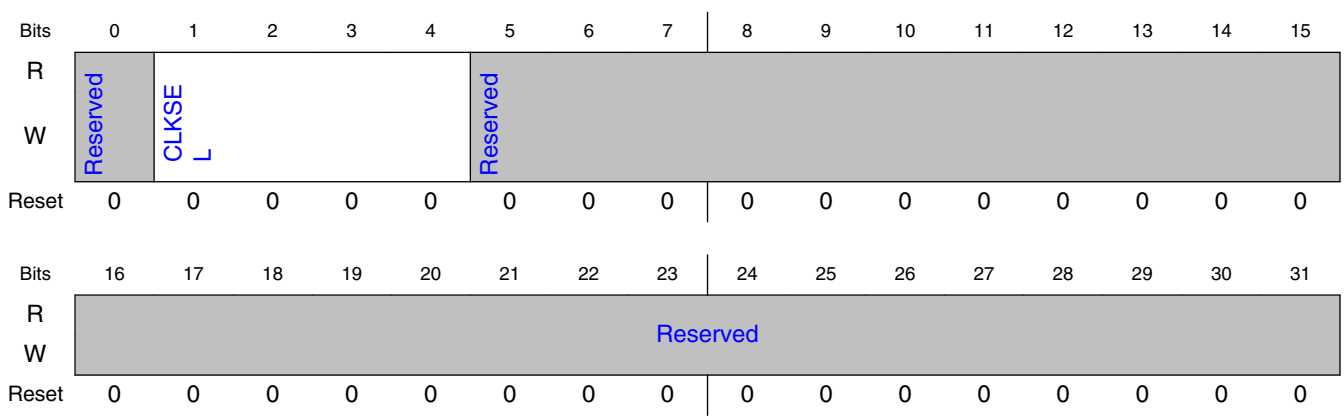
4.3.2.2 Function

The CLKC*n*CSR registers control the clock frequency selection for each core cluster.

Table 4-5. Core cluster assignments to CLKC*n*CSR registers

Register	Cluster Group	Core Cluster
CLKC1CSR	Cluster Group A (CGA)	Core Cluster 1

4.3.2.3 Diagram



4.3.2.4 Fields

Field	Function
0 —	- Reserved
1-4 CLKSEL	CLKSEL Clock Select. Selects the clock source for the corresponding core cluster. See Table 4-5 above. 0000b - Corresponding cluster group PLL1 output 0001b - Reserved

Table continues on the next page...

Field	Function
	0010b - Corresponding cluster group PLL1 output divide-by-2 0011b - Reserved 0111-1111b - Reserved
5-31 —	- Reserved

4.3.3 PLL cluster n general status register (PLLC1GSR)

4.3.3.1 Offset

Register	Offset
PLLC1GSR	800h

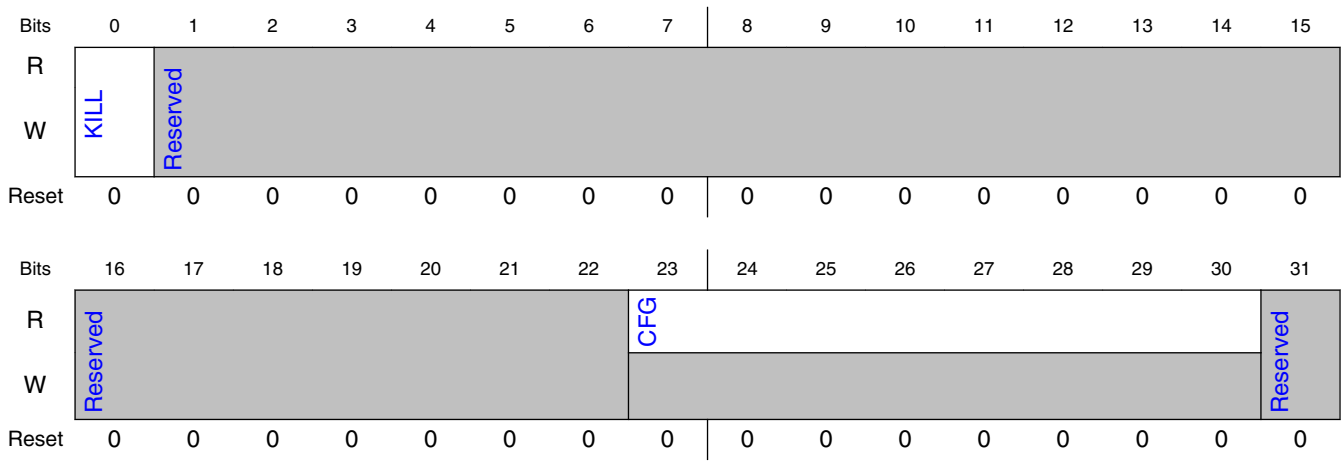
4.3.3.2 Function

The PLLC*n*GSR registers provide information regarding the cluster PLL configuration.

Table 4-6. PLL assignments to PLLC*n*GSR registers

Register	Cluster Group	PLL
PLLC1GSR	CGA	PLL1

4.3.3.3 Diagram



4.3.3.4 Fields

Field	Function
0 KILL	KILL Writing a 1 to this bit disables the PLL. If PLL*n*GSR[CFG] indicates 0b00000, the PLL is bypassed and this bit (KILL) is set. 0b - PLL is active. 1b - PLL is disabled.
1-22 —	- Reserved
23-30 CFG	CFG Reflects the current PLL multiplier configuration. Indicates the frequency for this PLL. PLLC1GSR[CFG] reflects the values programmed in RCW[CGA_PLL1_RAT]. Defined settings are from 00_0101 (5:1) through 10_1000 (40:1). All other encodings are reserved. **NOTE:** Not all ratios are supported due to frequency restrictions. Refer to the chip data sheet for the supported frequencies.
31 —	- Reserved

4.3.4 Platform clock domain control/status register (CLKPCSR)

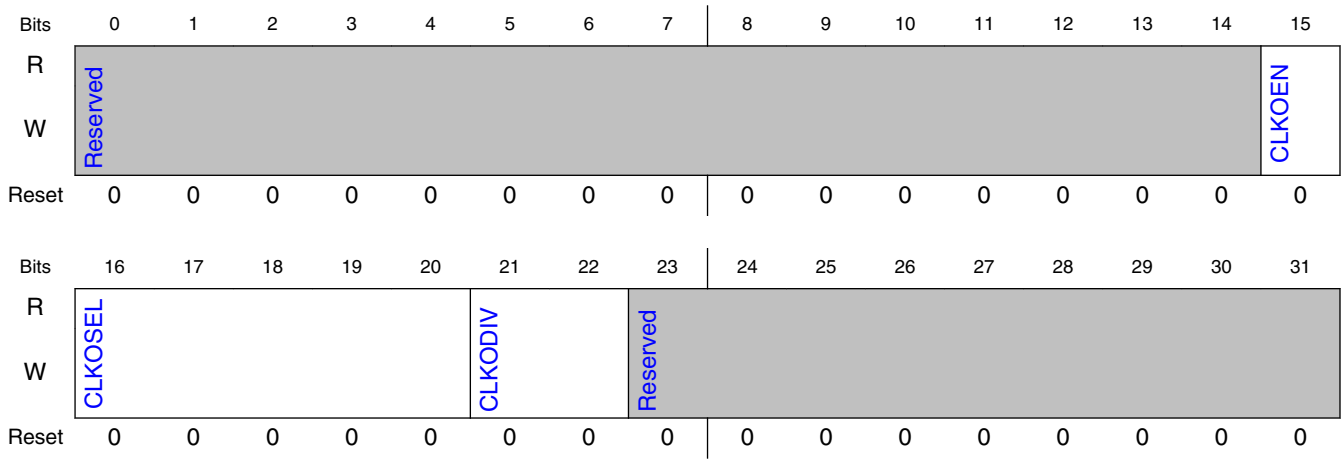
4.3.4.1 Offset

Register	Offset
CLKPCSR	A00h

4.3.4.2 Function

The CLKPCSR register selects which signal to observe on the CLK_OUT pad.

4.3.4.3 Diagram



4.3.4.4 Fields

Field	Function
0-14 —	- Reserved
15 CLKOEN	CLKOEN CLK_OUT pad enable 0b - Release CLK_OUT pad to high impedance 1b - Enable CLK_OUT pad
16-20 CLKOSEL	CLKOSEL Selects core related clock signal for observation on CLK_OUT pad Settings not shown below are reserved. 11111b - 125 MHz SYSCLK

Table continues on the next page...

Field	Function
21-22 CLKODIV	CLKODIV Selects division setting for clock-out mux output to reduce the frequency of the signal to a more observable/measurable range. 00b - Divide-by-1 01b - Divide-by-2 10b - Divide-by-4 11b - Divide-by-8
23-31 —	- Reserved

4.3.5 Platform PLL general status register (PLLPGSR)

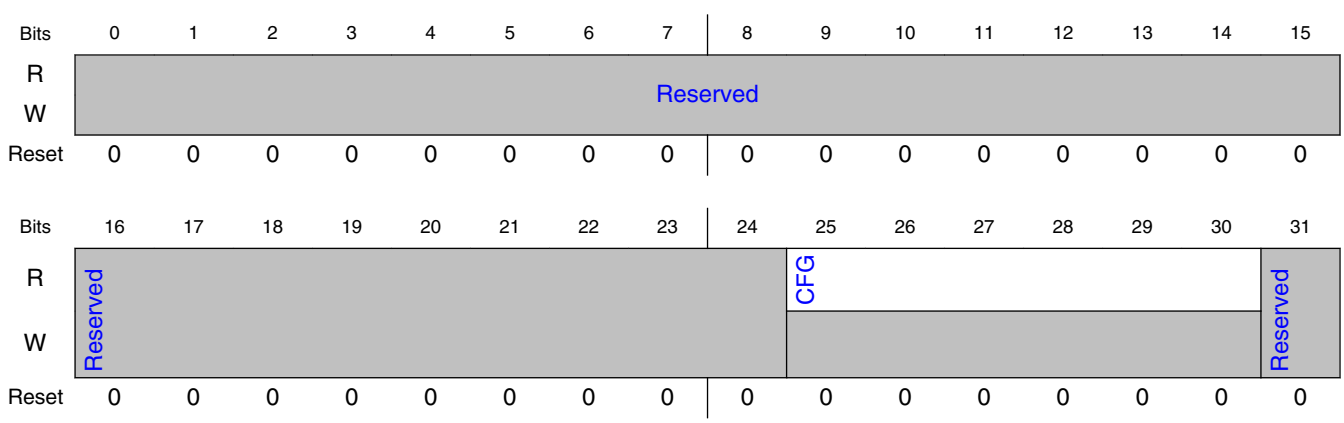
4.3.5.1 Offset

Register	Offset
PLLPGSR	C00h

4.3.5.2 Function

The PLLPGSR register provides information regarding the PLL's configuration.

4.3.5.3 Diagram



4.3.5.4 Fields

Field	Function
0-24 —	- Reserved
25-30 CFG	CFG Reflects the current PLL multiplier configuration. Indicates the frequency for this PLL. Reflects the values programmed in RCW[SYS_PLL_RAT]. Defined settings are from 0_0111 (7:1) through 1_0000 (16:1). All other encodings are reserved. **NOTE:** Not all ratios are supported due to frequency restrictions. Refer to the chip data sheet for the supported frequencies.
31 —	- Reserved

4.4 Functional description

This section describes the various ways to reset the chip, the POR sequence, the POR configurations, the reset configuration word (RCW), and the clocking on the device.

4.4.1 Power-on reset sequence

Assertion of the external PORESET_B signal initiates the power-on reset flow. See the chip data sheet for more information regarding power sequencing and PORESET_B input requirements.

After the negation of PORESET_B, the reset control logic begins cycling the device through its full reset and RCW configuration process. Initially, the RCW source POR configuration inputs are sampled to determine the configuration source. Next, the device begins loading the RCW data. The system PLL begins to lock according to the clock ratio/mode values communicated in the RCW data. Once the PLL locks and the RCW data is loaded, the clocking unit begins distributing PLL outputs throughout the device. Pre-boot initialization is then optionally performed, and the core is permitted to boot.

The detailed POR sequence for the device is as follows:

1. A 25 MHz crystal is used with an on-chip crystal oscillator through pins (EXTAL/XTAL) or an external clock oscillator/generator generates a 25 MHz square wave to EXTAL. See Clocking mode for details.

2. The external system logic asserts PORESET_B and power is applied to comply with the chip's data sheet. The system applies the stable POR configuration inputs. The cfg_eng_use, cfg_eng_use2, cfg_func_backup, and cfg_sysclk_sel signals are sampled at power cycle.
3. PORESET_B asserted causes all registers to be initialized to their default states and most I/O drivers to be released to high impedance (some clock, clock enables, and system control signals are active).

NOTE

The common on-chip processor (COP) requires the ability to independently assert PORESET_B and TRST_B to fully control the processor. If a JTAG/COP port is used, follow the JTAG/COP interface connection recommendations given in the chip's data sheet. If the JTAG interface and COP header are not being used, NXP recommends that TRST_B be tied to PORESET_B so that TRST_B is asserted when PORESET_B is asserted, ensuring that the JTAG scan chain is initialized during the power-on reset flow. See the JTAG configuration signals section in the chip's data sheet for more information.

4. External system logic negates PORESET_B after its required hold time and after POR configuration inputs have been valid for their required setup times.
5. The device samples the RCW source POR configuration inputs (cfg_rcw_src) on deassertion of PORESET_B to determine the RCW source.
6. Some of the I/O drivers are enabled; specifically, any signals required by the interface specified as the source of RCW data in cfg_rcw_src. All of the DDR I/Os become enabled at this point (though MCKE, MCK, MODT are enabled from the beginning).
7. The pre-boot loader (PBL) starts loading the RCW data from the interface specified by cfg_rcw_src configuration inputs and stores that 64 bytes of data to the RCWSR registers within the device configuration block. Note that if a hard-coded RCW option is used, this PBL RCW loading process is effectively bypassed and the device is automatically configured according to the specific RCW field encodings pre-assigned for the given hard-coded RCW option (see RCW Field Definitions for more information).
8. The PLLs begin to lock.
9. The sequence then waits for the PBL RCW process to be completed (loading of all 512 bits). If the PBL reports an error during its process of loading the RCW data, the device reset sequence is halted indefinitely, waiting for another PORESET_B.
10. The platform clock tree is then switched over and is driven by the output of the platform PLL.

11. All other I/O drivers are enabled at this point.
12. The PBL performs pre-boot initialization, if enabled by RCW, by reading data from the QuadSPI interface and writing to CCSR space or local memory space (OCRAM1 or OCRAM2, DDR). If the PBL reports an error during its pre-boot initialization process, the device reset sequence is halted indefinitely, waiting for the PORESET_B. In sequence to the PBI, the SCRATCHRW2 register must also be programmed to specify boot vector location.
13. System ready state. The peripheral interfaces are released to accept external requests and boot vector fetches by the core are allowed to proceed unless processor booting is further held off by the boot release register (BRR) in the device configuration module. The ASLEEP signal, if selected through RCW, negates, indicating the ready state of the system. After reaching this system ready state, the ASLEEP signal transitions to the asserted state when the device enters sleep mode.

NOTE

After completing reset, software should check the SerDes_PLLnRSTCTL[RST_DONE] field to make sure that each active SerDes PLL on the device has locked. Transactions or packet data cannot be transferred through the targeted lane(s) of the SerDes interface if the PLL associated with the lane(s) does not lock properly. Note that a SerDes PLL will not lock if the corresponding reference clock is not provided. Failure to lock SerDes PLLs results in RESET_REQ_B assertion, whether the system continues or halts is up to the board implementation as in RESET_REQ_B behavior

Figure below shows a timing diagram of the POR sequence.

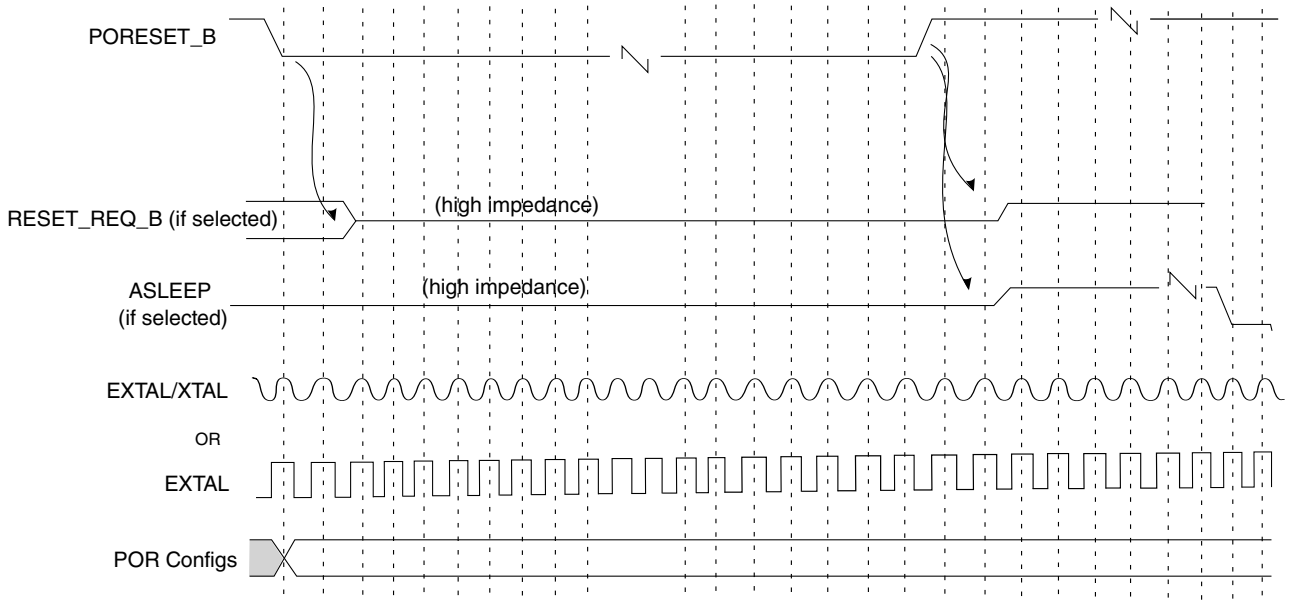


Figure 4-1. Power-on reset sequence

NOTE

Refer to the chip data sheet for the exact timings of PORESET assertion, hold, and setup.

4.4.2 Core Soft Reset

The following steps need to be performed by the software for the core soft reset:

1. Write '1' to the SCFG_CORESRENCR[CORESREN] bit in the SCFG chapter, to enable the soft reset to the corresponding core.
2. Write '1' to the SCFG_CORE0_SFT_RST[SOFT_RESET] bit in the SCFG chapter to enable the soft reset to the core 0. Writing to this bit triggers the corresponding interrupt to the core (interrupt ID 196). The interrupt needs to be configured as edge trigger interrupt.
3. Perform the GIC interrupt mapping:
   - Enable the interrupt.
   - Select edge trigger mode.
   - Route 196 to core 0.
4. In the core, corresponding ISR will be programmed with WFI
5. Core executes WFI instructions after receiving the interrupt.
6. Once the core executes WFI instruction, COP generates the corresponding core soft reset.

4.4.3 Reset state timing

The following table provides the state timing for diffrent RCW/PBI source.

Table 4-7. Reset state timing

RCW/PBI source	RCW loading time (no. of SYSCLK cycles)	RCW source interface frequency (RCW loading phase in terms of SYSCLK ratio)	System PLL locking time (no. of SYSCLK cycles)	RCW source interface frequency (PBI phase in terms of platform clock ratio)	PBI Time(no. of platform clock cycles)	No. of transactions in PBI
QuadSPI	63757	1.953 MHz	68616	15.6 MHz	7351	3 writes of 4 bytes each

The figure below shows the reset timeline diagram in accordance with the power-on reset sequence.

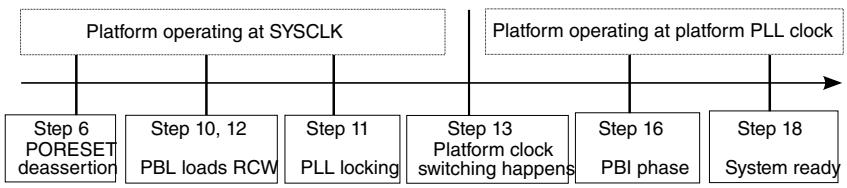


Figure 4-2. Reset Timeline Diagram

4.4.4 Power-on reset configuration

Various device functions are initialized by sampling certain signals during power on reset.

During PORESET_B, all other signal drivers connected to these signals must be in the high-impedance state.

All POR configuration signals have internal pull-up resistors so that if the desired setting is high, there is no need for a pull-up resistor on the board. See the chip data sheet for proper pull-down resistor values for POR configuration signals, when necessary.

This section describes the functions and modes configured by the POR configuration signals.

NOTE

In the following tables, the binary value 0 represents a signal pulled down to GND and a value of 1 represents a signal pulled up to that signal's corresponding V_{DD} voltage level, regardless of the sense of the functional signal name.

4.4.4.1 Reset configuration word (RCW) source

The reset configuration word (RCW) source input, cfg_rcw_src, is multiplexed on CLK_OUT. This configuration input selects the source for the RCW data as shown in the following table. The encoded values latched on this signal during POR are accessible in PORSR1[RCW_SRC], as described in the DCFG_PORSR1 register.

Table 4-8. RCW source encodings

cfg_rcw_src value (binary)	RCW source
0	Hard-coded RCW option
1	QuadSPI (QSPI)

4.4.4.2 Transconductance selection control

The cfg_eng_use and cfg_eng_use2 signals are used to provide transconductance values based on the crystal specification. The following table describes the transconductance configuration requirement of the crystal oscillator.

Table 4-9. Transconductance configuration requirement

cfg_eng_use2	cfg_eng_use	Transconductance
0	0	0.21x
0	1	0.55x
1	0	0.66x
1	1	1.00x

NOTE

Please refer device data sheet for input crystal specifications.
For external clock oscillator mode, these pins may be left connected to their default values (0'b11).

4.4.5 Reset configuration word (RCW)

The chip uses an external memory interface to import a subset of the reset configuration information from a memory device during reset.

Such information is called reset configuration word (RCW) data.

The pre-boot loader (PBL) loads RCW data from a non-volatile memory device interface, as specified by the RCW source configuration inputs (cfg_rcw_src-see the "Reset Configuration Word Source" section in this chapter for more information). See the Pre-Boot Loader chapter for details on the operation of the PBL. Note that this approach does not completely remove the necessity for at least a few power-on reset (POR) configuration signals. As noted, POR config signals are used to control RCW source information in addition to other low-level system configuration.

The RCW is 512 bits long in order to contain all necessary configuration information for the chip. RCW data is read from external memory and written to the RCW status registers (RCWSR) contained in the Device Configuration and Pin Control chapter, after which the device is configured as specified in the RCW. The "Required format of data structure consumed by PBL" section in the Pre-Boot Loader chapter provides details of the data structure that is required to reside in non-volatile memory.

NOTE

The chip makes all the required pins available for selected source interface for RCW. Any other multiplexing options on these pins will be overridden.

4.4.5.1 RCW Field Definitions

This table describes the function of the individual bits of the 512-bit (64-byte) RCW data structure.

NOTE

Unless noted otherwise, any bit ranges in the table listed as reserved must be populated with 0.

Table 4-10. RCW Field Descriptions

Bit(s) (of 0-511)	Field Name	Description	Notes/comments
PLL CONFIGURATION (BITS 0-127)			
0-1	SYS_PLL_CFG	System PLL Configuration.	Options: 00 For all valid Platform PLL frequencies 01-11 Reserved
2-6	SYS_PLL_RAT	System PLL Multiplier/Ratio	This field selects the platform clock:SYSCLK ratio. Options: 0_0000 Reserved 0_0100 4:1 (Asynchronous mode) All other encodings are reserved.

Table continues on the next page...

Table 4-10. RCW Field Descriptions (continued)

Bit(s) (of 0-511)	Field Name	Description	Notes/comments
7-23	Reserved. Must be set to all 0's.		
24-25	CGA_PLL1_CFG	Cluster Group A PLL 1 Configuration	Options 00 For all Cluster PLL frequencies 01-11 Reserved
26-31	CGA_PLL1_RAT	Cluster Group A PLL 1 Multiplier/Ratio	Options: 00_0110 6:1 (Asynchronous mode) 00_0111 7:1 (Asynchronous mode) 00_1000 8:1 (Asynchronous mode) 00_1001 9:1 (Asynchronous mode) 00_1010 10:1 (Asynchronous mode) All other encodings are reserved. **NOTE:** Refer Differences between silicon revisions 1.0 and 2.0 for the implementation in silicon revision 1.0.
32-95	Reserved. Must be set to all 0's.		
96-99	C1_PLL_SEL	Cluster 1 PLL Select.	Options: 0000 CGA_PLL1 /1 0010 CGA_PLL1 /2 All other encodings are reserved.
100-127	Reserved. Must be set to all 0's.		
SerDes PLL AND PROTOCOL CONFIGURATION (BITS 128-183)			
128-143	SRDS_PRTCL_S1	SerDes Protocol Select SerDes 1	See Table 28-2 for a complete list of the options and the definitions of this encoded field.
144-159	Reserved. Must be set to all 0's.		
160-161	SRDS_PLL_REF_CLK_SEL_S1	SerDes PLL reference clock select - SerDes 1.	This field selects the PLL reference clock frequency for SerDes 1 See Table 28-3 for different protocols.
162-165	Reserved. Must be set to all 0's.		
166	Reserved. Must be set to 0.		
167	Reserved. Must be set to 0.		
168-169	SRDS_PLL_PD_S1	SerDes PLL power down SerDes 1.	This field is used to power down the SerDes 1 PLLs. Bit 168 corresponds to SerDes 1-PLL1 and bit 169 corresponds to SerDes 1-PLL2. **NOTE:** When PLL1 is not required, it can be powered OFF if the source is internal. If the source is external, the PLL1 cannot be powered OFF. Option : 0 PLL is not powered down 1 PLL is powered down

Table continues on the next page...

Table 4-10. RCW Field Descriptions (continued)

Bit(s) (of 0-511)	Field Name	Description	Notes/comments
			This field is ignored if the respective SRDS_PRTCL_* field does not use a given SerDes PLL except for the case described in Disabling unused SerDes modules.
170-175	Reserved. Must be set to all 0's.		
176-177	Reserved. Must be set to 0x.		
178-183	Reserved. Must be set to all 0's.		
MISC PLL-RELATED (BITS 184-191)			
184-185	Reserved. Must be set to 00.		
186-187	Reserved. Must be set to 00.		
188	Reserved. Must be set to 1.		
189	SerDes_INT_REFCLK	SerDes internal reference clock selection	Options: 0 External differential reference clock (100 MHz/125 MHz) is used 1 Internal differential reference clock (125 MHz) is used **NOTE:** SerDes PLL1 and PLL2 use same clock source chosen by RCW[SerDes_INT_REFCLK]. Input clock to SerDes PLLs should meet all the requirements of all the protocols selected by RCW[SRDS_PRTCL_S1].
190-191	Reserved. Must be set to 00.		
BOOT CONFIGURATION (BITS 192-223)			
192-195	PBI_SRC	Pre-Boot Initialization Source. The pre-boot loader fetches address/data pairs from the selected interface for the purpose of pre-boot initialization of CCSR and/or local memory space.	The following restriction apply: • RCW and pre-boot initialization data must be loaded from the same non-volatile memory device The hard coded RCW source options are not considered their own memory interface for this purpose. Options: 010x QSPI All other encodings are reserved.
196-200	Reserved. Must be set to all 0's.		
201	BOOT_HO	Boot Holdoff	Options: 0 Core 0 not in hold off 1 Core 0 in hold off
202	SB_EN	Secure Boot Enable	**NOTE:** Note that secure boot is enabled if either this RCW bit is set or the Intent to Secure fuse value is set. See the Trust architecture overview section in the Secure Boot and Trust Architecture chapter for more information. Options:

Table continues on the next page...

Table 4-10. RCW Field Descriptions (continued)

Bit(s) (of 0-511)	Field Name	Description	Notes/comments
			0 Secure boot is not enabled 1 Secure boot is enabled
203-215	Reserved. Must be set to all 0's.		
216-223	Reserved. Must be set to all 0's.		
CLOCKING CONFIGURATION (BITS 224-255)			
224-231	Reserved. Must be set to all 0's.		
232	Reserved. Must be set to 0.		
233	Reserved. Must be set to 0.		
234	Reserved. Must be set to 0.		
235	Reserved. Must be set to 0.		
236-241	Reserved. Must be set to all 0's.		
242-243	SYS_PLL_SPD	System PLL Speed Select	Must be set to 01.
244	CGA_PLL1_SPD	Cluster Group A PLL1 Speed Select	This bit must be set based on the cluster group A PLL1 speed. 0 High-speed operation (>= 1 GHz) 1 Low-speed operation (< 1 GHz) **NOTE:** Refer Differences between silicon revisions 1.0 and 2.0 for the implementation in silicon revision 1.0.
245-255	Reserved. Must be set to all 0's.		
MEMORY AND HIGH SPEED I/O CONFIGURATION (BITS 256-287)			
256-263	Reserved. Must be set to all 0's.		
264	HOST_AGT_PEX	Host/Agent PEX. Configures Host/Agent mode for the PCI Express interface.	Options: 0 Host mode 1 Agent mode
265-287	Reserved. Must be set to all 0's.		
GENERAL PURPOSE INFORMATION (BITS 288-319)			
288-319	GP_INFO	General purpose information. This bit has no effect on functional logic; it may be used by software.	
ENGINEERING USE CONFIGURATION (BITS 320-351)			
320-351	Reserved. Must be set to all 0's.		
GROUP A PIN CONFIGURATION (BITS 352-383)			
352-353	Reserved. Must be set to 00.		
354	SDHC2_EXT_CLK	This field configures the functionality of SDHC2_EXT_CLK pins together with SDHC2_BASE_BASE field.	Options: 0 GPIO1[29] 1 FTM2_CH3 **NOTE:** This field is ignored if SDHC2_BASE_BASE is not equal to 2'b00
355	SDHC2_EXT_CMD	This field configures the functionality of	Options: 0 GPIO1[24]

Table continues on the next page...

Table 4-10. RCW Field Descriptions (continued)

Bit(s) (of 0-511)	Field Name	Description	Notes/comments
		SDHC2_EXT_CMD pins together with SDHC2_BASE_BASE field.	1 FTM1_CH1 **NOTE:** This field is ignored if SDHC2_BASE_BASE is not equal to 2'b00
356	SDHC2_EXT_DAT3	This field configures the functionality of SDHC2_EXT_DAT3 pins together with SDHC2_BASE_DAT321 field.	Options: 0 GPIO1[28] 1 FTM2_CH1 **NOTE:** This field is ignored if SDHC2_BASE_DAT321 is not equal to 2'b00
357	SDHC2_EXT_DAT2	This field configures the functionality of SDHC2_EXT_DAT2 pins together with SDHC2_BASE_DAT321 field.	Options: 0 GPIO1[27] 1 FTM1_CH2 **NOTE:** This field is ignored if SDHC2_BASE_DAT321 is not equal to 2'b00
358	SDHC2_EXT_DAT1	This field configures the functionality of SDHC2_EXT_DAT1 pins together with SDHC2_BASE_DAT321 field.	Options: 0 GPIO1[26] 1 FTM2_CH2 **NOTE:** This field is ignored if SDHC2_BASE_DAT321 is not equal to 2'b00
359	SDHC2_EXT_DAT0	This field configures the functionality of SDHC2_EXT_DAT0 pins together with SDHC2_BASE_BASE field.	Options: 0 GPIO1[25] 1 FTM1_CH3 **NOTE:** This field is ignored if SDHC2_BASE_BASE is not equal to 2'b00
360-361	EC1_EXT_SAI3	This field configures the extended functionality of EC1 pins together with EC1_BASE field.	Options: 00 GPIO2[3], GPIO2[11], GPIO2[12] 01 SAI3_TX_DATA, SAI3_TX_SYNC, SAI3_TX_BCLK 10 SAI3_RX_DATA, SAI3_RX_SYNC, SAI3_RX_BCLK 11 Reserved **NOTE:** This field is ignored if EC1_BASE is not equal to 2'b00
362-363	EC1_EXT_SAI4	This field configures the extended functionality of EC1 pins together with EC1_BASE field.	Options: 00 GPIO2[2], GPIO2[13], GPIO2[14] 01 SAI4_TX_DATA, SAI4_TX_SYNC, SAI4_TX_BCLK 10 SAI4_RX_DATA, SAI4_RX_SYNC, SAI4_RX_BCLK 11 Reserved **NOTE:** This field is ignored if EC1_BASE is not equal to 2'b00

Table continues on the next page...

Table 4-10. RCW Field Descriptions (continued)

Bit(s) (of 0-511)	Field Name	Description	Notes/comments
364	EC1_EXT_SAI2_TX	This field configures the extended functionality of EC1 pins together with EC1_BASE field.	Options: 0 GPIO2[4], GPIO2[6], GPIO2[7] 1 SAI2_TX_DATA, SAI2_TX_SYNC, SAI2_TX_BCLK **NOTE:** This field is ignored if EC1_BASE is not equal to 2'b00
365	EC1_EXT_SAI2_RX	This field configures the extended functionality of EC1 pins together with EC1_BASE field.	Options: 0 GPIO2[5], GPIO2[9], GPIO2[10] 1 SAI2_RX_DATA, SAI2_RX_SYNC, SAI2_RX_BCLK **NOTE:** This field is ignored if EC1_BASE is not equal to 2'b00
366-367	EC1_BASE	This field configures the functionality of EC1 pins.	Options: 00 See EC1_EXT_* bit definition 01 EC1 RGMII interface 10 USB 2.0 ULPI interface 11 Reserved
368-369	UART1_BASE	This field configures the functionality of UART1 pins.	Options: 00 GPIO1[0], GPIO1[1] 01 UART1_SOUT, UART1_SIN 10 PE_UART_TXD, UART_PE_RXD 11 Reserved
370-371	UART2_BASE_FLOW	This field configures the functionality of JTAG/UART2 pins together with the TJTAG_EN pins.	Options: 00 GPIO1[7], GPIO1[9], GPIO1[10] 01 UART2_RTS_B, UART2_CTS_B, GPIO1[10] 10 SAI5_TX_DATA, SAI5_TX_SYNC, SAI5_TX_BCLK 11 SAI5_RX_DATA, SAI5_RX_SYNC, SAI5_RX_BCLK **NOTE:** This bit is ignored if TJTAG_EN=1, in which case these pins are used as JTAG rather than as defined in bit settings.
372-373	SDHC1_BASE	This field configures the functionality of the eSDHC1 pins.	Options: 00 GPIO1[15:20] 01 SDHC1_CMD, SDHC1_DAT[0:3], SDHC1_CLK 10 SDHC1_CMD, SDHC1_DAT0, GPIO1[17], GPIO1[18], GPIO1[19], SDHC1_CLK 11 Reserved
374-375	SDHC2_BASE_DAT321	This field configures the functionality of the eSDHC1 DATA[3:1] pins together with the SDHC2_EXT_* field.	Options: 00 See SDHC2_EXT_* bit definition 01 SDHC2_DAT[3:1]. This option is only valid if SDHC2_BASE_BASE is configured for SDHC2 operation.

Table continues on the next page...

Table 4-10. RCW Field Descriptions (continued)

Bit(s) (of 0-511)	Field Name	Description	Notes/comments
			10 SAI1_RX_BCLK, SAI1_TX_BCLK, SAI1_TX_SYNC 11 SPI_MISO, SPICS2_B, SPI_CS1_B
376-377	SDHC2_BASE_BASE	This field configures the functionality of the eSDHC1 single bit mode pins together with the SDHC2_EXT_* field.	Options: 00 See SDHC2_EXT_* bit definition 01 SDHC2_CMD, SDHC2_DAT0, SDHC2_CLK 10 SAI1_RX_DATA, SAI1_RX_SYNC, SAI1_TX_DATA 11 SPI_MOSI, SPICS0_B, SPI_CLK **NOTE:** For 1-bit data (SDHC2_DAT0), this bit must be set to 01. For 4-bit data (SDHC2_DAT0/1/2/3), both the SDHC2_BASE_DAT321 and SDHC2_BASE_BASE bits must be set to 01.
378	UART2_BASE_DATA	This field configures the functionality of the JTAG/ UART2 pins together with the TJTAG_EN pin.	Options: 0 GPIO1[8], GPIO1[6] 1 UART2_SOUT, UART2_SIN **NOTE:** This bit is ignored if TJTAG_EN=1, in which case these pins are used as JTAG rather than as defined in bit settings.
379	EMI1_BASE	This field configures the functionality of EMI1 pins.	Options: 0 GPIO2[15], GPIO2[16] 1 EMI1_MDC, EMI1_MDIO **NOTE:** Refer SCFG_MDIOSELCR for EMI interface multiplexing and MDIO source/destination configuration.
380-381	GPIO_FTM_EXTCLK_ BASE	This field configures the functionality of TBSCAN_EN_B pin together with the TJTAG_EN pin.	Options: 00 GPIO1[30] 01 Reserved 10 FTM_EXTCLK 11 Reserved **NOTE:** This bit is ignored if TJTAG_EN=1, in which case this pin is used as TBSCAN_EN_B rather than as defined in bit setting.
382-383	CLK_OUT_BASE	This field configures the functionality of the CLK_OUT pin.	Options: 00 GPIO1[31] 01 CLK_OUT 10 RESET_REQ_B 11 Reserved **NOTE:** • RESET_REQ_B is selected through either RCW[382:383] or RCW[QSPI_IIC2]. Selection of

Table continues on the next page...

Table 4-10. RCW Field Descriptions (continued)

Bit(s) (of 0-511)	Field Name	Description	Notes/comments
			RESET_REQ_B in both the RCW fields simultaneously is prohibited. • Refer Differences between silicon revisions 1.0 and 2.0 for the implementation in silicon revision 1.0.
GROUP B PIN CONFIGURATION (BITS 384-415)			
384-415	Reserved. Must be set to all 0's.		
SoC-SPECIFIC CONFIGURATION (BITS 416-447)			
416-418	Reserved. Must be set to 00.		
419	SDHC1_CD	This field selects the functionality assigned to the SDHC1_CD_B pin.	Options: 0 GPIO1[21] 1 SDHC1_CD_B
420	SDHC1_WP	This field selects the functionality assigned to the SDHC1_WP pin.	Options: 0 GPIO1[22] 1 SDHC1_WP
421	QSPI_DATA0_GPIO	This field selects the functionality of the QSPI pins for 2-bit interface.	Options: 0 QSPI_A_DATA0, QSPI_A_SCK, QSPI_A_CS0 1 GPIO1[11], GPIO1[4], GPIO1[5]
422-423	QSPI_DATA1_GPIO	This field selects the functionality of the QSPI pins for 2-bit interface.	Options: 00 QSPI_A_DATA1 01 GPIO1[12] 10 Reserved 11 Reserved
424-425	QSPI_IIC2	This field selects the functionality of the QuadSPI pins for additional data bits for the 4-bit interface.	Options: 00 GPIO1[13], GPIO1[14] 01 IIC2_SCL, IIC2_SDA 10 QSPI_A_DATA2, QSPI_A_DATA3 11 GPIO1[13], RESET_REQ_B **NOTE:** • The QuadSPI multiplexing can be changed by software through SCFG_PMUXCR0 register. • RESET_REQ_B is selected through either RCW[382:383] or RCW[QSPI_IIC2]. Selection of RESET_REQ_B in both the RCW fields simultaneously is prohibited. • Refer Differences between silicon revisions 1.0 and 2.0 for the implementation in silicon revision 1.0.
426-428	Reserved. Must be set to all 0's.		
429-430	USB1_DRVVBUS_BASE	This field configures the functionality of the USB1_DRVVBUS pin.	Options: 00 GPIO2[0]

Table continues on the next page...

Table 4-10. RCW Field Descriptions (continued)

Bit(s) (of 0-511)	Field Name	Description	Notes/comments
			01 USB1_DRVVBUS 10 Reserved 11 Reserved
431-432	USB1_PWRFAULT_BASE	This field configures the functionality of the USB1_PWRFAULT pin.	Options: 00 GPIO2[1] 01 USB1_PWRFAULT 10 ASLEEP 11 Reserved
433	Reserved. Must be set to 0.		
434	SDHC1_VSEL	This field configures voltage of the EVDD IO domain.	Options: 0 GPIO1[23] 1 SDHC1_VSEL
435-437	Reserved. Must be set to all 0's.		
438	EMI1_DMODE	This field selects the EMI1-MDIO pin multiplexing related operating modes.	MDIO open-drain mode: Recommended setting for this bit is 1'b1. Options: 0 MDIO configured in normal functional mode 1 MDIO configured in open-drain mode.
439-440	EVDD_VSEL	This field configures voltage of the EVDD IO domain	Options: 00 1.8V 01 Reserved 10 3.3V 11 Reserved
441-442	IIC1_BASE	This field configures functionality of the IIC1 pins	Options: 00 GPIO1[2:3] 01 IIC1_SCL, IIC1_SDA 10 FTM1_CH0, FTM2_CH0 11 Reserved
443	Reserved. Must be set to 0.		
444	EMI1_CMODE	This field selects the EMI1-MDC pin multiplexing related operating modes.	MDC open-drain mode: Recommended setting for this bit is 1'b0. Options: 0 MDC configured in normal functional mode 1 MDC configured in open-drain mode
445-447	Reserved. Must be set to 000.		
PLL AND CLOCKING CONFIGURATION EXPANSION (BITS 448-511)			
448-471	Reserved. Must be set to all 0's.		

Table continues on the next page...

Table 4-10. RCW Field Descriptions (continued)

Bit(s) (of 0-511)	Field Name	Description	Notes/comments
472-481	SYSCLK_FREQ	SYSCLK frequency	This field is used for proper hardware configuration of the Arm generic timer and to determine the SYSCLK frequency by software. The value in this field is multiplied by 166.667 KHz. It is used to provide information to determine the frequency of operation of the Arm generic timer. This bit must be set to 0x258. 0x258 - 100 MHz
482-493	Reserved. Must be set to all 0's.		
494-511	Reserved. Must be set to all 0's.		

4.4.5.2 Hard-coded RCW options

If the hardcoded RCW option is used, the PBL RCW loading process is bypassed and the device is automatically configured according to the specific RCW field encodings that are pre-assigned for the hardcoded RCW option. The hardcoded RCW option can be useful if the board has no valid RCW present or if the customer is unable to load in the RCW from a non-volatile memory. Using a hardcoded RCW option allows the part to be initialized so that the desired RCW can be programmed or restored.

NOTE

- Hard-coded RCW option is not recommended as a feature to be used for functional bring-up (DDR and SerDes functionality). It can be used to bring-up the board with blank flash where flash needs to be programmed for RCW information through JTAG interface.
- If a hardcoded RCW option is used, the user should not enable the core and the DDR controllers. The user may enable the core and DDR controllers after a valid RCW is restored.

4.4.5.3 RCW settings for hard-coded options

Table 4-11. RCW settings for hard-coded RCW options

RCW field	cfg_rcw_src
PLL CONFIGURATION (BITS 0-127)	
SYS_PLL_CFG	2'b00

Table continues on the next page...

Table 4-11. RCW settings for hard-coded RCW options (continued)

RCW field	cfg_rcw_src
SYS_PLL_RAT (must be an even ratio)	4:1
CGA_PLL1_CFG	2'b00
CGA_PLL1_RAT	6:1
C1_PLL_SEL	CGA_PLL1 / 1
SERDES PLL AND PROTOCOL CONFIGURATION (BITS 128-183)	
SRDS_PRTCL_S1	0x000 (all lanes unused)
SRDS_PLL_REF_CLK_SEL_S1	2'b11
SRDS_PLL_PD_S1	2'b00 (PLL1 = not powered down, PLL2 = not powered down
MISC PLL-RELATED (BITS 184-191)	
SerDes_INT_REFCLK	1'b1
BOOT CONFIGURATION (BITS 192-223)	
PBI_SRC	Disabled
BOOT_HO	1'b0
SB_EN	Secure boot disabled
CLOCKING CONFIGURATION (BITS 224-255)	
SYS_PLL_SPD	2'b01
CGA_PLL1_SPD	1'b1
MEMORY AND HIGH SPEED I/O CONFIGURATION (BITS 256-287)	
HOST_AGT_PEX	1'b1 (Agent mode)
GENERAL PURPOSE INFORMATION (BITS 288-319)	
GP_INFO	All 0's
CHASSIS GROUP A PIN CONFIGURATION (BITS 352-383)	
SDHC2_EXT_CLK	1'b0
SDHC2_EXT_CMD	1'b0
SDHC2_EXT_DAT3	1'b0
SDHC2_EXT_DAT2	1'b0
SDHC2_EXT_DAT1	1'b0
SDHC2_EXT_DAT0	1'b0
EC1_EXT_SAI3	2'b00
EC1_EXT_SAI4	2'b00
EC1_EXT_SAI2_TX	1'b0
EC1_EXT_SAI2_RX	1'b0
EC1_BASE	2'b00 (GPIO)
UART1_BASE	2'b00 (GPIO)
UART2_BASE_FLOW	2'b00 (GPIO)
SDHC1_BASE	2'b00 (GPIO)
SDHC2_BASE_DAT321	2'b00 (GPIO)
SDHC2_BASE_BASE	2'b00 (GPIO)

Table continues on the next page...

Table 4-11. RCW settings for hard-coded RCW options (continued)

RCW field	cfg_rcw_src
UART2_BASE_DATA	1'b0 (GPIO)
EMI1_BASE	1'b0 (GPIO)
GPIO_FTM_EXTCLK_BASE	2'b00 (GPIO)
CLK_OUT_BASE	1'b0 (GPIO)
SoC SPECIFIC CONFIGURATION (BITS 416-447)	
SDHC1_CD	1'b0 (GPIO)
SDHC1_WP	1'b0 (GPIO)
QSPI_DATA0_GPIO	1'b0 (QSPI_DATA0)
QSPI_DATA1_GPIO	2'b00 (QSPI_DATA1)
QSPI_IIC2	2'b00 (GPIO)[1]
USB1_DRVVBUS_BASE	1'b0 (GPIO)
USB1_PWRFAULT_BASE	1'b0 (GPIO)
SDHC1_VSEL	1'b0 (GPIO)
EMI1_DMODE	1'b0
EVDD_VSEL	2'b11 (Auto)
IIC1_BASE	2'b00 (GPIO)
EMI1_CMODE	1'b0
PLL AND CLOCKING CONFIGURATION EXPANSION (BITS 448-511)	
SYSCLK_FREQ	0x258 (100 MHz)

1. Refer Differences between silicon revisions 1.0 and 2.0 for the implementation in silicon revision 1.0.

4.4.6 Clocking

The following sections describe the clocking within the chip.

4.4.6.1 Clocking mode

The chip supports the following clocking modes:

- Crystal (internal oscillator) based clocking
- External clock oscillator/generator based clocking (on-chip oscillator bypassed)

The following table provides the clocking configuration for both these modes:

Table 4-12. POR configurations

cfg_func_backup	cfg_sysclk_sel	Description
0	0	External clock oscillator/generator-based clocking provides 25 MHz square wave input (on EXTAL)

Table continues on the next page...

Table 4-12. POR configurations (continued)

cfg_func_backup	cfg_sysclk_sel	Description
1	1	Crystal-based clocking provides 25 MHz crystal based clocking

The selection of using the crystal or the external clock oscillator/generator is based on the interfaces used. Refer Guidelines for crystal or external clock oscillator/generator usage with different interfaces for details.

4.4.6.1.1 Crystal based (internal oscillator mode) clocking

In this mode, 25 MHz crystal is used with an on-chip crystal oscillator through pins (EXTAL/XTAL). This crystal oscillator provides a 25 MHz clock to the chip PLLs that generates the following clocks:

- 125 MHz single-ended clock to platform PLL (SYSCLK)
- 100 MHz single-ended clock to core PLL and USB PHY
- 125 MHz differential clock to SerDes PLL

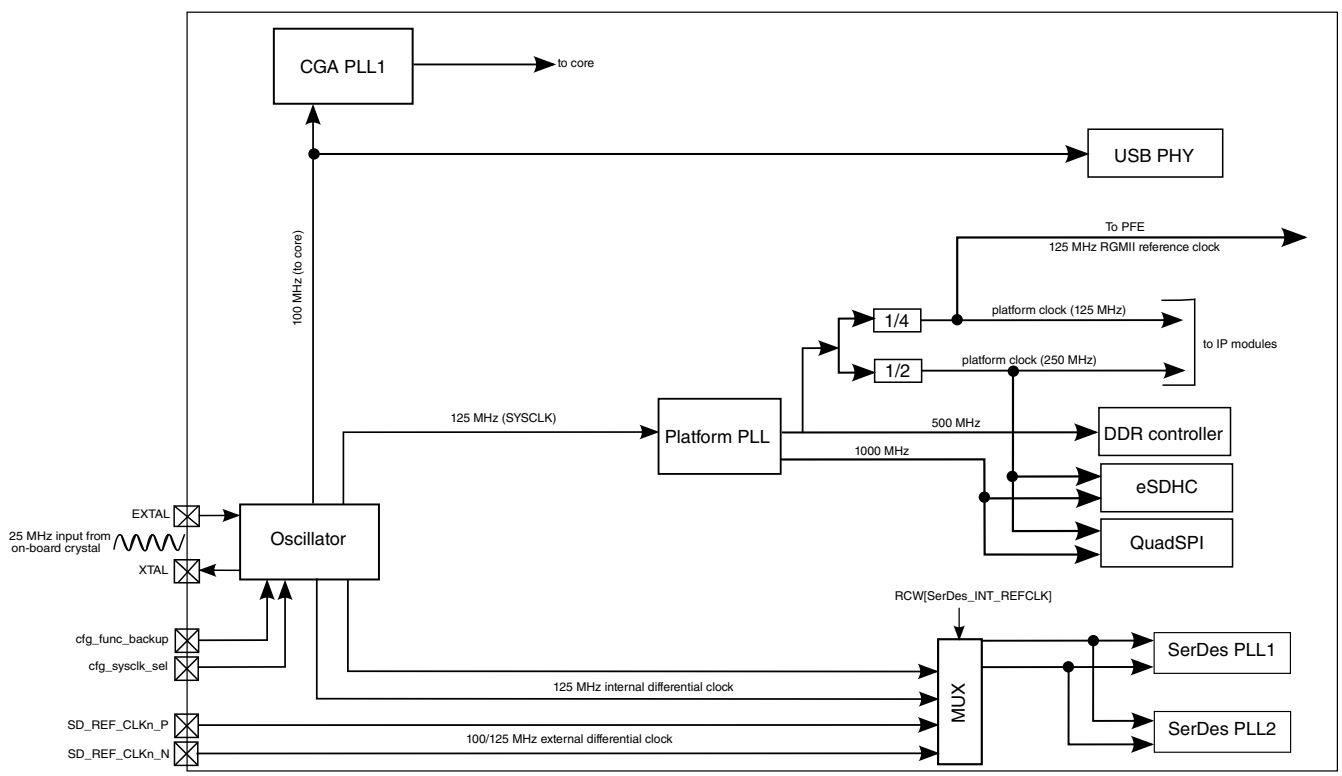


Figure 4-3. Crystal based clocking mode

4.4.6.1.2 External clock oscillator/generator based clocking (on-chip oscillator bypassed)

In this mode, an external clock oscillator/generator is used to generate and provide 25 MHz square wave on EXTAL. In this mode, the XTAL pin is grounded. A PLL takes this 25 MHz clock and generates the following clocks:

- 125 MHz single-ended clock to platform PLL (SYSCLK)
- 100 MHz single-ended clock to core PLL and USB PHY
- 125 MHz differential clock to SerDes PLL

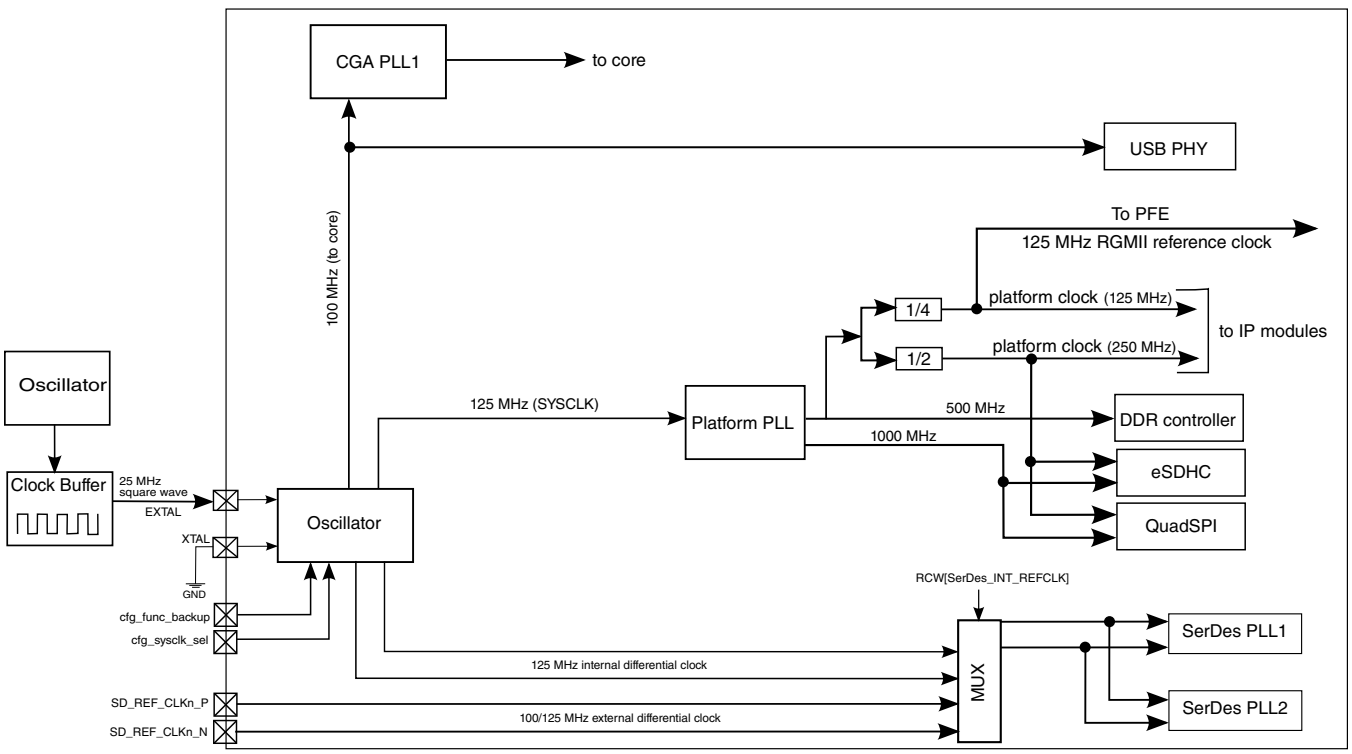


Figure 4-4. External clock oscillator/generator based clocking mode

4.4.6.2 Guidelines for crystal or external clock oscillator/generator usage with different interfaces

Depending on the system requirement, following guidelines must be followed while using the crystal or external clock oscillator/generator with different interfaces:

- Crystal cannot be used as primary clock input for the chip when the following function(s) are enabled:
  - SATA

- PCI Express as root complex and the TX_CLK clock output option is used to provide input clock to the Endpoint
- SGMII

- For all other interfaces, crystal can be used as primary clock input, such as RGMII, USB, PCI Express (without TX_CLK).

NOTE

The crystal and external clock oscillator/generator must follow the specifications as mentioned in the chip data sheet.

4.4.6.2.1 PCI Express use cases (with TX_CLK)

For the systems with PCI Express interface with TX_CLK, an external clock oscillator/ generator is required, as shown in the figure below:

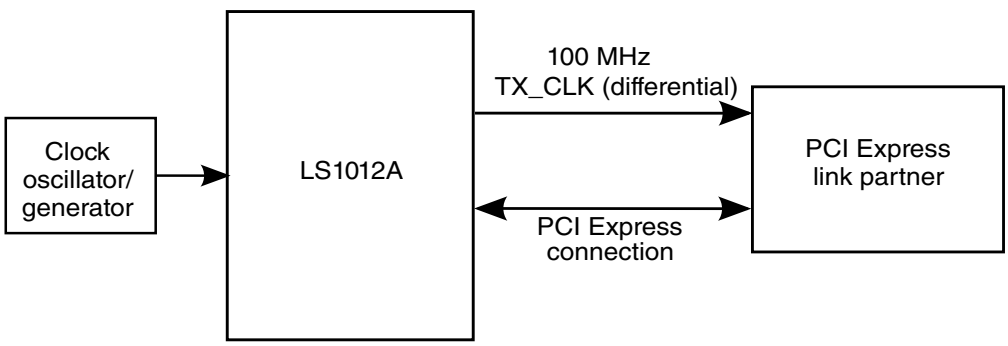


Figure 4-5. PCI Express clocking

TX_CLK can be used when the chip is PCI Express root complex. The chip is capable of supplying the clock input of the PCI Express EP device (directly or through the PCI Express slot connector).

4.4.6.2.2 RGMII-only use cases

For the systems with RGMII PHY, either the crystal or clock oscillator/generator can be used. See the figure below for the RGMII PHY clocking with both crystal as well as clock oscillator/generator modes.

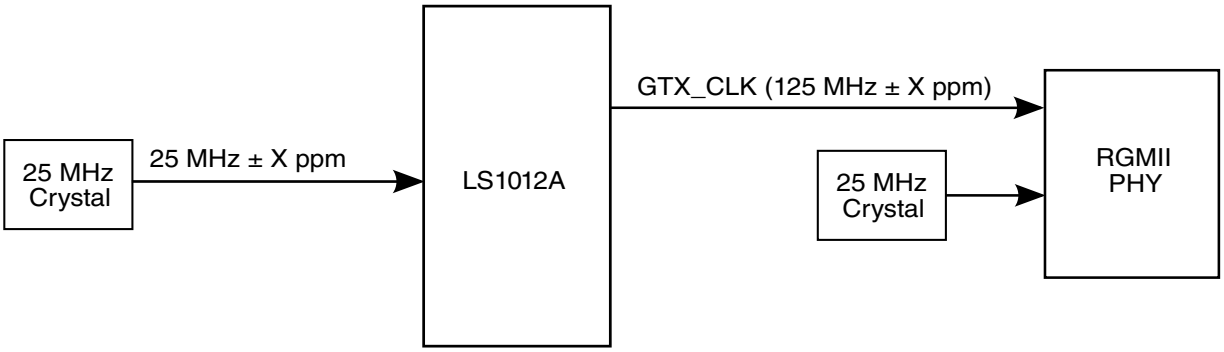


Figure 4-6. RGMII PHY clocking using crystal

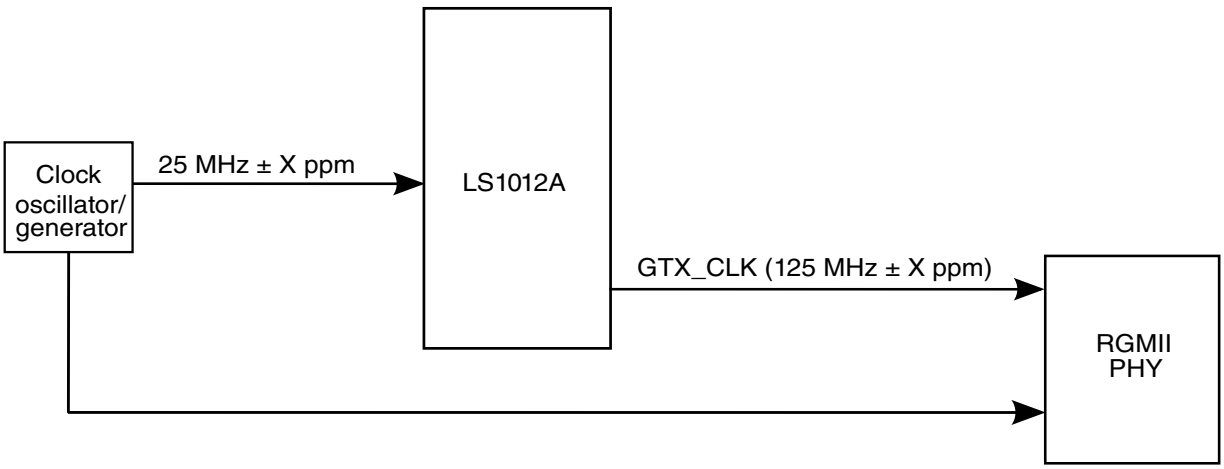


Figure 4-7. RGMII PHY clocking using clock oscillator/generator

NOTE

Crystal or clock oscillator/generator must satisfy the clock tolerance of GTX_CLK required by the RGMII PHY.

4.4.6.2.3 SGMII use cases

For the systems with SGMII PHY, an external clock oscillator/generator is required, as shown in the figure below:

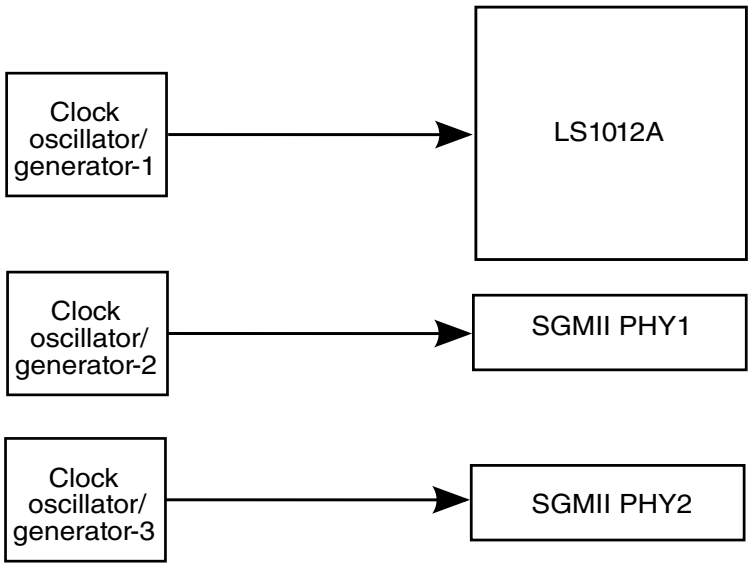


Figure 4-8. MAC to SGMII PHY clocking

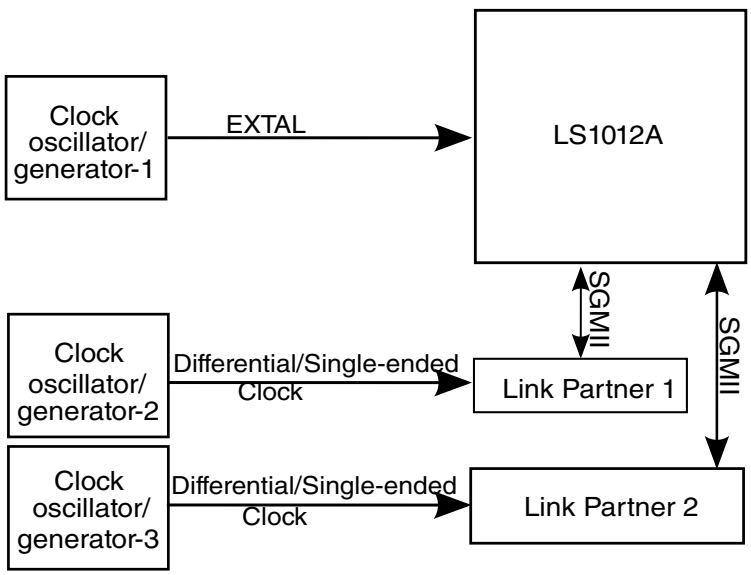


Figure 4-9. MAC to MAC clocking

NOTE

Refer Differences between silicon revisions 1.0 and 2.0 for the SGMII PHY clocking implementation in silicon revision 1.0.

4.4.6.2.4 SATA use cases

For the systems with a SATA interface, an external clock oscillator/generator is required, as shown in the figure below:

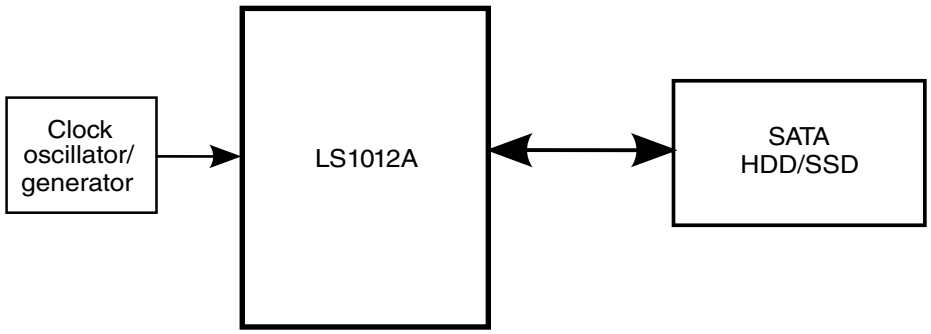


Figure 4-10. SATA clocking

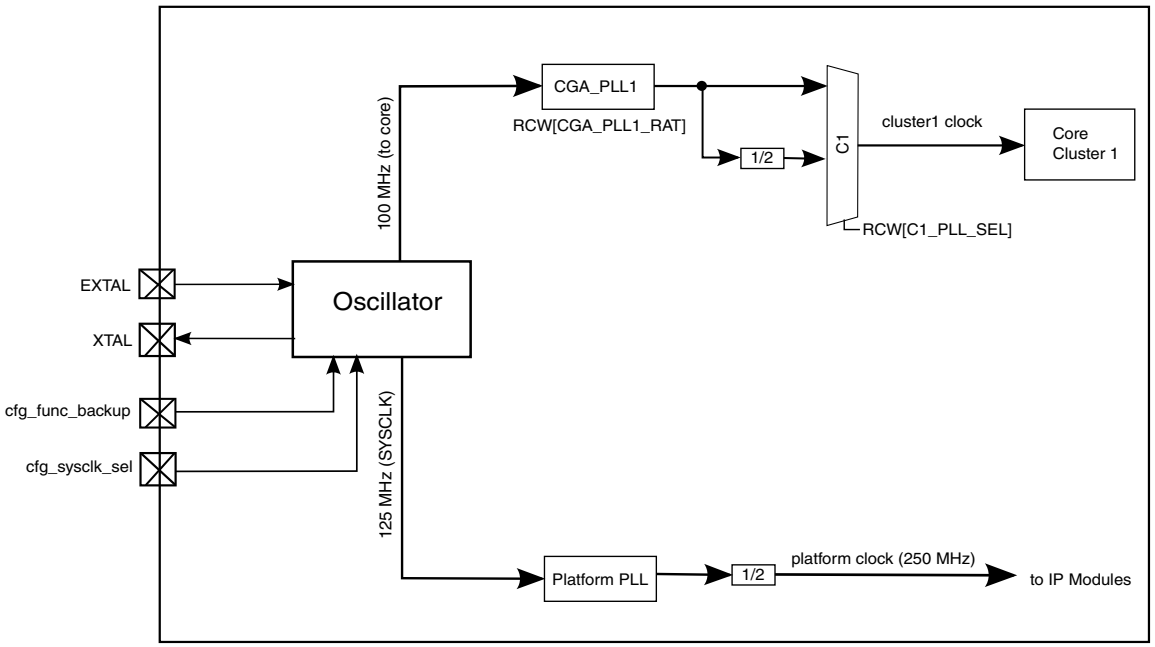
Crystal cannot be used when SATA is enabled on the chip.

4.4.6.3 IP Logic Clock Distribution and Configuration

The chip takes primary clocking input either from the 25 MHz crystal (EXTAL/XTAL) or from the external clock oscillator/generator in the form of 25 MHz square wave. As shown in Figure 4-11, the clocking input generates different frequencies which can then be passed to a variety of internal logic, including cores and peripheral IP modules. The platform PLL is used to provide clocking to the DDR memory controller complexes.

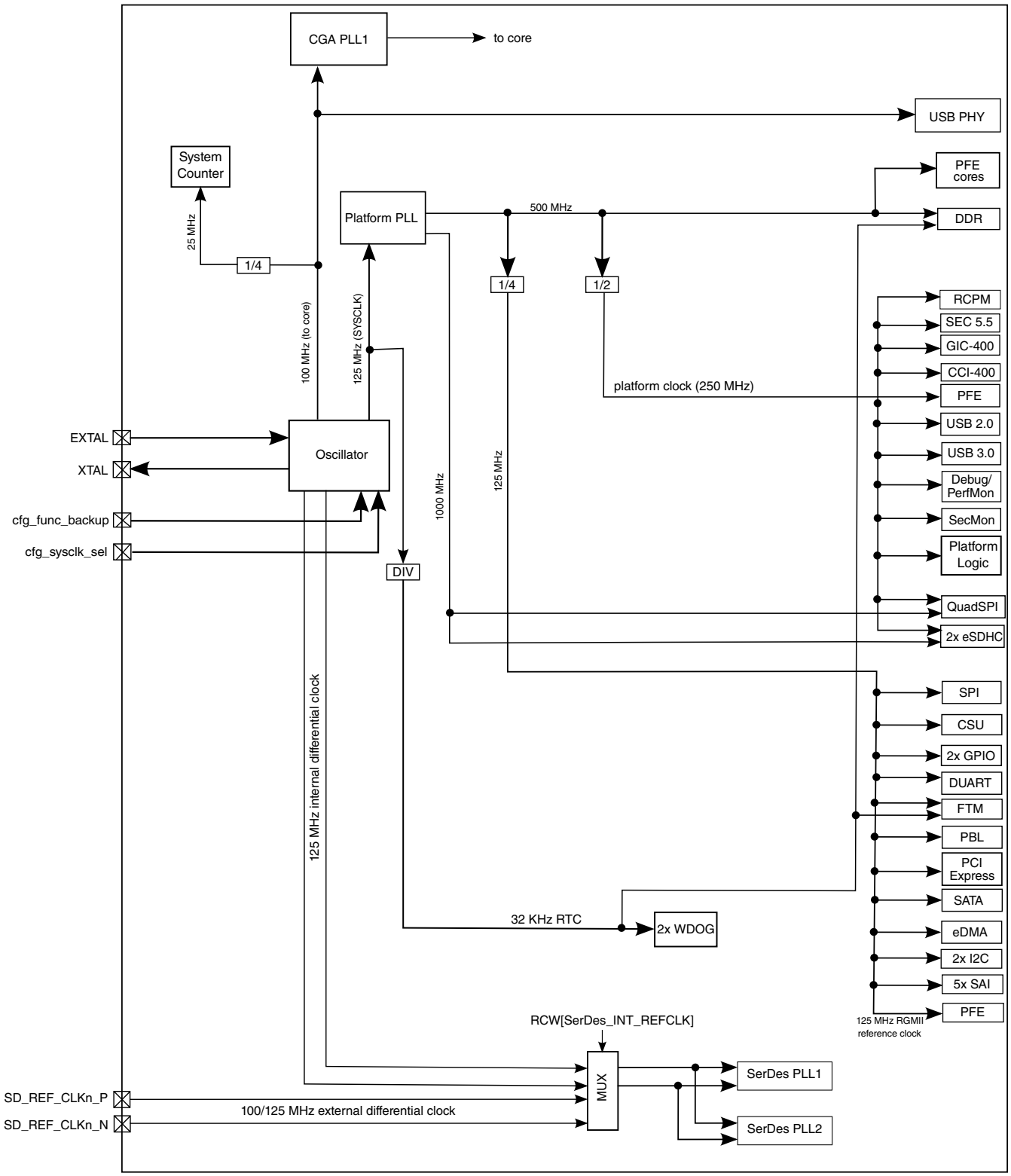
Note that many of the IP modules contain logic allowing software to further modify their external interface clocks within the IP module. See the applicable IP module chapter for details.

The following figures describes internal logic clock distribution along with the means to configure the various ratios and clock sources. Note that the following figures are not intended to reflect external interface clocking. Although sometimes dependent on or derived from logic clocking, a full description of external interface clocking is described within the applicable IP module chapter of this reference manual. Each of the two SerDes PLLs are clocked by the clock that is internally generated by the internal oscillator PLL. However, the interfaces can also be clocked by the external source (SD1_REF_CLK1). (See Valid reference clocks and PLL configurations for SerDes protocols for details regarding valid combinations of external reference clocks and RCW configurations.)



Note: Either a 25 MHz crystal is used with an on-chip crystal oscillator through pins (EXTAL/XTAL) or 25 MHz square wave from external clock oscillator/generator can be used on EXTAL and XTAL should be connected to GND.

Figure 4-11. Clock subsystem block diagram - cluster group A



Note: Either a 25 MHz crystal is used with an on-chip crystal oscillator through pins (EXTAL/XTAL) or 25 MHz square wave from external clock oscillator/generator can be used on EXTAL and XTAL should be connected to GND.

Figure 4-12. Clock subsystem block diagram - IP modules

4.4.6.4 CLK_OUT configuration

The CLK_OUT signal can be configured to offer one of a variety of internal clock signals to external hardware for debug or diagnostic purposes.

(See the clocking register (CLKPCSR) for details.)

Chapter 5
Interrupt Assignments

5.1 Introduction

This chapter describes GIC-400 interrupt assignments for the chip. These are the on-chip interrupt sources from peripheral logic within the integrated device.

5.2 Internal interrupt sources

The implementation of the GIC-400 handles up to 256 interrupt requests to the Arm core. The following table shows the assignments of the internal interrupt sources.

Table 5-1. Interrupt assignments

Internal interrupt number	Interrupt source	Comments
0-31	Arm internal interrupts	These are software generated interrupts for communication among cores and are triggered through GIC-400 registers. See Arm generic interrupt controller (GIC-400) for more information.
32-64	Reserved	
65	Thermal monitor unit alarm	See TMU interrupt enable register (TIER) for more information.
66	Thermal monitor unit critical alarm	
67-85	Reserved	
86	DUART	See Interrupt control logic for more information.
87	Reserved	
88	I^2C 1	All I2C inerrupts are ORed together. See I2C Bus Status Register (I2C_IBSR) for more information.
89	I^2C 2	
90-91	Reserved	
92	USB3	All USB 3.0 interrupts are ORed together and connected to this interrupt.

Table continues on the next page...

Table 5-1. Interrupt assignments (continued)

Internal interrupt number	Interrupt source	Comments
		See the USBSTS register in Table 33-3 for more information.
93	Reserved	
94	eSDHC1 (SD/eMMC)	See Interrupt status register (IRQSTAT) for more information.
95	Reserved	
96	SPI	See Interrupts/DMA requests for more information.
97	eSDHC2 (SD/eMMC)	See Interrupt status register (IRQSTAT) for more information.
98	GPIO1	All GPIO inerrupts are ORed together. See GPIO interrupt event register (GPIER) for more information.
99	GPIO2	
100	Reserved	
101	SATA3	See Chained interrupt separation enabled for more information.
102	EPU	All EPU interrupts are ORed together and connected to this interrupt.
103	SEC job queue 1	This interrupt is reserved non-E personality. Refer security reference manual for E personality.
104	SEC job queue 2	
105	SEC job queue 3	
106	SEC job queue 4	
107	SEC global	
108	Platform control	All Platform Control interrupts are ORed together and connected to this interrupt. See Platform Control for more information.
109	Reserved	
110	Security monitor (secure)	See HPSVSR register in security reference manual for more information.
111	Reserved	
112	CSU	See Central Security Unit for more information.
113-114	Reserved	
115	WDOG 1	See Interrupt event for more information.
116	WDOG 2	
117	Reserved	
118	FTM1	All FTM inerrupts are ORed together. See FTM Interrupts for more information.
119	FTM2	
120-121	Reserved	
122	SCFG_PFE	See PFE PCS status register 1 (PFEPCSSR1), PFE interrupt enable control register (PFEINTENCR1), PFE PCS status register 2 (PFEPCSSR2), PFE interrupt enable control register 2 (PFEINTENCR2),

Table continues on the next page...

Table 5-1. Interrupt assignments (continued)

Internal interrupt number	Interrupt source	Comments
		PFE error control register (PFEERRCR), and PFE error interrupt enable control register (PFEERRINTENCR) for more information.
123-124	Reserved	
125	TZASC	All TZASC interrupts are ORed together and connected to this interrupt. See TrustZone Address Space Controller (TZASC) for more information.
126-130	Reserved	
131	QSPI	All QuadSPI interrupts are ORed together and connected to this interrupt. See Interrupt Signals for more information.
132-134	Reserved	
135	eDMA	The eDMA 32-channel interrupts are ORed together and connected to this interrupt. See Enable Error Interrupt Register (EEI) for more information.
136	A53 core 0 CTI IRQ	Core 0 cross trigger interrupt See Arm® Cortex®-A53 core for more information.
137	Reserved	
138	A53 core 0 PMU IRQ	See Arm® Cortex®-A53 core for more information.
139	Reserved	
140	A53 AXI Error / A53 Int Err	See Arm® Cortex®-A53 core for more information.
141	CCI400 ERRORIRQ / CCI EVNTCNTOVERFLOW	CCI-400 error interrupt CCI-400 transaction bus error due to any transaction error in CCI-400 interconnect and connected to nERRORIRQ of CCI-400. See The Cache coherent interconnect (CCI-400) module as implemented on the chip for more information.
142	PEX1 INTA	See PCI Express Interrupt Pin Register (Interrupt_Pin_Register) for more information.
143	PEX1 INTB	
144	PEX1 INTC	
145	PEX1 INTD	
146-147	Reserved	
148	PEX1 MSI	See PCI Express MSI implementation for more information.
149	PEX1 PME	Refer PM_PME messages in the Power Management chapter of PCI Express™ Base Specification, Revision 3.0 for more information.

Table continues on the next page...

Table 5-1. Interrupt assignments (continued)

Internal interrupt number	Interrupt source	Comments
150	PEX1 CFG err interrupt	See Command Register[SERR] and PCI Express Device Status Register (Device_Status_Register) for more information.
151-170	Reserved	
171	USB2	See Interrupts for more information.
172-179	Reserved	
180	SAI1	All SAI interrupts are ORed together and connected to this interrupt. See SAI Transmit Control Register (TCSR) and SAI Receive Control Register (RCSR) for more information.
181	SAI2	
182	SAI3	
183	SAI4	
184	SAI5	
185-195	Reserved	
196	Soft reset core 0	Edge triggered only This interrupt facilitates core 0 soft reset sequence and should be configured as edge interrupt. See Core Soft Reset for more information.
197-203	Reserved	
204	HIF interrupt	PFE
205	HIF NOCPY interrupt	
206	gpt_tmr_irq, magic packet	
207-223	Reserved	
224	COMMIRQ0	
225-227	Reserved	
228	MBEE	Internal RAM multi-bit ECC error
229	Virtual CPU interface maintenance interrupt core 0	
230-255	Reserved	

NOTE

- There is no interrupt for MDIO. Software needs to use polling mechanism.
- All reserved interrupts should be disabled.

Chapter 6
Arm Modules

6.1 Introduction

The chip implements the following Arm modules:

- Arm® Cortex®-A53 core
- Arm generic interrupt controller (GIC-400)
- Cache coherent interconnect (CCI-400)
- Arm CoreLink™ TrustZone address space controller (TZC-380)

This chapter provides a brief overview of the core and interrupt controller. For more information on these modules, see the Arm documentation that accompanies this reference manual.

Table 6-1. Related resources from Arm

Resource	IP Revision
Arm® Cortex®-A53 MPCore Processor Technical Reference Manual	r0p4
CoreLink™ GIC-400 Generic Interrupt Controller Technical Reference Manual	r0p1

For information on other Arm modules, see the following:

- Cache Coherent Interconnect (CCI-400)
- Arm CoreLink™ TrustZone Address Space Controller (TZC-380)

6.2 Arm® Cortex®-A53 core

The Arm® Cortex®-A53 processor is an extremely power efficient Armv8 processor capable of supporting 32-bit and 64-bit code seamlessly. It makes use of a highly efficient 8-stage in-order pipeline balanced with advanced fetch and data access techniques for performance.

LS1012A features one high-performance Cortex-A53 core:

- 64 and 32-bit execution states for scalable high performance
- Coherent interconnect supporting AMBA® 4 technology
- New instruction set, A64
- In-order pipeline with symmetric dual-issue of most instructions
- 32 KB instruction cache, 32 KB data cache, 256 KB unified L2 cache
- NEON technology - Accelerates multimedia and signal processing algorithms such as video encode/decode, 2D/3D graphics, gaming, audio and speech processing, image processing, telephony, and sound synthesis. Also useful in accelerating floating point code with SIMD execution.
- Hardware-accelerated cryptography - 3x-10x better software encryption performance Useful for small granule decrypt/encrypt too small to efficiently offload to HW accelerator (e.g. https)
- Floating point unit - Hardware support for floating point operations in half-, single- and double-precision floating point arithmetic. IEE754-2008 enhancements are included.
- TrustZone® Technology - Ensures reliable implementation of security applications ranging from digital rights management to electronic payment.
- Double Precision Floating Point SIMD - Allows SIMD vectorisation to be applied to a much wider set of algorithms (e.g. scientific / High Performance Computing (HPC) and supercomputer).
- 64-bit Virtual address reach - Enables virtual memory beyond 4GB 32b limit. Important for modern desktop and server software using memory mapped file I/O, sparse addressing.
- Enhanced Cache management - User space cache operations improve dynamic code generation efficiency, Data Cache Zero for fast clear.
- Power-saving features - Hierarchical clock gating
- Hardware virtualization support

6.3 Arm generic interrupt controller (GIC-400)

Generic interrupt controller (GIC) is a centralized resource for supporting and managing interrupts in a system that includes at least one processor. It provides:

- registers for managing interrupt sources, interrupt behavior, and interrupt routing to one or more processors
- support for the following:
  - the Arm architecture security extensions
  - the Arm architecture virtualization extensions

- enabling, disabling, and generating processor interrupts from hardware (peripheral) interrupt sources
- Software-generated interrupts (SGIs) interrupt masking and prioritization uniprocessor and multiprocessor environments

6.4 On-Chip RAM memory controller (OCRAM)

The on-chip RAM is implemented as a slave device on the 64-bit system AXI bus. There are two OCRAMs in the chip and access can be controlled through CSU peripheral register programming.

Each OCRAM occupies a 64 KB of address region and the start address of each OCRAM (64 KB each) in the memory map is given below:

- OCRAM1: 0x1000_0000
- OCRAM2: 0x1001_0000.

NOTE

The OCRAM1 and OCRAM2 memory content is not initialized to zero by the chip. The software must initialize both OCRAM1 and OCRAM2 in full width (64-bit) access and then perform a write to addresses 0x2014_0534 and 0x2014_0544 with the value 0x0000_0000 to clear the single and multi-bit ECC errors for OCRAM1 and OCRAM2. Also, the ECC error interrupt needs to be cleared in GIC (MBEE). Once completed, the host processor enables interrupt related to ECC errors from OCRAM1 and OCRAM2.

Chapter 7
CSU, OCRAM, and MSCM

7.1 Central Security Unit

The CSU manages the system security policy for accessing device peripherals. The CSU allows trusted code to set individual security access privileges for each of the peripherals, using one of eight security access privilege levels.

CSU features include:

- Configuration of peripheral access permissions for those peripherals unable to control their own access permissions
  - Note that some peripherals/bus slaves have their own access control capabilities (ability to reject transactions from unauthorized masters).
- Optional locking of individual CSU settings until the next POR
- Additional general purpose security related control bits

The CSU controls four parameters, allowing for eight security access privilege levels.

- Secure World vs Non-Secure World access
- Supervisor vs User access
- Read access
- Write access

Supervisor access is superset access to both User and Supervisor. Secure World access is superset access to both Non-Secure and Secure World.

The OCRAM is 64KB of On-Chip SRAM, restricted to Secure World access only by the CSU.

CSU related platform control registers are found in the Miscellaneous System Control Module. The MSCM provides error reporting related to attempted access control violations blocked by the CSU.

7.1.1 CSU Memory Map/Register Definition

The following registers, CSL and SA are collectively referred to as CSU Security Control Registers (SCRs) and can be written only by software executing in the TrustZone Secure Supervisor Mode.

CSU memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
151_0000	Config Security Level (CSU_CSL)	32	R/W	0033_0033h	7.1.1.1/266
151_0218	Secure Access register (CSU_SA0)	32	R/W	0000_0000h	7.1.1.2/271
151_021C	Secure Access register (CSU_SA1)	32	R/W	0000_0000h	7.1.1.2/271

7.1.1.1 Config Security Level (CSU_CSL)

Each Config Security Level Register holds the config security level bits (access permission bits) for two targets. In the below table all peripherals are initially Secure User/Supervisor Read/Write Enabled, with no access by non-secure world masters. The CSL registers allow secure world to make the various peripherals' registers accessible to non-secure world software and bus masters.

Table 7-1. CSL Device Peripherals

Offset	CSL Register[bits]	Reset value (in binary)	Peripheral
0xBASE_0000	CSL0[24:16]	0_0011_0011	Reserved
	CSL0[8:0]	0_0011_0011	PCI express controller 1 IO configuration space and memory space
0xBASE_0004	CSL1[24:16]	0_0011_0011	Reserved
	CSL1[8:0]	0_0011_0011	Reserved
0xBASE_0008	CSL2[24:16]	0_0011_0011	OCRAM1
	CSL2[8:0]	0_0011_0011	Reserved
0xBASE_000C	CSL3[24:16]	0_0011_0011	PCI express controller 1 registers
	CSL3[8:0]	0_0011_0011	OCRAM2
0xBASE_0010	CSL4[24:16]	0_0011_0011	QuadSPI (QSPI) memory space
	CSL4[8:0]	0_0011_0011	Reserved
0xBASE_0014	CSL5[24:16]	0_0011_0011	SATA controller registers
	CSL5[8:0]	0_0011_0011	USB 3.0 controllers registers
0xBASE_0018	CSL6[24:16]	0_0011_0011	Reserved
	CSL6[8:0]	0_0011_0011	Reserved

Table continues on the next page...

Table 7-1. CSL Device Peripherals (continued)

Offset	CSL Register[bits]	Reset value (in binary)	Peripheral
0xBASE_001C	CSL7[24:16]	0_0011_0011	Reserved
	CSL7[8:0]	0_0011_0011	Reserved
0xBASE_0020	CSL8[24:16]	0_0011_0011	Reserved
	CSL8[8:0]	0_0011_0011	Reserved
0xBASE_0024	CSL9[24:16]	0_0011_0011	Reserved
	CSL9[8:0]	0_0011_0011	Reserved
0xBASE_0028	CSL10[24:16]	0_0011_0011	Reserved
	CSL10[8:0]	0_0011_0011	Reserved
0xBASE_002C	CSL11[24:16]	0_0011_0011	Reserved
	CSL11[8:0]	0_0011_0011	PFE registers
0xBASE_0030	CSL12[24:16]	0_0011_0011	Reserved
	CSL12[8:0]	0_0011_0011	Reserved
0xBASE_0034	CSL13[24:16]	0_0011_0011	Reserved
	CSL13[8:0]	0_0011_0011	Reserved
0xBASE_0038	CSL14[24:16]	0_0011_0011	Reserved
	CSL14[8:0]	0_0011_0011	Reserved
0xBASE_003C	CSL15[24:16]	0_0011_0011	Reserved
	CSL15[8:0]	0_0011_0011	Reserved
0xBASE_0040	CSL16[24:16]	0_0011_0011	SerDes registers
	CSL16[8:0]	0_0011_0011	Reserved
0xBASE_0044	CSL17[24:16]	0_0011_0011	Reserved
	CSL17[8:0]	0_0011_0011	Reserved
0xBASE_0048	CSL18[24:16]	0_0011_0011	Reserved
	CSL18[8:0]	0_0011_0011	Reserved
0xBASE_004C	CSL19[24:16]	0_0011_0011	Reserved
	CSL19[8:0]	0_0011_0011	Reserved
0xBASE_0050	CSL20[24:16]	0_0011_0011	Reserved
	CSL20[8:0]	0_0011_0011	Serial peripheral interface 1 (SPI1) registers
0xBASE_0054	CSL21[24:16]	0_0011_0011	QuadSPI (QSPI) controller registers
	CSL21[8:0]	0_0011_0011	Enhanced Secured Digital Host Controller 1 (eSDHC 1) registers
0xBASE_0058	CSL22[24:16]	0_0011_0011	Reserved
	CSL22[8:0]	0_0011_0011	Reserved
0xBASE_005C	CSL23[24:16]	0_0011_0011	I^2C Controller 1 registers
	CSL23[8:0]	0_0011_0011	USB 2.0 controller registers
0xBASE_0060	CSL24[24:16]	0_0011_0011	Reserved
	CSL24[8:0]	0_0011_0011	I^2C Controller 2 registers

Table continues on the next page...

Table 7-1. CSL Device Peripherals (continued)

Offset	CSL Register[bits]	Reset value (in binary)	Peripheral
0xBASE_0064	CSL25[24:16]	0_0011_0011	Reserved
	CSL25[8:0]	0_0011_0011	DUART 1 registers
0xBASE_0068	CSL26[24:16]	0_0011_0011	Watchdog 2 registers
	CSL26[8:0]	0_0011_0011	Watchdog 1 registers
0xBASE_006C	CSL27[24:16]	0_0011_0011	Enhanced DMA (eDMA) registers
	CSL27[8:0]	0_0011_0011	System Counter registers
0xBASE_0070	CSL28[24:16]	0_0011_0011	Direct Memory Access Multiplexer 2
	CSL28[8:0]	0_0011_0011	Direct Memory Access Multiplexer 1
0xBASE_0074	CSL29[24:16]	0_0011_0011	DDR controller registers
	CSL29[8:0]	0_0011_0011	Reserved
0xBASE_0078	CSL30[24:16]	0_0011_0011	DCFG, CCU, RCPM
	CSL30[8:0]	0_0011_0011	Secure Boot ROM controller registers
0xBASE_007C	CSL31[24:16]	0_0011_0011	Security suse processor (SFP) registers
	CSL31[8:0]	0_0011_0011	Thermal Monitoring Unit (TMU) registers
0xBASE_0080	CSL32[24:16]	0_0011_0011	Security Monitor registers
	CSL32[8:0]	0_0011_0011	Supplment configuration unit (SCFG) registers
0xBASE_0084	CSL33[24:16]	0_0011_0011	Reserved
	CSL33[8:0]	0_0011_0011	SEC 5.5 registers
0xBASE_0088	CSL34[24:16]	0_0011_0011	Reserved
	CSL34[8:0]	0_0011_0011	Reserved
0xBASE_008C	CSL35[24:16]	0_0011_0011	General Purpose IO (GPIO) controller 2 registers
	CSL35[8:0]	0_0011_0011	General Purpose IO (GPIO) controller 1 registers
0xBASE_0090	CSL36[24:16]	0_0011_0011	Reserved
	CSL36[8:0]	0_0011_0011	Reserved
0xBASE_0094	CSL37[24:16]	0_0011_0011	Platform Control registers Similar to SCFG, it stores the status and address attributes of the blocked/failed transactions to the modules
	CSL37[8:0]	0_0011_0011	Central Security Unit (CSU) registers
0xBASE_0098	CSL38[24:16]	0_0011_0011	Reserved
	CSL38[8:0]	0_0011_0011	SAI5 registers
0xBASE_009C	CSL39[24:16]	0_0011_0011	Reserved

Table continues on the next page...

Table 7-1. CSL Device Peripherals (continued)

Offset	CSL Register[bits]	Reset value (in binary)	Peripheral
	CSL39[8:0]	0_0011_0011	Reserved
0xBASE_00A0	CSL40[24:16]	0_0011_0011	SDHC2 registers
	CSL40[8:0]	0_0011_0011	Reserved
0xBASE_00A4	CSL41[24:16]	0_0011_0011	Serial Audio Interface 2 (SAI2) controller registers
	CSL41[8:0]	0_0011_0011	Serial Audio Interface 1 (SAI1) controller registers
0xBASE_00A8	CSL42[24:16]	0_0011_0011	Serial Audio Interface 4 (SAI4) controller registers
	CSL42[8:0]	0_0011_0011	Serial Audio Interface 3 (SAI3) controller registers
0xBASE_00AC	CSL43[24:16]	0_0011_0011	FlexTimer Module 2 controller registers
	CSL43[8:0]	0_0011_0011	FlexTimer Module 1 controller registers
0xBASE_00B0	CSL44[24:16]	0_0011_0011	Reserved
	CSL44[8:0]	0_0011_0011	Reserved
0xBASE_00B4	CSL45[24:16]	0_0011_0011	Reserved
	CSL45[8:0]	0_0011_0011	Reserved
0xBASE_00B8	CSL46[24:16]	0_0011_0011	Reserved
	CSL46[8:0]	0_0011_0011	Reserved
CSL47-CSL59		0_0011_0011	Reserved
CSL60[8:0]		0_0011_0011	DCSR

Address: 151_0000h base + 0h offset = 151_0000h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved							L2	SL15	SL14	SL13	SL12	SL11	SL10	SL9	SL8
Reset	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved							L1	SL7	SL6	SL5	SL4	SL3	SL2	SL1	SL0
Reset	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1

CSU_CSL field descriptions

Field	Description
31–25 -	This field is reserved.
24 L2	Lock bit corresponding to the slave. (It is written by secure software.) 0 Not locked. Bits 16-23 can be written by software. 1 Locked. Bits 16-23 cannot be written by software.

Table continues on the next page...

CSU_CSL field descriptions (continued)

Field	Description
23 SL15	SL15 0 Non-secured supervisor write access disabled 1 Non-secured supervisor write access enabled
22 SL14	SL14 0 Non-secured user write access disabled 1 Non-secured user write access enabled
21 SL13	SL13 0 Secured supervisor write access disabled 1 Secured supervisor write access enabled
20 SL12	SL12 0 Secured user write access disabled 1 Secured user write access enabled
19 SL11	SL11 0 Non-secured supervisor read access disabled 1 Non-secured supervisor read access enabled
18 SL10	SL10 0 Non-secured user read access disabled 1 Non-secured user read access enabled
17 SL9	SL9 0 Secured supervisor read access disabled 1 Secured supervisor read access enabled
16 SL8	SL8 0 Secured user read access disabled 1 Secured user read access enabled
15–9 -	This field is reserved.
8 L1	Lock bit corresponding to the slave. (It is written by secure software.) 0 Not locked. Bits 0-7 can be written by software. 1 Locked. Bits 0-7 cannot be written by software.
7 SL7	SL7 0 Non-secured supervisor write access disabled 1 Non-secured supervisor write access enabled
6 SL6	SL6 0 Non-secured user write access disabled 1 Non-secured user write access enabled
5 SL5	SL5

Table continues on the next page...

CSU_CSL field descriptions (continued)

Field	Description
	0 Secured supervisor write access disabled 1 Secured supervisor write access enabled
4 SL4	SL4 0 Secured user write access disabled 1 Secured user write access enabled
3 SL3	SL3 0 Non-secured supervisor read rccess disabled 1 Non-secured supervisor read access enabled
2 SL2	SL2 0 Non-secured user read access disabled 1 Non-secured user Read access enabled
1 SL1	SL1 0 Secured supervisor read access disabled 1 Secured supervisor read access enabled
0 SL0	SL0 0 Secured user read access disabled 1 Secured user read access enabled

7.1.1.2 Secure Access register (CSU_SA*n*)

The following table provides the mapping of CSU secure access register bits.

Table 7-2. Mapping of CSU secure access register bits

Offset	Register Bits	Reset	Description
0xBASE_218	csu_sa0[5-4]	0x0	Software-override bit for SPNIDEN signal connectivity to core, CCI-400, and debug components.
0xBASE_218	csu_sa0[7-6]	0x0	Software-override bit for NIDEN signal connectivity to core, CCI-400, and debug components.
0xBASE_218	csu_sa0[9-8]	0x0	Software-override bit for SPIDEN signal connectivity to core, and debug components
0xBASE_218	csu_sa0[11-10]	0x0	Software-Override bit for DBGEN signal connectivity to core and debug components
0xBASE_218	csu_sa0[13-12]	0x0	CP15SDISABLE for core 0
0xBASE_218	csu_sa0[15-14]	0x0	-

Table continues on the next page...

Table 7-2. Mapping of CSU secure access register bits (continued)

Offset	Register Bits	Reset	Description
0xBASE_218	csu_sa0[17-16]	0x0	CFGDISABLE for GIC-400 and core
0xBASE_218	csu_sa0[19-18]	0x0	Non-secure attribute control for debug components
0xBASE_218	csu_sa0[23-20]	0x0	-
0xBASE_218	csu_sa0[25-24]	0x0	-
0xBASE_218	csu_sa0[29-28]	0x0	-
0xBASE_218	csu_sa0[31-30]	0x0	Secure attribute for PCI Express controller 1
0xBASE_21C	csu_sa1[1-0]	0x0	-
0xBASE_21C	csu_sa1[3-2]	0x0	Trust Zone address space controller bypass mux. The TZASC module is disabled and logically bypassed.
0xBASE_21C	csu_sa1[5-4]	0x0	Secure boot lock for Trust-Zone address space controller registers

Address: 151_0000h base + 218h 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
R W	L15_n	SA15_n	L14_n	SA14_n	L13_n	SA13_n	L12_n	SA12_n	L11_n	SA11_n	L10_n	SA10_n	L9_n	SA9_n	L8_n	SA8_n
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	L7_n	SA7_n	L6_n	SA6_n	L5_n	SA5_n	L4_n	SA4_n	L3_n	SA3_n	L2_n	SA2_n	L1_n	SA1_n	L0_n	SA0_n
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CSU_SA*n* field descriptions

Field	Description
31 L15_n	Lock bit set by secure software. 0 Stands for no lock condition and (2*n)th bit can be written by software. 1 Stands for locking of (2*n) th bit and once set software cannot write on the SA field.
30 SA15_n	Secured access indicator 0 Stands for Secured Access indication for nth Type 1 Master corresponding to this 32-bit register. 1 Stands for Non-Secured Access for that master.

Table continues on the next page...

CSU_SA*n* field descriptions (continued)

Field	Description
29 L14_n	Lock bit set by secure software. 0 Stands for no lock condition and (2*n)th bit can be written by software. 1 Stands for locking of (2*n) th bit and once set software cannot write on the SA field.
28 SA14_n	Secured access indicator 0 Stands for Secured Access indication for nth Type 1 Master corresponding to this 32-bit register. 1 Stands for Non-Secured Access for that master.
27 L13_n	Lock bit set by secure software. 0 Stands for no lock condition and (2*n)th bit can be written by software. 1 Stands for locking of (2*n) th bit and once set software cannot write on the SA field.
26 SA13_n	Secured access indicator 0 Stands for Secured Access indication for nth Type 1 Master corresponding to this 32-bit register. 1 Stands for Non-Secured Access for that master.
25 L12_n	Lock bit set by secure software. 0 Stands for no lock condition and (2*n)th bit can be written by software. 1 Stands for locking of (2*n) th bit and once set software cannot write on the SA field.
24 SA12_n	Secured access indicator 0 Stands for Secured Access indication for nth Type 1 Master corresponding to this 32-bit register. 1 Stands for Non-Secured Access for that master.
23 L11_n	Lock bit set by secure software. 0 Stands for no lock condition and (2*n)th bit can be written by software. 1 Stands for locking of (2*n) th bit and once set software cannot write on the SA field.
22 SA11_n	Secured access indicator 0 Stands for Secured Access indication for nth Type 1 Master corresponding to this 32-bit register. 1 Stands for Non-Secured Access for that master.
21 L10_n	Lock bit set by secure software. 0 Stands for no lock condition and (2*n)th bit can be written by software. 1 Stands for locking of (2*n) th bit and once set software cannot write on the SA field.
20 SA10_n	Secured access indicator 0 Stands for Secured Access indication for nth Type 1 Master corresponding to this 32-bit register. 1 Stands for Non-Secured Access for that master.
19 L9_n	Lock bit set by secure software. 0 Stands for no lock condition and (2*n)th bit can be written by software. 1 Stands for locking of (2*n) th bit and once set software cannot write on the SA field.
18 SA9_n	Secured access indicator 0 Stands for Secured Access indication for nth Type 1 Master corresponding to this 32-bit register. 1 Stands for Non-Secured Access for that master.

Table continues on the next page...

CSU_SA*n* field descriptions (continued)

Field	Description
17 L8_n	Lock bit set by secure software. 0 Stands for no lock condition and (2*n)th bit can be written by software. 1 Stands for locking of (2*n) th bit and once set software cannot write on the SA field.
16 SA8_n	Secured access indicator 0 Stands for Secured Access indication for nth Type 1 Master corresponding to this 32-bit register. 1 Stands for Non-Secured Access for that master.
15 L7_n	Lock bit set by secure software. 0 Stands for no lock condition and (2*n)th bit can be written by software. 1 Stands for locking of (2*n) th bit and once set software cannot write on the SA field.
14 SA7_n	Secured access indicator 0 Stands for Secured Access indication for nth Type 1 Master corresponding to this 32-bit register. 1 Stands for Non-Secured Access for that master.
13 L6_n	Lock bit set by secure software. 0 Stands for no lock condition and (2*n)th bit can be written by software. 1 Stands for locking of (2*n) th bit and once set software cannot write on the SA field.
12 SA6_n	Secured access indicator 0 Stands for Secured Access indication for nth Type 1 Master corresponding to this 32-bit register. 1 Stands for Non-Secured Access for that master.
11 L5_n	Lock bit set by secure software. 0 Stands for no lock condition and (2*n)th bit can be written by software. 1 Stands for locking of (2*n) th bit and once set software cannot write on the SA field.
10 SA5_n	Secured access indicator 0 Stands for Secured Access indication for nth Type 1 Master corresponding to this 32-bit register. 1 Stands for Non-Secured Access for that master.
9 L4_n	Lock bit set by secure software. 0 Stands for no lock condition and (2*n)th bit can be written by software. 1 Stands for locking of (2*n) th bit and once set software cannot write on the SA field.
8 SA4_n	Secured access indicator 0 Stands for Secured Access indication for nth Type 1 Master corresponding to this 32-bit register. 1 Stands for Non-Secured Access for that master.
7 L3_n	Lock bit set by secure software. 0 Stands for no lock condition and (2*n)th bit can be written by software. 1 Stands for locking of (2*n) th bit and once set software cannot write on the SA field.
6 SA3_n	Secured access indicator 0 Stands for Secured Access indication for nth Type 1 Master corresponding to this 32-bit register. 1 Stands for Non-Secured Access for that master.

Table continues on the next page...

CSU_SA*n* field descriptions (continued)

Field	Description
5 L2_n	Lock bit set by secure software. 0 Stands for no lock condition and (2*n)th bit can be written by software. 1 Stands for locking of (2*n) th bit and once set software cannot write on the SA field.
4 SA2_n	Secured access indicator 0 Stands for Secured Access indication for nth Type 1 Master corresponding to this 32-bit register. 1 Stands for Non-Secured Access for that master.
3 L1_n	Lock bit set by secure software. 0 Stands for no lock condition and (2*n)th bit can be written by software. 1 Stands for locking of (2*n) th bit and once set software cannot write on the SA field.
2 SA1_n	Secured access indicator 0 Stands for Secured Access indication for nth Type 1 Master corresponding to this 32-bit register. 1 Stands for Non-Secured Access for that master.
1 L0_n	Lock bit set by secure software. 0 Stands for no lock condition and (2*n)th bit can be written by software. 1 Stands for locking of (2*n) th bit and once set software cannot write on the SA field.
0 SA0_n	Secured access indicator 0 Stands for Secured Access indication for nth Type 1 Master corresponding to this 32-bit register. 1 Stands for Non-Secured Access for that master.

7.1.2 Initialization Policy

For peripherals whose access controls are configured in the CSU

1. Set (and optionally lock) the CSU_CSLn register fields to set each peripheral's access permissions as a target
2. Set (and optionally lock) the CSU_SA register fields

NOTE

Once set, CSU lock bits can only be cleared by a device reset.

7.2 On-Chip RAM memory controller (OCRAM)

The on-chip RAM is implemented as a slave device on the 64-bit system AXI bus. There are two OCRAMs in the chip and access can be controlled through CSU peripheral register programming.

Each OCRAM occupies a 64 KB of address region and the start address of each OCRAM (64 KB each) in the memory map is given below:

- OCRAM1: 0x1000_0000
- OCRAM2: 0x1001_0000.

NOTE

The OCRAM1 and OCRAM2 memory content is not initialized to zero by the chip. The software must initialize both OCRAM1 and OCRAM2 in full width (64-bit) access and then perform a write to bit 5 and 6 at address 0x2014_0534 and 0x2014_0544, respectively with the value 0x0 to clear the single and multi-bit ECC errors for OCRAM1 and OCRAM2. Also, the ECC error interrupt needs to be cleared in GIC (MBEE). Once completed, the host processor enables interrupt related to ECC errors from OCRAM1 and OCRAM2

7.3 Miscellaneous System Control Module (MSCM)

The MSCM (platform control) stores the status and address attributes of a transaction to a peripheral which was blocked by the CSU. The MSCM also enables CSU interrupts.

7.3.1 MSCM Access Control and TrustZone Security (ACTZS)Memory Map/Register Definition

The ACTZS configuration portion of the MSCM programming model map is shown in the table below. It is partitioned into two sections:

Offset addresses 0xC00 - 0xC18 define interrupt configurability of CSU related access violation reporting.

Offset addresses 0xD00 - 0xDDC contain captured access address and attribute information for CSL-detected violations.

Attempted writes to read-only registers are simply ignored (RO/WI). This section contains the target access fault information like CSL*n* attribute check logic plus an array of 128-bit register structures containing captured CSL*n* fault information.

All the register accesses are privilege/supervisor

MSCM_ACTZS memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
152_0C10	ACTZS CSL Interrupt Enable Register (MSCM_ACTZS_CSLIER)	32	R/W	0000_0000h	7.3.1.1/277
152_0C14	ACTZS CSL Interrupt Register (MSCM_ACTZS_CSLIR)	32	R/W	0000_0000h	7.3.1.2/279
152_0C18	ACTZS CSL Interrupt Overrun Register (MSCM_ACTZS_CSOVR)	32	R/W	0000_0000h	7.3.1.3/281

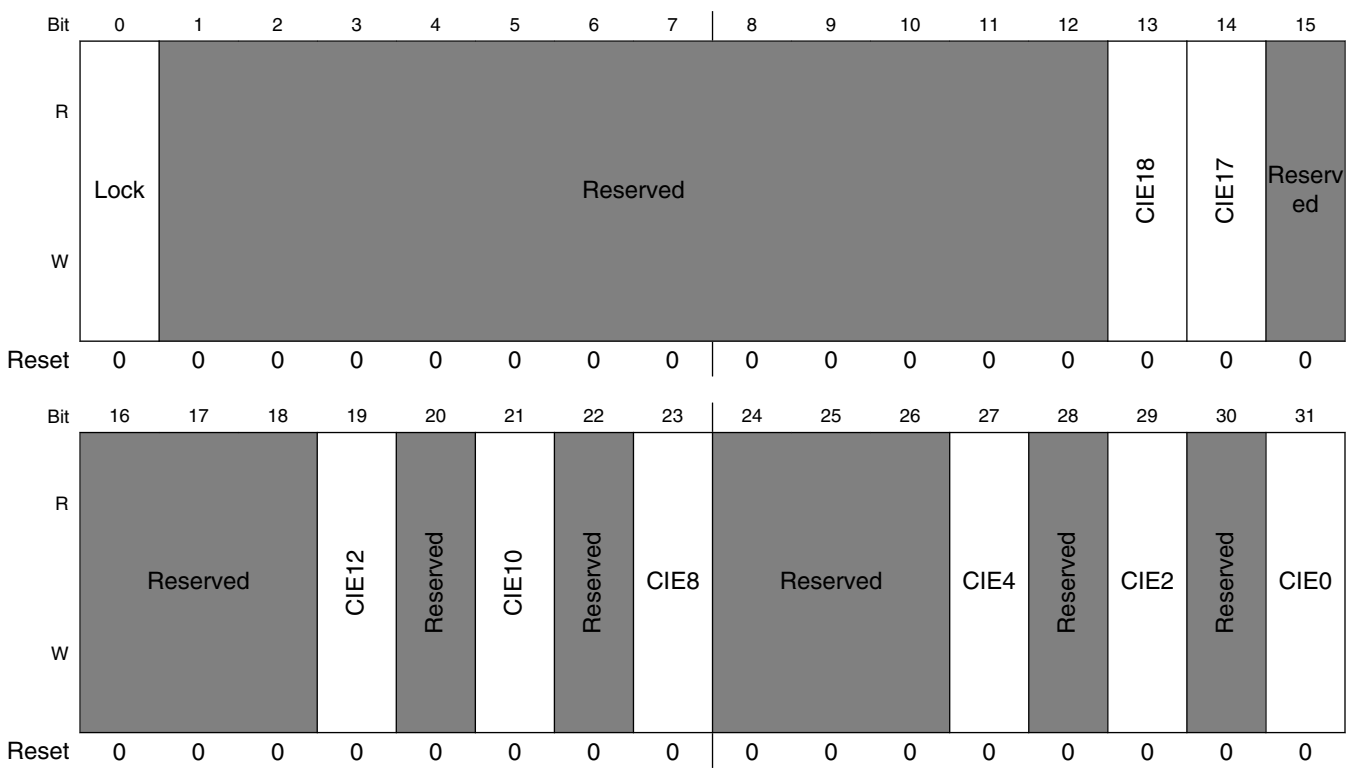
7.3.1.1 ACTZS CSL Interrupt Enable Register (MSCM_ACTZS_CSLIER)

The CSLn interrupt enable register is a 32-bit register containing a bit map of interrupt enables for CSL*n* logic reporting access check violations.

As the individual CSLn levels are evaluated for each bus transfer, access violations are error terminated and the resulting error status flags collected and posted in the MSCM_CSLIR.

The CSLIER also contains a lock bit (RO) that may be set to disable writes to the register, preserving the enabled/disabled state of the CSLn interrupts.

Address: 152_0000h base + C10h offset = 152_0C10h



MSCM_ACTZS_CSLIER field descriptions

Field	Description
0 Lock	Read-Only. This register bit provides a mechanism to "lock" the configuration state defined by MSCM_CSLIER. Once asserted, attempted writes to the MSCM_CSLIER register are ignored until the next system reset clears the flag. 0 writes to the MSCM_CSLIER are allowed 1 writes to the MSCM_CSLIER are ignored
1–12 -	This field is reserved.
13 CIE18	CSLn Interrupt Enable. This field enables/disables each CSLn access violation alert interrupt. if CIE18 = 0, the CSLn access check interrupt is disabled. if CIE18 = 1, the CSLn access check interrupt is enabled. The individual CIE18 are mapped to OCRAM2 memory space.
14 CIE17	CSLn Interrupt Enable. This field enables/disables each CSLn access violation alert interrupt. if CIE17 = 0, the CSLn access check interrupt is disabled. if CIE17 = 1, the CSLn access check interrupt is enabled. The individual CIE17 are mapped to OCRAM1 memory space.
15–18 -	This field is reserved. This field is reserved
19 CIE12	CSLn Interrupt Enable. This field enables/disables each CSLn access violation alert interrupt. if CIE12= 0, the CSLn access check interrupt is disabled. if CIE12= 1, the CSLn access check interrupt is enabled. The individual CIE12 are mapped to USB3 Controller Register space(CCSR).
20 -	This field is reserved. Reserved
21 CIE10	CSLn Interrupt Enable. This field enables/disables each CSLn access violation alert interrupt. if CIE10= 0, the CSLn access check interrupt is disabled. if CIE10= 1, the CSLn access check interrupt is enabled. The individual CIE10 are mapped to SATA Register space(CCSR).
22 -	This field is reserved. Reserved
23 CIE8	CSLn Interrupt Enable. This field enables/disables each CSLn access violation alert interrupt. if CIE8 = 0, the CSLn access check interrupt is disabled. if CIE8 = 1, the CSLn access check interrupt is enabled. The individual CIE8 are mapped to Register Configuration Space of all the IP modules in CCSR Space except PCI Express controllers, SATA controller, USB 3 Controller Register configuration space.
24–26 -	This field is reserved. Reserved
27 CIE4	CSLn Interrupt Enable. This field enables/disables each CSLn access violation alert interrupt. if CIE4 = 0, the CSLn access check interrupt is disabled. if CIE4 = 1, the CSLn access check interrupt is enabled. The individual CIE4 are mapped to QuadSPI memory space.

Table continues on the next page...

MSCM_ACTZS_CSLIER field descriptions (continued)

Field	Description
28 -	This field is reserved. Reserved
29 CIE2	CSLn Interrupt Enable. This field enables/disables each CSLn access violation alert interrupt. if CIE2 = 0, the CSLn access check interrupt is disabled. if CIE2 = 1, the CSLn access check interrupt is enabled. The individual CIE2 are mapped to PCI Express controller 1 Register space(CCSR).
30 -	This field is reserved. Reserved
31 CIE0	CSLn Interrupt Enable. This field enables/disables each CSLn access violation alert interrupt. if CIE0 = 0, the CSLn access check interrupt is disabled. if CIE0 = 1, the CSLn access check interrupt is enabled. The individual CIE0 are mapped to PCI Express controller 1 IO Config Space and memory space.

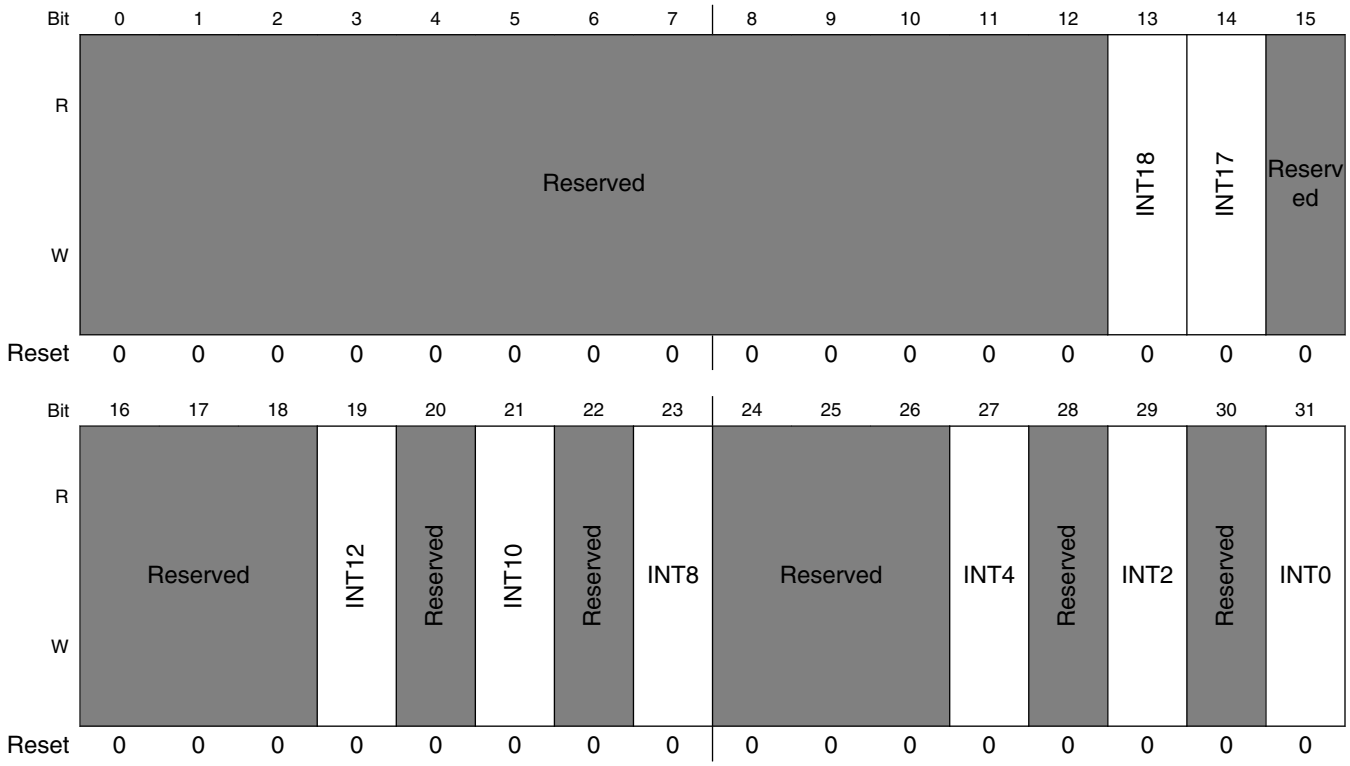
7.3.1.2 ACTZS CSL Interrupt Register (MSCM_ACTZS_CSLIR)

The CSLn interrupt register is a 32-bit register containing a bit map of interrupts for CSL*n* logic reporting access check violations.

As the individual CSL*n* levels are evaluated for each bus transfer, access violations are error terminated and the resulting error status flags collected and posted in the MSCM_CSLIR.

The MSCM_CSLIR records all CSL*n* access violations regardless of the state of the interrupt enable (MSCM_CSLIER). Accordingly, this register can be used by both interrupt service routines and/or bus error exception handlers to quickly determine the source of the CSL*n* access check violation. Each individual interrupt flag in this register is cleared by writing a "1" to it; this would typically be done after the captured fail address and attribute information is retrieved from the appropriate MSCM_CSF*R register. Additionally, the clearing of an interrupt flag in this register also clears the corresponding bit in the MSCM_CSLOVR register and rearms the logic for capturing the failed access information.

Address: 152_0000h base + C14h offset = 152_0C14h



MSCM_ACTZS_CSLIR field descriptions

Field	Description
0–12 -	This field is reserved.
13 INT18	CSLn Interrupt Enable. This field enables/disables each CSLn access violation alert interrupt. if INT18 = 0, the CSLn access check interrupt is disabled. if INT18 = 1, the CSLn access check interrupt is enabled. The individual CIE18 are mapped to OCRAM2 memory space.
14 INT17	CSLn Interrupt Enable. This field enables/disables each CSLn access violation alert interrupt. if INT17 = 0, the CSLn access check interrupt is disabled. if INT17 = 1, the CSLn access check interrupt is enabled. The individual CIE17 are mapped to OCRAM1 memory space.
15–18 -	This field is reserved. This field is reserved.
19 INT12	CSLn Interrupt Enable. This field enables/disables each CSLn access violation alert interrupt. if INT12= 0, the CSLn access check interrupt is disabled. if INT12 = 1, the CSLn access check interrupt is enabled. The individual CIE12 are mapped to USB 3 Controller Register space(CCSR).
20 -	This field is reserved. Reserved
21 INT10	CSLn Interrupt Enable. This field enables/disables each CSLn access violation alert interrupt.

Table continues on the next page...

MSCM_ACTZS_CSLIR field descriptions (continued)

Field	Description
	if INT10 = 0, the CSLn access check interrupt is disabled. if INT10 = 1, the CSLn access check interrupt is enabled. The individual CIE10 are mapped to SATA register space(CCSR).
22 -	This field is reserved. Reserved
23 INT8	CSLn Interrupt Enable. This field enables/disables each CSLn access violation alert interrupt. if INT8 = 0, the CSLn access check interrupt is disabled. if INT8 = 1, the CSLn access check interrupt is enabled. The individual CIE8 are mapped to Register Configuration Space of all the IP modules in CCSR Space except PCI Express controllers, , SATA controller, USB 3 Controller Register configuration space.
24–26 -	This field is reserved. Reserved
27 INT4	CSLn Interrupt Enable. This field enables/disables each CSLn access violation alert interrupt. if INT4 = 0, the CSLn access check interrupt is disabled. if INT4 = 1, the CSLn access check interrupt is enabled. The individual CIE4 are mapped to QuadSPI memory space.
28 -	This field is reserved. Reserved
29 INT2	CSLn Interrupt Enable. This field enables/disables each CSLn access violation alert interrupt. if INT2 = 0, the CSLn access check interrupt is disabled. if INT2 = 1, the CSLn access check interrupt is enabled. The individual CIE2 are mapped to PCI Express controller 1 Register space (CCSR).
30 -	This field is reserved. Reserved
31 INT0	CSLn Interrupt Enable. This field enables/disables each CSLn access violation alert interrupt. if INT0 = 0, the CSLn access check interrupt is disabled. if INT0 = 1, the CSLn access check interrupt is enabled. The individual CIE0 are mapped to PCI Express controller 1 IO config space and memory space.

7.3.1.3 ACTZS CSL Interrupt Overrun Register (MSCM_ACTZS_CSOVR)

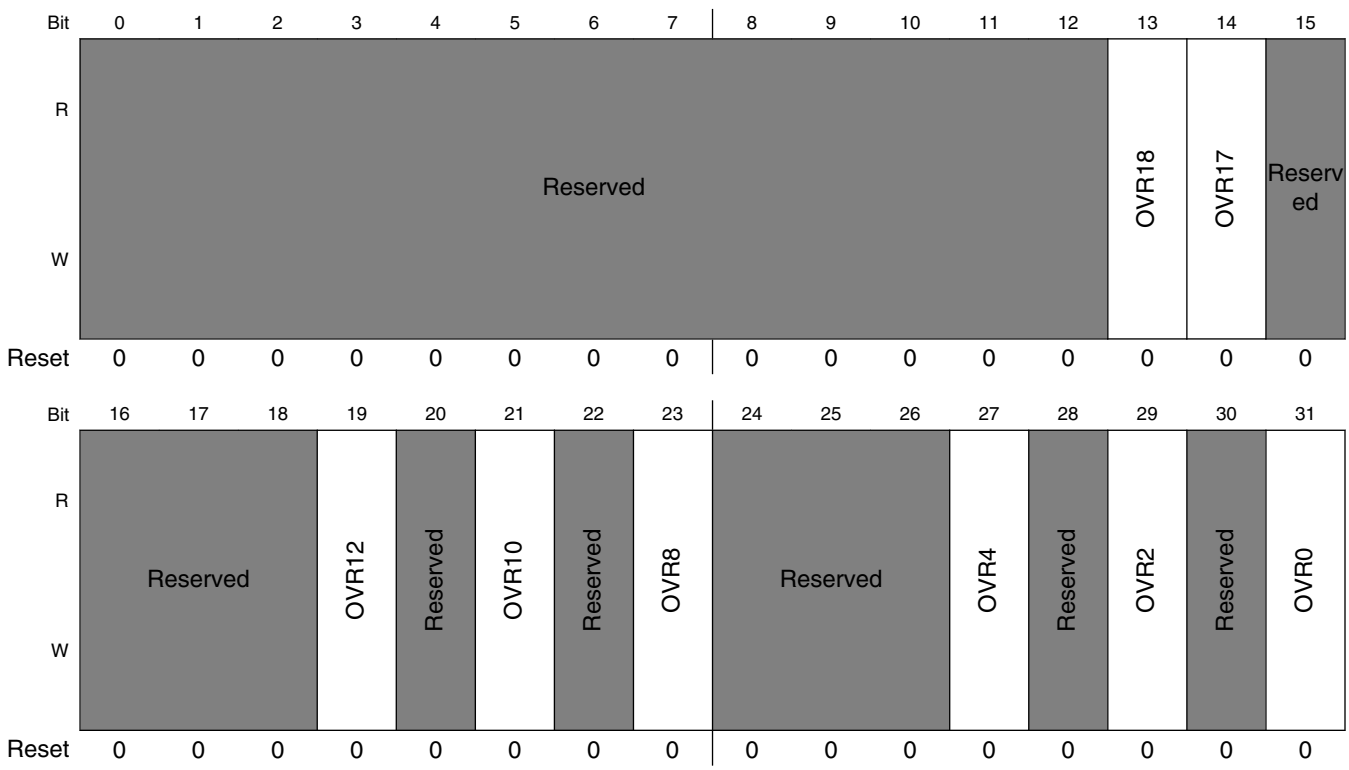
The CSL*n* interrupt overrun register is a 32-bit read-only register containing a bit map of overrun interrupt conditions for the CSLn logic reporting access check violations.

The overrun condition is simply defined as the detection of another CSLn access check violation before the previous one has been full processed and cleared. Stated differently, if a CSLn access check violation is detected and the corresponding MSCM_CSLIR[i] bit still asserted, the MSCM_CSOVR[i] bit is set.

Additionally, for the slave ACTZS ports using the AXI bus protocol, it is possible to detect simultaneous access violations on the read and write channels. For this condition, the CSLn Fail Status Registers (MSCM_CSFARn, MSCM_CSFCRn, MSCM_CSFIRn) capture the information associated with the write access and set the appropriate overrun indicator in the MSCM_CSOVR.

Each individual interrupt flags is cleared by writing a "1" to it and this is typically be done after the captured fail address and attribute information is retrieved from the appropriate MSCM_CSF*R register. The clearing of an interrupt flag in the MSCM_CSLIR also clears the corresponding bit in the MSCM_CSOVR register and rearms the logic for capturing the fail access information.

Address: 152_0000h base + C18h offset = 152_0C18h



MSCM_ACTZS_CSOVR field descriptions

Field	Description
0–12 -	This field is reserved.
13 OVR18	CSLn Interrupt Enable. This field enables/disables each CSLn access violation alert interrupt. if OVR18 = 0, the CSLn access check interrupt is disabled. if OVR18 = 1, the CSLn access check interrupt is enabled. The individual CIE18 are mapped to OCRAM2 memory space.
14 OVR17	CSLn Interrupt Enable. This field enables/disables each CSLn access violation alert interrupt. if OVR17 = 0, the CSLn access check interrupt is disabled.

Table continues on the next page...

MSCM_ACTZS_CSOVR field descriptions (continued)

Field	Description
	if OVR17 = 1, the CSLn access check interrupt is enabled. The individual CIE17 are mapped to OCRAM1 memory space.
15–18 -	This field is reserved. This field is reserved.
19 OVR12	CSLn Interrupt Enable. This field enables/disables each CSLn access violation alert interrupt. if OVR12= 0, the CSLn access check interrupt is disabled. if OVR12 = 1, the CSLn access check interrupt is enabled. The individual CIE12 are mapped to USB 3 Controller Register space(CCSR).
20 -	This field is reserved. Reserved
21 OVR10	CSLn Interrupt Enable. This field enables/disables each CSLn access violation alert interrupt. if OVR10 = 0, the CSLn access check interrupt is disabled. if OVR10 = 1, the CSLn access check interrupt is enabled. The individual CIE10 are mapped to SATA Register space(CCSR).
22 -	This field is reserved. Reserved
23 OVR8	CSLn Interrupt Enable. This field enables/disables each CSLn access violation alert interrupt. if OVR8 = 0, the CSLn access check interrupt is disabled. if OVR8 = 1, the CSLn access check interrupt is enabled. The individual CIE8 are mapped to Register Configuration Space of all the IP modules in CCSR Space except PCI Express controllers, , SATA controller, USB 3 Controller Register config space.
24–26 -	This field is reserved. Reserved
27 OVR4	CSLn Interrupt Enable. This field enables/disables each CSLn access violation alert interrupt. if OVR4 = 0, the CSLn access check interrupt is disabled. if OVR4 = 1, the CSLn access check interrupt is enabled. The individual CIE4 are mapped to QuadSPI memory space.
28 -	This field is reserved. Reserved
29 OVR2	CSLn Interrupt Enable. This field enables/disables each CSLn access violation alert interrupt. if OVR2 = 0, the CSLn access check interrupt is disabled. if OVR2 = 1, the CSLn access check interrupt is enabled. The individual CIE2 are mapped to PCI Express controller 1 Register space(CCSR).
30 -	This field is reserved. Reserved
31 OVR0	CSLn Interrupt Enable. This field enables/disables each CSLn access violation alert interrupt. if OVR0 = 0, the CSLn access check interrupt is disabled. if OVR0 = 1, the CSLn access check interrupt is enabled. The individual CIE0 are mapped to PCI Express controller 1 IO config space and memory space.

7.3.2 ACTZS CSLn Fail Status Capture Registers (Memory Map/ Register Definition)

This section of the MSCM_ACTZS programming model contains an array of four word (128-bit) data values containing address and attribute information corresponding to CSLn access check violations. The format of this data structure is identical to the fail status information captured by the TZASC when they detect a security violation.

When a CSL*n* access check violation is detected, the bus transaction is error terminated, the appropriate bit in the MSCM_CSLIR set and the fail address and attribute information captured in the corresponding data structure. The contents of the captured fail data is unaffected until the interrupt flag is cleared by writing a 1 to it, at which time, the capturing of fail information is rearmed.

The LS1 implementation supports n = [0-14] and contains an array of ten 128-bit data structures and four reserved structures as defined in table below.

Table 7-3. MSCM CSLn Fail Status Capture Registers

Base Offset Address	Source
0xD00	PCI Express controller 1 IO Config Space and memory space
0xD10	Reserved
0xD20	PCI Express controller 1 Register space(CCSR)
0xD30	Reserved
0xD40	QuadSPI memory space
0xD50-0xD70	Reserved
0xD80	Register configuration space of all the IP modules in CCSR Space except PCI Express controllers, SATA controller, USB 3 controller register config space
0xD90	Reserved
0xDA0	SATA Register space(CCSR)
0xDB0	Reserved
0xDC0	USB 3 controller Register space(CCSR)
0xDD0 - 0xE00	Reserved
0xE10	OCRAM1 Memory Space
0xE20	OCRAM2 Memory Space
0xE30 - 0xFFC	Reserved

All the register accesses are privilege/supervisor.

MSCM memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
152_0D00	ACTZS CSLn Fail Status Address (Low) Register (MSCM_CSFAR0)	32	R/W	0000_0000h	7.3.2.1/286
152_0D08	ACTZS CSLn Fail Status Control Register (MSCM_CSFCR0)	32	R/W	0000_0000h	7.3.2.2/287
152_0D0C	ACTZS CSLn Fail Status Master ID Register (MSCM_CSFIR0)	32	R/W	0000_0000h	7.3.2.3/288
152_0D20	ACTZS CSLn Fail Status Address (Low) Register (MSCM_CSFAR2)	32	R/W	0000_0000h	7.3.2.1/286
152_0D28	ACTZS CSLn Fail Status Control Register (MSCM_CSFCR2)	32	R/W	0000_0000h	7.3.2.2/287
152_0D2C	ACTZS CSLn Fail Status Master ID Register (MSCM_CSFIR2)	32	R/W	0000_0000h	7.3.2.3/288
152_0D40	ACTZS CSLn Fail Status Address (Low) Register (MSCM_CSFAR4)	32	R/W	0000_0000h	7.3.2.1/286
152_0D48	ACTZS CSLn Fail Status Control Register (MSCM_CSFCR4)	32	R/W	0000_0000h	7.3.2.2/287
152_0D4C	ACTZS CSLn Fail Status Master ID Register (MSCM_CSFIR4)	32	R/W	0000_0000h	7.3.2.3/288
152_0D80	ACTZS CSLn Fail Status Address (Low) Register (MSCM_CSFAR8)	32	R/W	0000_0000h	7.3.2.4/289
152_0D88	ACTZS CSLn Fail Status Control Register (MSCM_CSFCR8)	32	R/W	0000_0000h	7.3.2.5/290
152_0D8C	ACTZS CSLn Fail Status Master ID Register (MSCM_CSFIR8)	32	R/W	0000_0000h	7.3.2.6/291
152_0DA0	ACTZS CSLn Fail Status Address (Low) Register (MSCM_CSFAR10)	32	R/W	0000_0000h	7.3.2.4/289
152_0DA8	ACTZS CSLn Fail Status Control Register (MSCM_CSFCR10)	32	R/W	0000_0000h	7.3.2.5/290
152_0DAC	ACTZS CSLn Fail Status Master ID Register (MSCM_CSFIR10)	32	R/W	0000_0000h	7.3.2.6/291
152_0DC0	ACTZS CSLn Fail Status Address (Low) Register (MSCM_CSFAR12)	32	R/W	0000_0000h	7.3.2.7/292
152_0DC8	ACTZS CSLn Fail Status Control Register (MSCM_CSFCR12)	32	R/W	0000_0000h	7.3.2.8/293
152_0DCC	ACTZS CSLn Fail Status Master ID Register (MSCM_CSFIR12)	32	R/W	0000_0000h	7.3.2.9/294
152_0E10	ACTZS CSLn Fail Status Address (Low) Register (MSCM_CSFAR17)	32	R/W	0000_0000h	7.3.2.10/ 295
152_0E18	ACTZS CSLn Fail Status Control Register (MSCM_CSFCR17)	32	R/W	0000_0000h	7.3.2.11/ 296
152_0E1C	ACTZS CSLn Fail Status Master ID Register (MSCM_CSFIR17)	32	R/W	0000_0000h	7.3.2.12/ 297
152_0E20	ACTZS CSLn Fail Status Address (Low) Register (MSCM_CSFAR18)	32	R/W	0000_0000h	7.3.2.10/ 295

Table continues on the next page...

MSCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
152_0E28	ACTZS CSLn Fail Status Control Register (MSCM_CSFCR18)	32	R/W	0000_0000h	7.3.2.11/ 296
152_0E2C	ACTZS CSLn Fail Status Master ID Register (MSCM_CSFIR18)	32	R/W	0000_0000h	7.3.2.12/ 297

7.3.2.1 ACTZS CSLn Fail Status Address (Low) Register (MSCM_CSFAR*n*)

The CSFARn is a 32-bit read-only register for capturing the low-order 32 bits of the 40-bit physical address of the last captured access check violation detected by the CSLn logic. When the CSLn logic detects an access violation, the address and attributes associated with the memory reference are loaded into the CSFARn, CSFCRn and CSFIRn registers, and the appropriate flag in the CSL*n* interrupt register (CSLIR) is asserted.

For the slave ACTZS ports using the AXI bus protocol, it is possible to detect simultaneous access violations on the read and write channels. For this condition, the CSL*n* fail status registers (MSCM_CSFARn, MSCM_CSFCRn, MSCM_CSFIRn) capture the information associated with the write access and additionally set the appropriate overrun indicator in the MSCM_CSOVR.

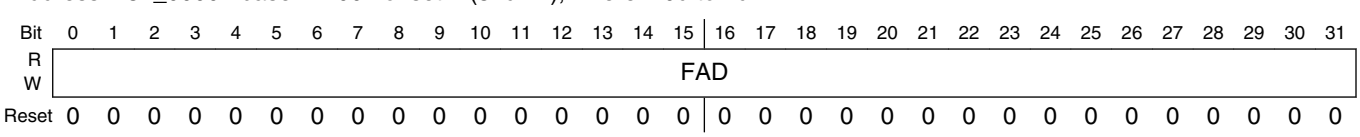
The contents of the MSCM_CSFARn is unaffected until the corresponding MSCM_CSLIR interrupt flag is cleared by writing a 1 to it, at which time, the capturing of fail information is rearmed.

Attempted privileged mode writes are ignored.

NOTE

The next sequential word at offset address 0xD04 + 16*n is reserved for systems where the address space is larger than 4 Gbytes.

Address: 152_0000h base + D00h offset + (32d × i), where i=0d to 2d



MSCM_CSFAR*n* field descriptions

Field	Description
0–31 FAD	CSLn Fail Address. This read-only field specifies the system address from the last captured CSLn access check violation.

7.3.2.2 ACTZS CSLn Fail Status Control Register (MSCM_CSFCR*n*)

The CSFCRn is a 32-bit read-only register for capturing specific attribute bits of the last captured access check violation detected by the CSLn logic. When the CSLn logic detects an access violation, the address and attributes associated with the memory reference are loaded into the CSFARn, CSFCRn and CSFIRn registers, and the appropriate flag in the CSLn Interrupt Register (CSLIR) is asserted.

For the slave ACTZS ports using the AXI bus protocol, it is possible to detect simultaneous access violations on the read and write channels. For this condition, the CSLn Fail Status Registers (MSCM_CSFARn, MSCM_CSFCRn, MSCM_CSFIRn) capture the information associated with the write access and additionally set the appropriate overrun indicator in the MSCM_CSOVR.

The contents of the MSCM_CSFCRn is unaffected until the corresponding MSCM_CSLIR interrupt flag is cleared by writing a 1 to it, at which time, the capturing of fail information is rearmed.

Attempted privileged mode writes are ignored.

Address: 152_0000h base + D08h offset + (32d × i), where i=0d to 2d

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

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

MSCM_CSFCR*n* field descriptions

Field	Description
0–6 -	This field is reserved. Reserved
7 FWT	CSLn Fail Write. This read-only field specifies the read/write attribute from the last captured CSLn access check violation. if FWT = 0, then the last captured CSLn access check violation was a read. if FWT = 1, then the last captured CSLn access check violation was a write.

Table continues on the next page...

MSCM_CSFCR*n* field descriptions (continued)

Field	Description
8–9 -	This field is reserved. Reserved
10 FNS	CSLn Fail Nonsecure. This read-only field specifies the secure/nonsecure attribute from the last captured CSLn access check violation. if FNS = 0, the last captured CSLn access check violation was a secure access. if FNS = 1, the last captured CSLn access check violation was a nonsecure access.
11 FPR	CSLn Fail Privileged. This read-only field specifies the user/privileged attribute from the last captured CSLn access check violation. if FPR = 0, the last captured CSLn access check violation was a user mode access. if FPR = 1, the last captured CSLn access check violation was a privileged access.
12–31 -	This field is reserved. Reserved

7.3.2.3 ACTZS CSLn Fail Status Master ID Register (MSCM_CSFIR*n*)

The CSFIRn is a 32-bit read-only register for capturing the 4 LSB bits of the AXI ID attribute of the last captured access check violation detected by the CSLn logic. When the CSLn logic detects an access violation, the address and attributes associated with the memory reference are loaded into the CSFARn, CSFCRn and CSFIRn registers, and the appropriate flag in the CSL*n* interrupt register (CSLIR) is asserted.

For the slave ACTZS ports using the AXI bus protocol, it is possible to detect simultaneous access violations on the read and write channels. For this condition, the CSLn fail status registers (MSCM_CSFARn, MSCM_CSFCRn, MSCM_CSFIRn) capture the information associated with the write access and additionally set the appropriate overrun indicator in the MSCM_CSOVR.

The contents of the MSCM_CSFIRn is unaffected until the corresponding MSCM_CSLIR interrupt flag is cleared by writing a 1 to it, at which time, the capturing of fail information is rearmed.

Attempted privileged mode writes are ignored.

Address: 152_0000h base + D0Ch offset + (32d × i), where i=0d to 2d

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

MSCM_CSFIR*n* field descriptions

Field	Description
0–26 -	This field is reserved. Reserved
27–31 FMID	CSLn Fail Master ID. This read-only field specifies the master ID from the last captured CSLn access check violation.

7.3.2.4 ACTZS CSLn Fail Status Address (Low) Register (MSCM_CSFAR*n*)

The CSFARn is a 32-bit read-only register for capturing the low-order 32 bits of the address of the last captured access check violation detected by the CSLn logic. When the CSLn logic detects an access violation, the address and attributes associated with the memory reference are loaded into the CSFARn, CSFCRn and CSFIRn registers, and the appropriate flag in the CSL*n* interrupt register (CSLIR) is asserted.

For the slave ACTZS ports using the AXI bus protocol, it is possible to detect simultaneous access violations on the read and write channels. For this condition, the CSL*n* fail status registers (MSCM_CSFARn, MSCM_CSFCRn, MSCM_CSFIRn) capture the information associated with the write access and additionally set the appropriate overrun indicator in the MSCM_CSOVR.

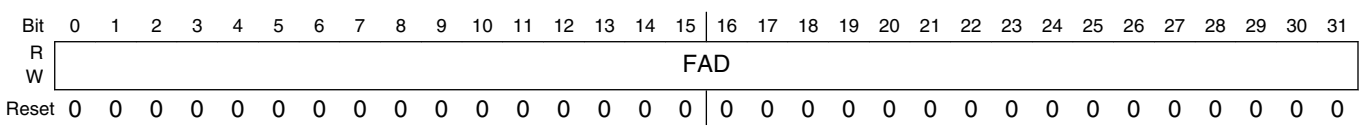
The contents of the MSCM_CSFARn is unaffected until the corresponding MSCM_CSLIR interrupt flag is cleared by writing a 1 to it, at which time, the capturing of fail information is rearmed.

Attempted privileged mode writes are ignored.

NOTE

The next sequential word at offset address 0xD04 + 16*n is reserved for systems where the address space is larger than 4 Gbytes.

Address: 152_0000h base + D80h offset + (32d × i), where i=0d to 1d



MSCM_CSFAR*n* field descriptions

Field	Description
0–31 FAD	CSLn Fail Address. This read-only field specifies the system address from the last captured CSLn access check violation.

7.3.2.5 ACTZS CSLn Fail Status Control Register (MSCM_CSFCR*n*)

The CSFCRn is a 32-bit read-only register for capturing specific attribute bits of the last captured access check violation detected by the CSLn logic. When the CSLn logic detects an access violation, the address and attributes associated with the memory reference are loaded into the CSFARn, CSFCRn and CSFIRn registers, and the appropriate flag in the CSLn Interrupt Register (CSLIR) is asserted.

For the slave ACTZS ports using the AXI bus protocol, it is possible to detect simultaneous access violations on the read and write channels. For this condition, the CSLn Fail Status Registers (MSCM_CSFARn, MSCM_CSFCRn, MSCM_CSFIRn) capture the information associated with the write access and additionally set the appropriate overrun indicator in the MSCM_CSOVR.

The contents of the MSCM_CSFCRn is unaffected until the corresponding MSCM_CSLIR interrupt flag is cleared by writing a 1 to it, at which time, the capturing of fail information is rearmed.

Attempted privileged mode writes are ignored.

Address: 152_0000h base + D88h offset + (32d × i), where i=0d to 1d

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

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

MSCM_CSFCR*n* field descriptions

Field	Description
0–6 -	This field is reserved. Reserved
7 FWT	CSLn Fail Write. This read-only field specifies the read/write attribute from the last captured CSLn access check violation. if FWT = 0, then the last captured CSLn access check violation was a read. if FWT = 1, then the last captured CSLn access check violation was a write.
8–9 -	This field is reserved. Reserved
10 FNS	CSLn Fail Nonsecure. This read-only field specifies the secure/nonsecure attribute from the last captured CSLn access check violation. if FNS = 0, the last captured CSLn access check violation was a secure access. if FNS = 1, the last captured CSLn access check violation was a nonsecure access.

Table continues on the next page...

MSCM_CSFCR*n* field descriptions (continued)

Field	Description
11 FPR	CSLn Fail Privileged. This read-only field specifies the user/privileged attribute from the last captured CSLn access check violation. if FPR = 0, the last captured CSLn access check violation was a user mode access. if FPR = 1, the last captured CSLn access check violation was a privileged access.
12–31 -	This field is reserved. Reserved

7.3.2.6 ACTZS CSLn Fail Status Master ID Register (MSCM_CSFIR*n*)

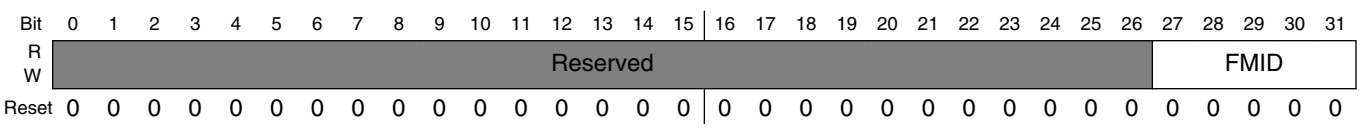
The CSFIRn is a 32-bit read-only register for capturing the 4 LSB bits of the AXI ID attribute of the last captured access check violation detected by the CSLn logic. When the CSLn logic detects an access violation, the address and attributes associated with the memory reference are loaded into the CSFARn, CSFCRn and CSFIRn registers, and the appropriate flag in the CSL*n* interrupt register (CSLIR) is asserted.

For the slave ACTZS ports using the AXI bus protocol, it is possible to detect simultaneous access violations on the read and write channels. For this condition, the CSLn fail status registers (MSCM_CSFARn, MSCM_CSFCRn, MSCM_CSFIRn) capture the information associated with the write access and additionally set the appropriate overrun indicator in the MSCM_CSOVR.

The contents of the MSCM_CSFIRn is unaffected until the corresponding MSCM_CSLIR interrupt flag is cleared by writing a 1 to it, at which time, the capturing of fail information is rearmed.

Attempted privileged mode writes are ignored.

Address: 152_0000h base + D8Ch offset + (32d × i), where i=0d to 1d



MSCM_CSFIR*n* field descriptions

Field	Description
0–26 -	This field is reserved. Reserved
27–31 FMID	CSLn Fail Master ID. This read-only field specifies the master ID from the last captured CSLn access check violation.

7.3.2.7 ACTZS CSLn Fail Status Address (Low) Register (MSCM_CSFAR*n*)

The CSFARn is a 32-bit read-only register for capturing the low-order 32 bits of the address of the last captured access check violation detected by the CSLn logic. When the CSLn logic detects an access violation, the address and attributes associated with the memory reference are loaded into the CSFARn, CSFCRn and CSFIRn registers, and the appropriate flag in the CSL*n* interrupt register (CSLIR) is asserted.

For the slave ACTZS ports using the AXI bus protocol, it is possible to detect simultaneous access violations on the read and write channels. For this condition, the CSL*n* fail status registers (MSCM_CSFARn, MSCM_CSFCRn, MSCM_CSFIRn) capture the information associated with the write access and additionally set the appropriate overrun indicator in the MSCM_CSOVR.

The contents of the MSCM_CSFARn is unaffected until the corresponding MSCM_CSLIR interrupt flag is cleared by writing a 1 to it, at which time, the capturing of fail information is rearmed.

Attempted privileged mode writes are ignored.

NOTE

The next sequential word at offset address 0xD04 + 16*n is reserved for systems where the address space is larger than 4 Gbytes.

Address: 152_0000h base + DC0h offset + (16d × i), where i=0d to 0d

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

MSCM_CSFAR*n* field descriptions

Field	Description
0–31 FAD	CSLn Fail Address. This read-only field specifies the system address from the last captured CSLn access check violation.

7.3.2.8 ACTZS CSLn Fail Status Control Register (MSCM_CSFCR*n*)

The CSFCRn is a 32-bit read-only register for capturing specific attribute bits of the last captured access check violation detected by the CSLn logic. When the CSLn logic detects an access violation, the address and attributes associated with the memory reference are loaded into the CSFARn, CSFCRn and CSFIRn registers, and the appropriate flag in the CSLn Interrupt Register (CSLIR) is asserted.

For the slave ACTZS ports using the AXI bus protocol, it is possible to detect simultaneous access violations on the read and write channels. For this condition, the CSLn Fail Status Registers (MSCM_CSFARn, MSCM_CSFCRn, MSCM_CSFIRn) capture the information associated with the write access and additionally set the appropriate overrun indicator in the MSCM_CSOVR.

The contents of the MSCM_CSFCRn is unaffected until the corresponding MSCM_CSLIR interrupt flag is cleared by writing a 1 to it, at which time, the capturing of fail information is rearmed.

Attempted privileged mode writes are ignored.

Address: 152_0000h base + DC8h offset + (16d × i), where i=0d to 0d

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

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

MSCM_CSFCR*n* field descriptions

Field	Description
0–6 -	This field is reserved. Reserved
7 FWT	CSLn Fail Write. This read-only field specifies the read/write attribute from the last captured CSLn access check violation. if FWT = 0, then the last captured CSLn access check violation was a read. if FWT = 1, then the last captured CSLn access check violation was a write.
8–9 -	This field is reserved. Reserved
10 FNS	CSLn Fail Nonsecure. This read-only field specifies the secure/nonsecure attribute from the last captured CSLn access check violation. if FNS = 0, the last captured CSLn access check violation was a secure access. if FNS = 1, the last captured CSLn access check violation was a nonsecure access.

Table continues on the next page...

MSCM_CSFCR*n* field descriptions (continued)

Field	Description
11 FPR	CSLn Fail Privileged. This read-only field specifies the user/privileged attribute from the last captured CSLn access check violation. if FPR = 0, the last captured CSLn access check violation was a user mode access. if FPR = 1, the last captured CSLn access check violation was a privileged access.
12–31 -	This field is reserved. Reserved

7.3.2.9 ACTZS CSLn Fail Status Master ID Register (MSCM_CSFIR*n*)

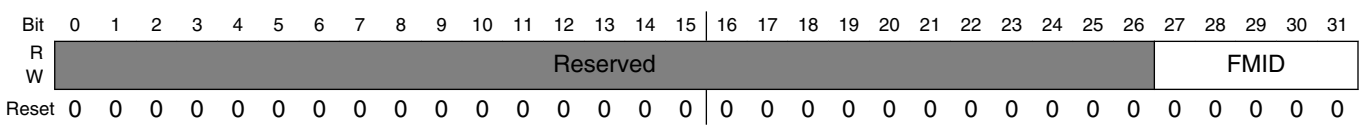
The CSFIRn is a 32-bit read-only register for capturing the 4 LSB bits of the AXI ID attribute of the last captured access check violation detected by the CSLn logic. When the CSLn logic detects an access violation, the address and attributes associated with the memory reference are loaded into the CSFARn, CSFCRn and CSFIRn registers, and the appropriate flag in the CSL*n* interrupt register (CSLIR) is asserted.

For the slave ACTZS ports using the AXI bus protocol, it is possible to detect simultaneous access violations on the read and write channels. For this condition, the CSLn fail status registers (MSCM_CSFARn, MSCM_CSFCRn, MSCM_CSFIRn) capture the information associated with the write access and additionally set the appropriate overrun indicator in the MSCM_CSOVR.

The contents of the MSCM_CSFIRn is unaffected until the corresponding MSCM_CSLIR interrupt flag is cleared by writing a 1 to it, at which time, the capturing of fail information is rearmed.

Attempted privileged mode writes are ignored.

Address: 152_0000h base + DCCh offset + (16d × i), where i=0d to 0d



MSCM_CSFIR*n* field descriptions

Field	Description
0–26 -	This field is reserved. Reserved
27–31 FMID	CSLn Fail Master ID. This read-only field specifies the master ID from the last captured CSLn access check violation.

7.3.2.10 ACTZS CSLn Fail Status Address (Low) Register (MSCM_CSFAR*n*)

The CSFARn is a 32-bit read-only register for capturing the low-order 32 bits of the address of the last captured access check violation detected by the CSLn logic. When the CSLn logic detects an access violation, the address and attributes associated with the memory reference are loaded into the CSFARn, CSFCRn and CSFIRn registers, and the appropriate flag in the CSL*n* interrupt register (CSLIR) is asserted.

For the slave ACTZS ports using the AXI bus protocol, it is possible to detect simultaneous access violations on the read and write channels. For this condition, the CSL*n* fail status registers (MSCM_CSFARn, MSCM_CSFCRn, MSCM_CSFIRn) capture the information associated with the write access and additionally set the appropriate overrun indicator in the MSCM_CSOVR.

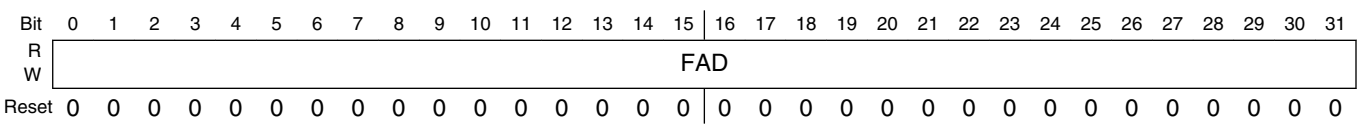
The contents of the MSCM_CSFARn is unaffected until the corresponding MSCM_CSLIR interrupt flag is cleared by writing a 1 to it, at which time, the capturing of fail information is rearmed.

Attempted privileged mode writes are ignored.

NOTE

The next sequential word at offset address 0xD04 + 16*n is reserved for systems where the address space is larger than 4 Gbytes.

Address: 152_0000h base + E10h offset + (16d × i), where i=0d to 1d



MSCM_CSFAR*n* field descriptions

Field	Description
0–31 FAD	CSLn Fail Address. This read-only field specifies the system address from the last captured CSLn access check violation.

7.3.2.11 ACTZS CSLn Fail Status Control Register (MSCM_CSFCR*n*)

The CSFCRn is a 32-bit read-only register for capturing specific attribute bits of the last captured access check violation detected by the CSLn logic. When the CSLn logic detects an access violation, the address and attributes associated with the memory reference are loaded into the CSFARn, CSFCRn and CSFIRn registers, and the appropriate flag in the CSLn Interrupt Register (CSLIR) is asserted.

For the slave ACTZS ports using the AXI bus protocol, it is possible to detect simultaneous access violations on the read and write channels. For this condition, the CSLn Fail Status Registers (MSCM_CSFARn, MSCM_CSFCRn, MSCM_CSFIRn) capture the information associated with the write access and additionally set the appropriate overrun indicator in the MSCM_CSOVR.

The contents of the MSCM_CSFCRn is unaffected until the corresponding MSCM_CSLIR interrupt flag is cleared by writing a 1 to it, at which time, the capturing of fail information is rearmed.

Attempted privileged mode writes are ignored.

Address: 152_0000h base + E18h offset + (16d × i), where i=0d to 1d

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

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

MSCM_CSFCR*n* field descriptions

Field	Description
0–6 -	This field is reserved. Reserved
7 FWT	CSLn Fail Write. This read-only field specifies the read/write attribute from the last captured CSLn access check violation. if FWT = 0, then the last captured CSLn access check violation was a read. if FWT = 1, then the last captured CSLn access check violation was a write.
8–9 -	This field is reserved. Reserved
10 FNS	CSLn Fail Nonsecure. This read-only field specifies the secure/nonsecure attribute from the last captured CSLn access check violation. if FNS = 0, the last captured CSLn access check violation was a secure access. if FNS = 1, the last captured CSLn access check violation was a nonsecure access.

Table continues on the next page...

MSCM_CSFCR*n* field descriptions (continued)

Field	Description
11 FPR	CSLn Fail Privileged. This read-only field specifies the user/privileged attribute from the last captured CSLn access check violation. if FPR = 0, the last captured CSLn access check violation was a user mode access. if FPR = 1, the last captured CSLn access check violation was a privileged access.
12–31 -	This field is reserved. Reserved

7.3.2.12 ACTZS CSLn Fail Status Master ID Register (MSCM_CSFIR*n*)

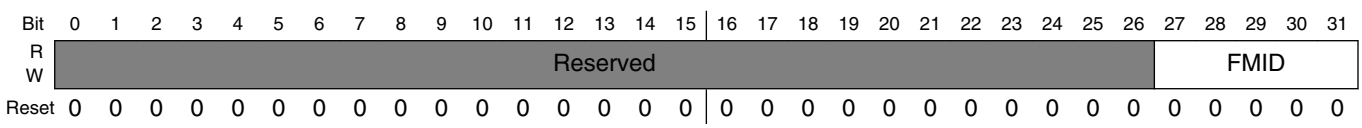
The CSFIRn is a 32-bit read-only register for capturing the 4 LSB bits of the AXI ID attribute of the last captured access check violation detected by the CSLn logic. When the CSLn logic detects an access violation, the address and attributes associated with the memory reference are loaded into the CSFARn, CSFCRn and CSFIRn registers, and the appropriate flag in the CSL*n* interrupt register (CSLIR) is asserted.

For the slave ACTZS ports using the AXI bus protocol, it is possible to detect simultaneous access violations on the read and write channels. For this condition, the CSLn fail status registers (MSCM_CSFARn, MSCM_CSFCRn, MSCM_CSFIRn) capture the information associated with the write access and additionally set the appropriate overrun indicator in the MSCM_CSOVR.

The contents of the MSCM_CSFIRn is unaffected until the corresponding MSCM_CSLIR interrupt flag is cleared by writing a 1 to it, at which time, the capturing of fail information is rearmed.

Attempted privileged mode writes are ignored.

Address: 152_0000h base + E1Ch offset + (16d × i), where i=0d to 1d



MSCM_CSFIR*n* field descriptions

Field	Description
0–26 -	This field is reserved. Reserved
27–31 FMID	CSLn Fail Master ID. This read-only field specifies the master ID from the last captured CSLn access check violation.

Chapter 8
System Counter

8.1 Overview

The system counter implements the Arm generic timer requirements mentioned in the chapter "System Level Implementation of the Generic Timer" of Arm V8, for Arm V8-A architecture profile.

The Arm generic timer provides a single system counter to measure the passing of time in real-time, even if individual cores or clusters of cores may be operating at different frequencies.

This timer also supports virtual counters that measure the passing of virtual time. That is, a virtual counter can measure the passing of time on a particular virtual machine.

The timer can trigger events after a period of time has passed.

- It can be used as count-up or as count-down timer.
- It can operate in real-time or in virtual-time.

8.2 Secure system counter register descriptions

The secure system counter memory map provides register list for the secure world to access the counter values.

8.2.1 Secure_System_counter Memory map

system_counter base address: 2B0_0000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Control register (CNTCR)	32	RW	0000_0000h
8h	LSB of counter count value register (CNTCV1)	32	RW	0000_0000h
Ch	MSB of counter count value register (CNTCV2)	32	RW	0000_0000h
20h	Counter frequency mode table base frequency register (CNTFID0)	32	RW	See description.

8.2.2 Control register (CNTCR)

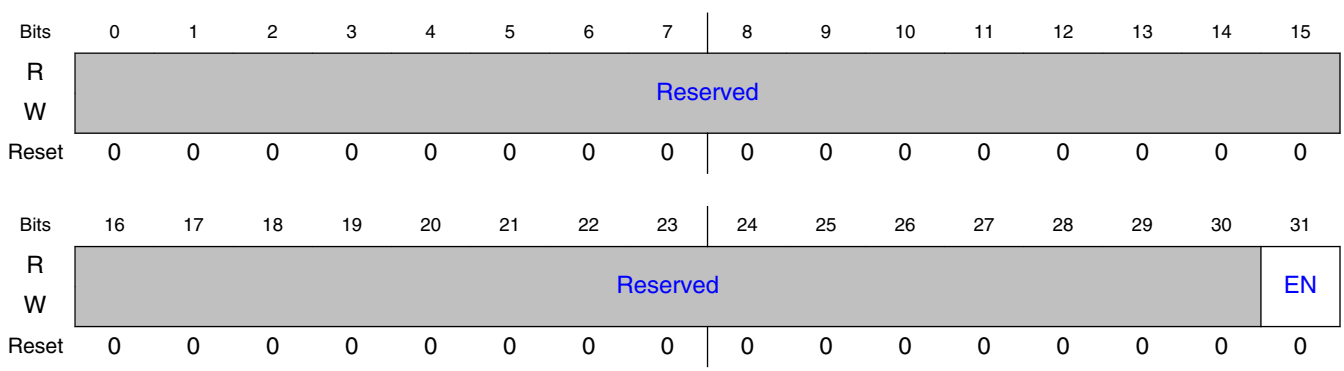
8.2.2.1 Offset

Register	Offset
CNTCR	0h

8.2.2.2 Function

This register enables the counter.

8.2.2.3 Diagram



8.2.2.4 Fields

Field	Function
0-30 —	-
31 EN	Enables the counter 0b - System counter disabled. 1b - System counter enabled.

8.2.3 LSB of counter count value register (CNTCV1)

8.2.3.1 Offset

Register	Offset
CNTCV1	8h

8.2.3.2 Function

This register indicates the current LSB count value of the 64-bit counter. The register is writable only by secure writes. When the counter is enabled, the effect of writing the register is unpredictable.

8.2.3.3 Diagram

Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CNCTV1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CNCTV1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

8.2.3.4 Fields

Field	Function
0-31 CNCTV1	Counter count value bits CNCTV[31:0]. The EN bit must be cleared before writing to this bits, otherwise the effect of the write is unpredictable. Writes to these registers are rare. In a system that uses security, these registers are writable only by secure writes. 00000000000000000000000000000000b - System counter disabled. 00000000000000000000000000000001b - System counter enabled.

8.2.4 MSB of counter count value register (CNTCV2)

8.2.4.1 Offset

Register	Offset
CNTCV2	Ch

8.2.4.2 Function

This register indicates the current MSB count value of the 64-bit counter. The register is writable only by secure writes. When the counter is enabled, the effect of writing the register is unpredictable.

8.2.4.3 Diagram

Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CNCTV2															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CNCTV2															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

8.2.4.4 Fields

Field	Function
0-31 CNCTV2	Counter Count Value bits CNCTV[63:32]. The EN bit must be cleared before writing to this bits, otherwise the effect of the write is UNPREDICTABLE. Writes to these registers are rare. In a system that uses security, these registers are writable only by Secure writes.

8.2.5 Counter frequency mode table base frequency register (CNTFID0)

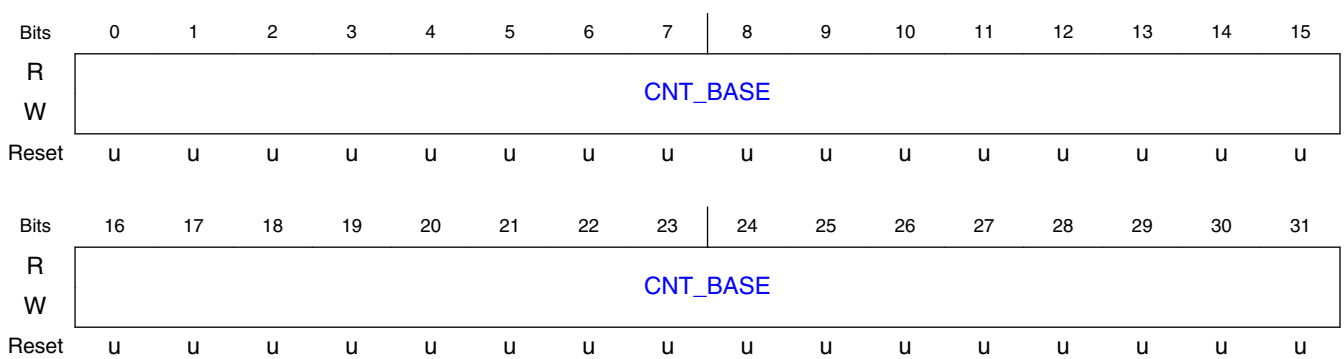
8.2.5.1 Offset

Register	Offset
CNTFID0	20h

8.2.5.2 Function

This register specifies the base frequency of the system counter. The system counter always works with SYSCLK_FREQ/4 frequency clock. The initial value of this register is 25 MHz (100 MHz/4). Software needs to update this register initially to reflect the correct frequency based on system clock frequency.

8.2.5.3 Diagram



8.2.5.4 Fields

Field	Function
0-31 CNT_BASE	System counter base frequency. Holds the clock frequency of the counter. Initial value is set to 25 MHz that corresponds to 100 MHz SYSCLK_FREQ.

8.3 Non-secure system counter register descriptions

The non-secure counter memory map provides register list for the non-secure world to access the counter values.

8.3.1 Non_secure_system_counter Memory map

sys_counter_non_secure base address: 2B1_0000h

Offset	Register	Width (In bits)	Access	Reset value
0h	LSB of Counter Count Value (CNCTV_RO1)	32	RO	0000_0000h
4h	MSB of counter count value register (CNCTV2_RO2)	32	RO	0000_0000h

8.3.2 LSB of Counter Count Value (CNCTV_RO1)

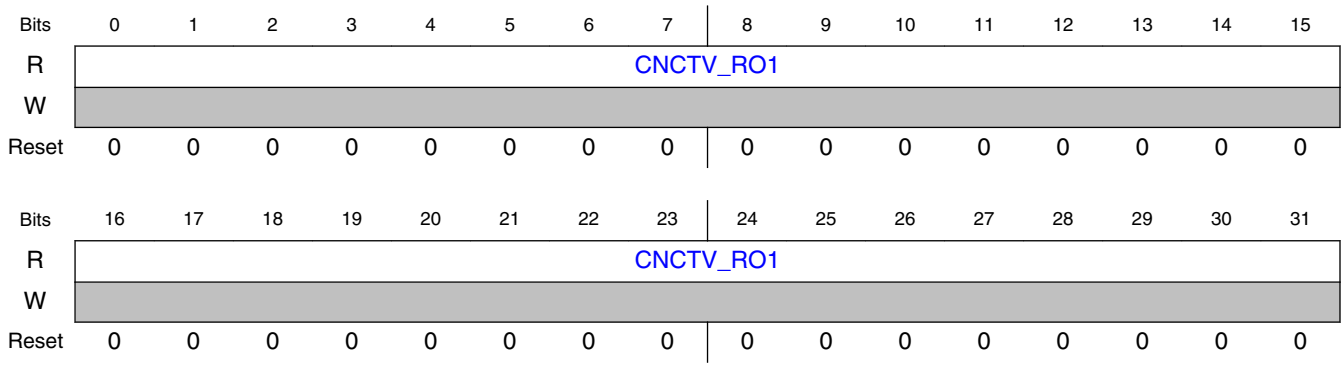
8.3.2.1 Offset

Register	Offset
CNCTV_RO1	0h

8.3.2.2 Function

This register is read-only and contains the current LSB count value of the 64-bit counter.

8.3.2.3 Diagram



8.3.2.4 Fields

Field	Function
0-31 CNCTV_RO1	Counter count value bits CNCTV[31:0]

8.3.3 MSB of counter count value register (CNCTV2_RO2)

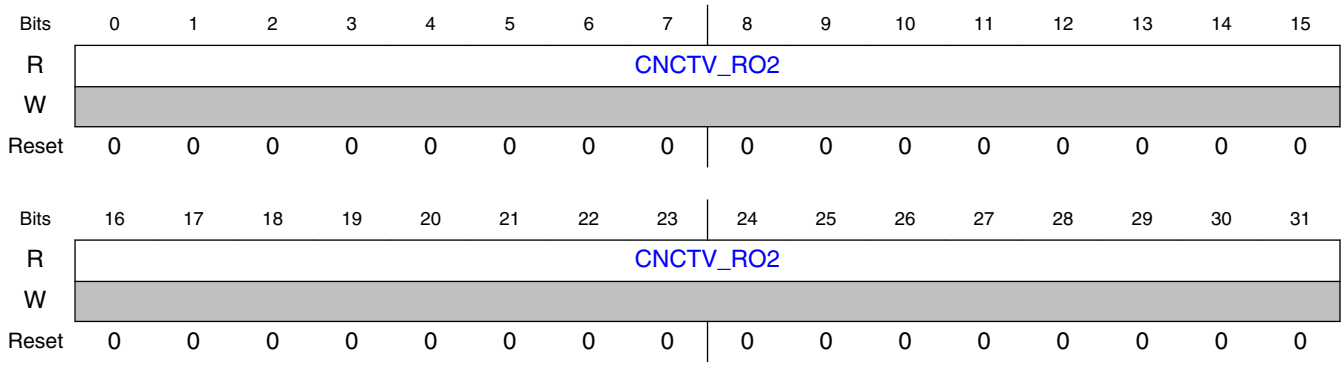
8.3.3.1 Offset

Register	Offset
CNCTV2_RO2	4h

8.3.3.2 Function

The register is read-only and contains the current MSB count value of the 64-bit counter.

8.3.3.3 Diagram



8.3.3.4 Fields

Field	Function
0-31 CNCTV_RO2	Counter count value bits CNCTV[63:32]

Chapter 9
Cache Coherent Interconnect (CCI-400)

9.1 The Cache coherent interconnect (CCI-400) module as implemented on the chip

The register offset in CCI-400 starts from 0x9000 and should be treated as 0x0000 due to implementation. See Figure 9-1 below for CCI-400 connectivity.

CCI-400 supports coherent/snoop transactions (write unique and read once) from masters. A master that doesn't have the capability to generate coherent/snoop transactions have chip defined registers (Snoop configuration control register (SNPCNFGCR)) to generate snoop transactions. Once the corresponding field in this register is set, all the transactions are considered as snoop/coherent data.

Unlike the other masters where certain data transactions can be coherent/snoop, the masters whose snoop is enabled through this register, all transactions are considered snoop/coherent.

Similarly there exists QoS (WR_QoS1 register (WR_QoS1), WR QoS2 register (WR_QoS2), RD QoS1 register (RD_QoS1), and RD QoS2 register (RD_QoS2)) for programming QoS for data transactions from the corresponding masters. The masters which can generate QoS are not mentioned in this register.

Only nERRORIRQ interrupt is connected to the interrupt controller.

NOTE

CCI-400 has per core lock-reservation monitor individually for secure and non-secure exclusive (ARLOCK/AWLOCK) transactions (that is, a total of eight monitors). These transactions are terminated at the CCI-400 level, as the system beyond CCI-400 does not support the "lock transactions". Transaction should be marked with domain setting, such as "01- Inner sharable" or "10- Outersharable".

CCI-400 includes the following connections:

- 2x full ACE slave ports. One (S4 128-b ACE interface) is used to connect the A53 core platform. The other (S3) is left unused.
- Slave interfaces:
  - S0: Ace-Lite slave interface (SEC and PFE masters through NIC-301; all initiators operate at DDR/4 frequency)
  - S1: Ace-Lite slave interface (PCI Express and SATA masters through NIC-301; all initiators operate at DDR/4 frequency)
  - S2: Ace-Lite slave interface (USB 3.0, USB 2.0, and 2x eSDHC masters through NIC-301; all initiators operate at DDR/4 frequency)
  - S3: Interface unused
  - S4: Ace slave interface (A53 core cluster)
- Master interfaces:
  - M0: ACE-Lite master interface (OCRAM, QuadSPI; PCI express controller slaves for outbound transactions)
  - M1: ACE-Lite master interface (DDR controller)
  - M2: Unused

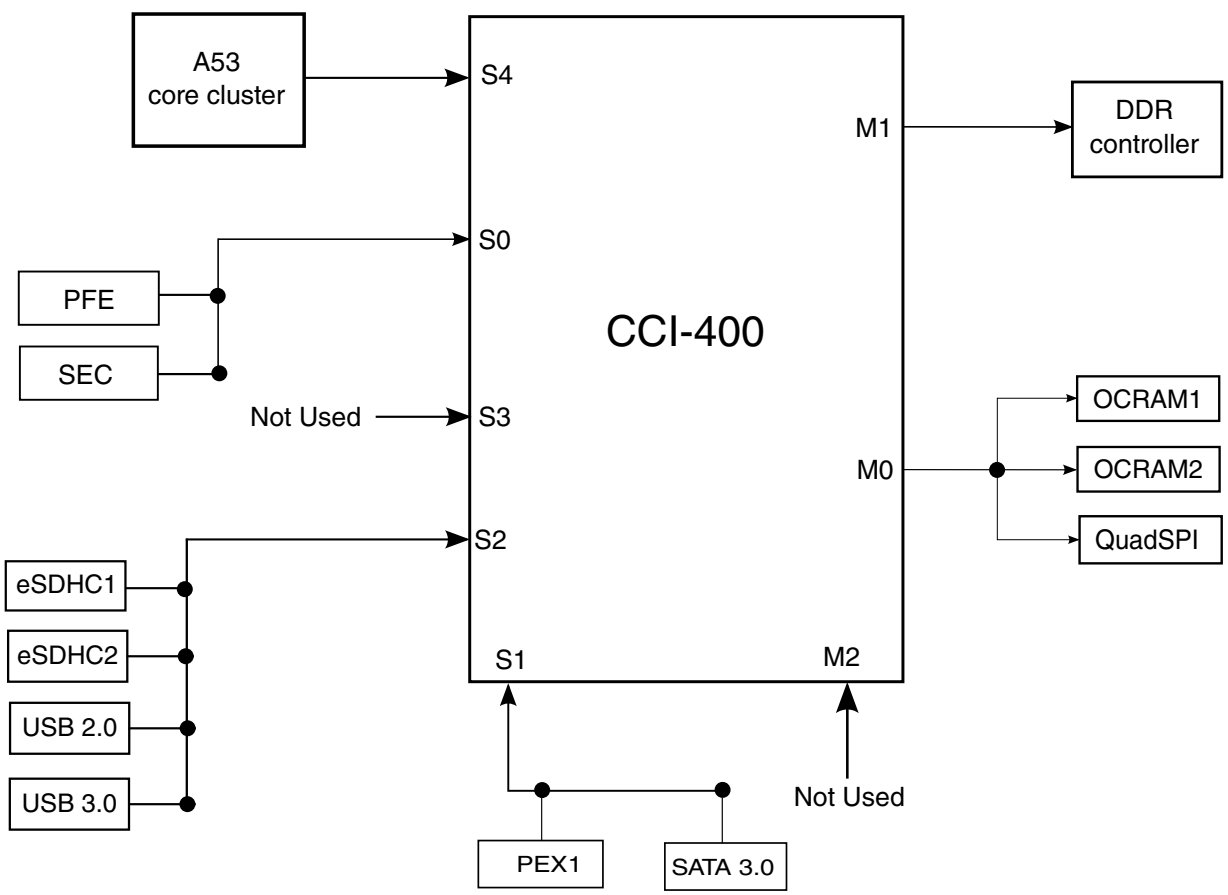


Figure 9-1. Chip interconnect

The following table provides the snoop transaction configurations for different modules:

Table 9-1. Snoop transaction configuration for modules

Module	Snoop transaction configuration
SEC	All transactions from SEC are tagged as snoop configuration if the SCFG_SNPCNFGCR[SECRDSNP], SCFG_SNPCNFGCR[SECWRSNP], SEC_MCFGR[ARCACHE], and SEC_MCFGR[AWCACHE] bits are set. Refer Snoop configuration control register (SNPCNFGCR) and Master Configuration (MCFGR) in security reference manual for details.
PEX1	Snoop transactions are dependent on PCI Express protocol and controlled by PCI Express IATU registers. Refer PEX register descriptions for details.
SATA	All transactions from SATA are tagged as snoop configuration if the SCFG_SNPCNFGCR[SATARDSNP], SCFG_SNPCNFGCR[SATAWRSNP], SATA_AXICC[EARC], and SATA_AXICC[EAWC] bits are set. Refer Snoop configuration control register (SNPCNFGCR) and AXICC - AXI CACHE Control register (AXI cache control register (AXICC)) for details.
USB3.0	All transactions from USB 3.0 are tagged as snoop configuration if the SCFG_SNPCNFGCR[USB*n*RDSNP], SCFG_SNPCNFGCR[USB*n*WRSNP] and USBx_GSBUSCFG0 bits are set. Refer Snoop configuration control register (SNPCNFGCR) and Global SoC bus configuration register 0 (GSBUSCFG0) for details.
eDMA	All transactions from eDMA are tagged as snoop configuration if the SCFG_SNPCNFGCR[EDMASNP] bit is set. Refer Snoop configuration control register (SNPCNFGCR) for details.
USB2	Snoop transactions are controlled by USB_SNOOP1 and USB_SNOOP2 registers. Refer Snoop n (SNOOP1 - SNOOP2) for details.
eSDHC1, 2	Snoop transactions are controlled by eSDHC control register. Refer eSDHC control register (ESDHCCTL) for details.
PFE	Snoop transactions are controlled by WR_QoS1, WR_QoS2, RD_QoS1, and RD_QoS2 registers. Refer WR_QoS1 register (WR_QoS1), WR QoS2 register (WR_QoS2), RD QoS1 register (RD_QoS1), and RD QoS2 register (RD_QoS2) for details.

9.2 Register Descriptions

9.2.1 CCI400 Registers Memory map

CCI base address: 118_0000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Control Override Register (Control_Override_Register)	32	RW	See description.
4h	Speculation Control Register (Speculation_Control_Register)	32	RW	See description.
8h	Secure Access Register (Secure_Access_Register)	32	RW	See description.

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
Ch	Status Register (Status_Register)	32	RO	See description.
10h	Imprecise Error Register (Imprecise_Error_Register)	32	RW	See description.
1000h	Snoop Control Registers (Snoop_Control_Register_S0)	32	RW	See description.
1004h	Shareable Override Registers (Shareable_Override_Register_S0)	32	RW	See description.
1100h	Read Channel QoS Value Override Register (Read_Qos_Override_Register_S0)	32	RW	See description.
1104h	Write Qos Override Register (Write_Qos_Override_Register_S0)	32	RW	See description.
110Ch	Qos Control Register (Qos_Control_Register_S0)	32	RW	See description.
1110h	Max OT Registers (Max_OT_Register_S0)	32	RW	See description.
1130h	Regulator Target Registers (Target_Latency_Register_S0)	32	RW	See description.
1138h	QoS Range Register (Qos_Range_Register_S0)	32	RW	See description.
2000h	Snoop Control Registers (Snoop_Control_Register_S1)	32	RW	See description.
2004h	Shareable Override Registers (Shareable_Override_Register_S1)	32	RW	See description.
2100h	Read Channel QoS Value Override Register (Read_Qos_Override_Register_S1)	32	RW	See description.
2104h	Write Qos Override Register (Write_Qos_Override_Register_S1)	32	RW	See description.
210Ch	Qos Control Register (Qos_Control_Register_S1)	32	RW	See description.
2110h	Max OT Registers (Max_OT_Register_S1)	32	RW	See description.
2130h	Regulator Target Registers (Target_Latency_Register_S1)	32	RW	See description.
2138h	QoS Range Register (Qos_Range_Register_S1)	32	RW	See description.
2268h	QoS Regulator Scale Factor Registers (Latency_Regulation_Register_S0)	32	RW	See description.
3000h	Snoop Control Registers (Snoop_Control_Register_S2)	32	RW	See description.
3004h	Shareable Override Registers (Shareable_Override_Register_S2)	32	RW	See description.
3100h	Read Channel QoS Value Override Register (Read_Qos_Override_Register_S2)	32	RW	See description.
3104h	Write Qos Override Register (Write_Qos_Override_Register_S2)	32	RW	See description.

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
310Ch	Qos Control Register (Qos_Control_Register_S2)	32	RW	See description.
3110h	Max OT Registers (Max_OT_Register_S2)	32	RW	See description.
3130h	Regulator Target Registers (Target_Latency_Register_S2)	32	RW	See description.
3138h	QoS Range Register (Qos_Range_Register_S2)	32	RW	See description.
3268h	QoS Regulator Scale Factor Registers (Latency_Regulation_Register_S1)	32	RW	See description.
4000h	Snoop Control Registers (Snoop_Control_Register_S3)	32	RW	See description.
4004h	Shareable Override Registers (Shareable_Override_Register_S3)	32	RW	See description.
4100h	Read Channel QoS Value Override Register (Read_Qos_Override_Register_S3)	32	RW	See description.
4104h	Write Qos Override Register (Write_Qos_Override_Register_S3)	32	RW	See description.
410Ch	Qos Control Register (Qos_Control_Register_S3)	32	RW	See description.
4110h	Max OT Registers (Max_OT_Register_S3)	32	RW	See description.
4130h	Regulator Target Registers (Target_Latency_Register_S3)	32	RW	See description.
4138h	QoS Range Register (Qos_Range_Register_S3)	32	RW	See description.
4268h	QoS Regulator Scale Factor Registers (Latency_Regulation_Register_S2)	32	RW	See description.
5000h	Snoop Control Registers (Snoop_Control_Register_S4)	32	RW	See description.
5004h	Shareable Override Registers (Shareable_Override_Register_S4)	32	RW	See description.
5100h	Read Channel QoS Value Override Register (Read_Qos_Override_Register_S4)	32	RW	See description.
5104h	Write Qos Override Register (Write_Qos_Override_Register_S4)	32	RW	See description.
510Ch	Qos Control Register (Qos_Control_Register_S4)	32	RW	See description.
5110h	Max OT Registers (Max_OT_Register_S4)	32	RW	See description.
5130h	Regulator Target Registers (Target_Latency_Register_S4)	32	RW	See description.
5138h	QoS Range Register (Qos_Range_Register_S4)	32	RW	See description.
5268h	QoS Regulator Scale Factor Registers (Latency_Regulation_Register_S3)	32	RW	See description.

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
6268h	QoS Regulator Scale Factor Registers (Latency_Regulation_Register_S4)	32	RW	See description.

9.2.2 Control Override Register (Control_Override_Register)

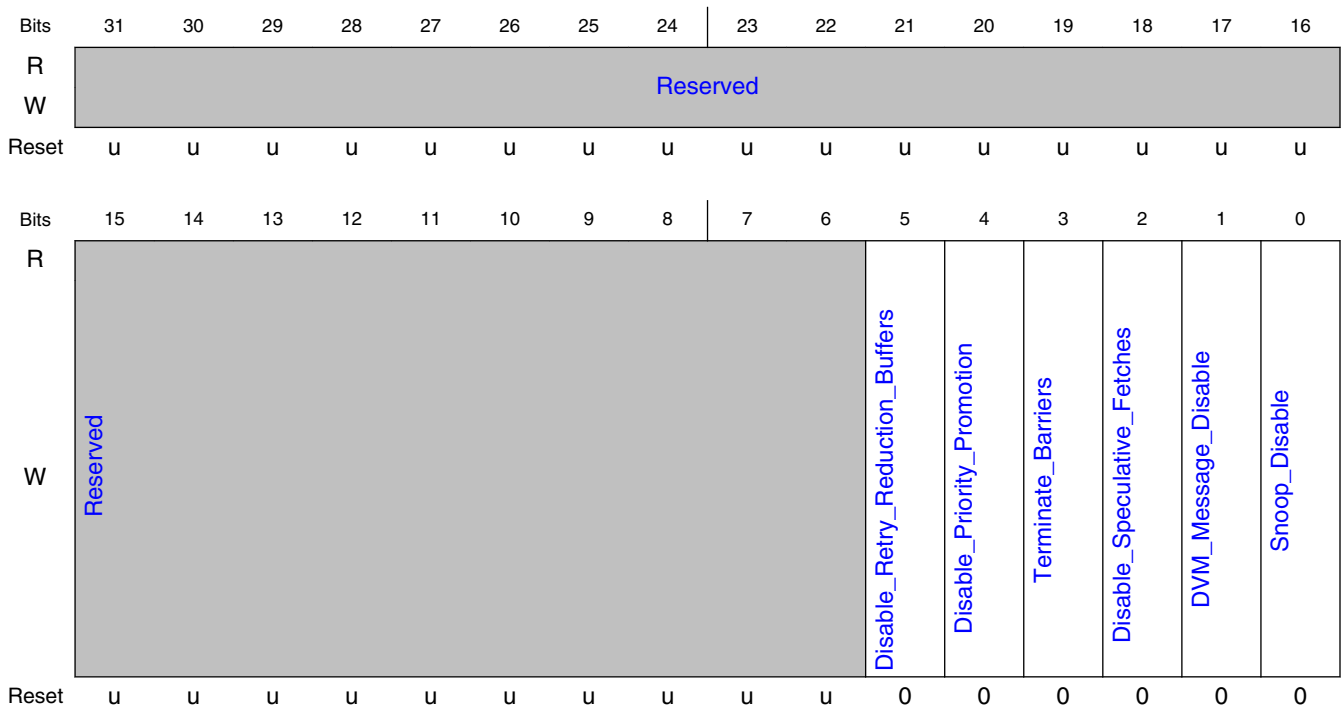
9.2.2.1 Offset

Register	Offset
Control_Override_Register	0h

9.2.2.2 Function

The Control Override register is an additional control register that provides a fail-safe override for some CCI-400 functions, if these cause problems that you cannot otherwise work around. If you cannot avoid using them, only set them using non-bufferable transactions, and before barriers, shareable transactions, or DVM messages are issued into the CCI-400. This can be, for example, very early in the boot sequence, prior to the installation of any Secure OS. You can access the Control Override Register using Secure transactions only, irrespective of the programming of the Secure Access Register . Available in all CCI-400 configurations.

9.2.2.3 Diagram



9.2.2.4 Fields

Field	Function
31-6 —	Reserved
5 Disable_Retry_Reduction_Buffers	Disable retry reduction buffers for speculative fetches 0b - Retry reduction buffers enabled. 1b - Retry reduction buffers disabled.
4 Disable_Priority_Promotion	ARQOSARBS inputs are ignored 0b - The CCI-400 uses ARQOSARBS inputs to promote the priority of earlier requests. 1b - The CCI-400 ignores ARQOSARBS inputs.
3 Terminate_Barriers	Terminate Barriers 0b - Master interfaces terminate barriers according to the BARRIERTERMINATE inputs. 1b - All master interfaces terminate barriers.
2 Disable_Speculative_Fetches	Disable Speculative Fetches 0b - Send speculative fetches according to the Speculation Control Register. See Speculation Control Register. 1b - Disable speculative fetches from all master interfaces.
1	DVM Message Disable 0b - Send DVM messages according to the Snoop Control Registers. See Snoop Control Registers.

Table continues on the next page...

Field	Function
DVM_Message_Disable	1b - Disable propagation of all DVM messages.
0 Snoop_Disable	Snoop Disable 0b - Snoop masters according to the Snoop Control Registers. See Snoop Control Registers. 1b - Disable all snoops, but not DVM messages.

9.2.3 Speculation Control Register (Speculation_Control_Register)

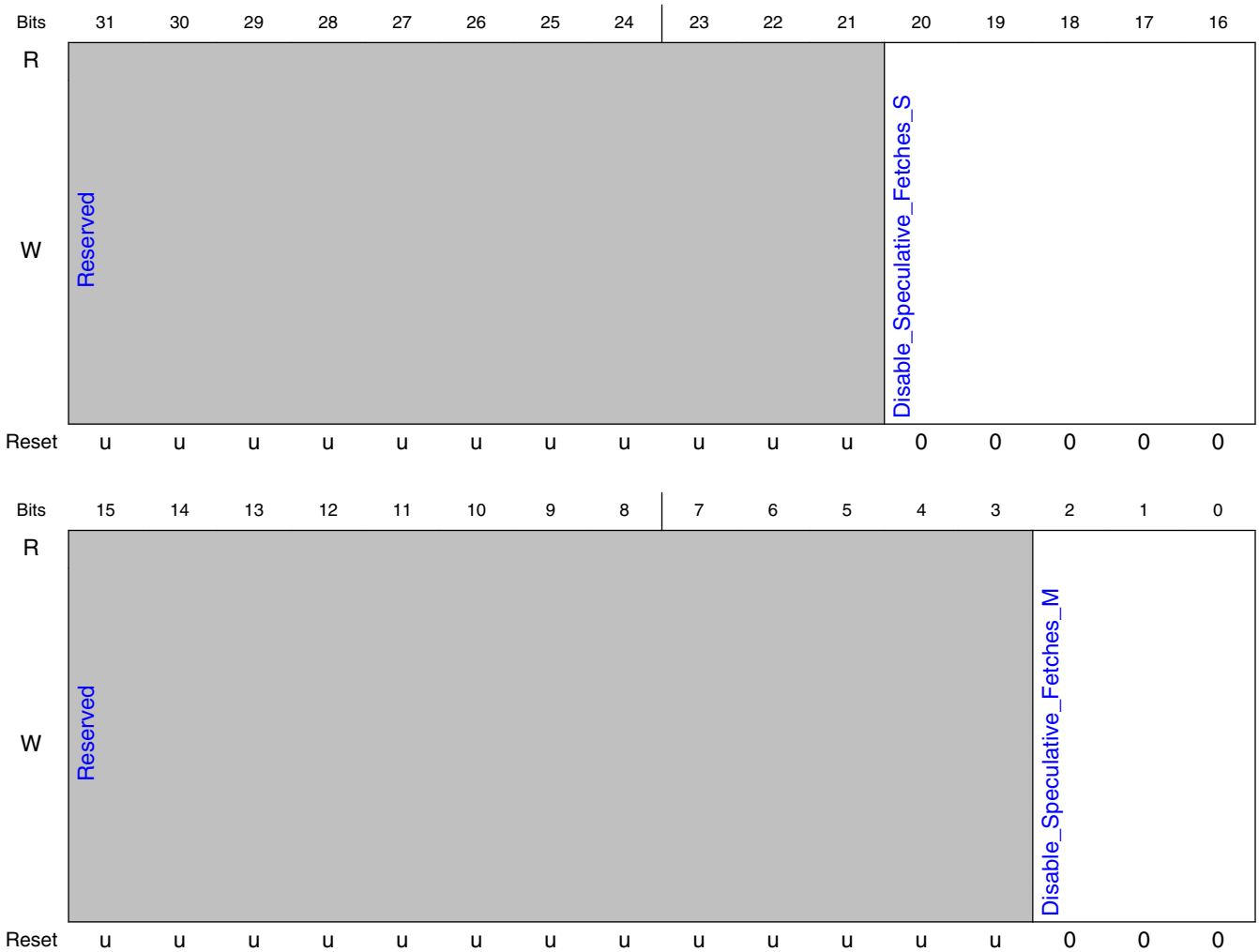
9.2.3.1 Offset

Register	Offset
Speculation_Control_Register	4h

9.2.3.2 Function

The Speculation Control register disables speculative fetches for a master interface or for traffic through a specific slave interface. Speculative fetches are not issued if they are disabled in either the slave or master interface for a particular transaction. Access controlled by Secure Access Register, see Secure Access Register. Available in all CCI-400 configurations.

9.2.3.3 Diagram



9.2.3.4 Fields

Field	Function
31-21 —	Reserved
20-16 Disable_Speculative_Fetches_S	Disable speculative fetches from slave Disable speculative fetches for transactions through a slave interface. One bit for each slave interface: S4, S3, S2, S1, and S0: 00000b - Enable speculative fetches. 00001b - Disable speculative fetches.
15-3 —	Reserved

Table continues on the next page...

Field	Function
2-0 Disable_Speculative_Fetches_M	Disable speculative fetches from master Disable speculative fetches from a master interface. One bit for each master interface: M2, M1, and M0. 000b - Enable speculative fetches. 001b - Disable speculative fetches.

9.2.4 Secure Access Register (Secure_Access_Register)

9.2.4.1 Offset

Register	Offset
Secure_Access_Register	8h

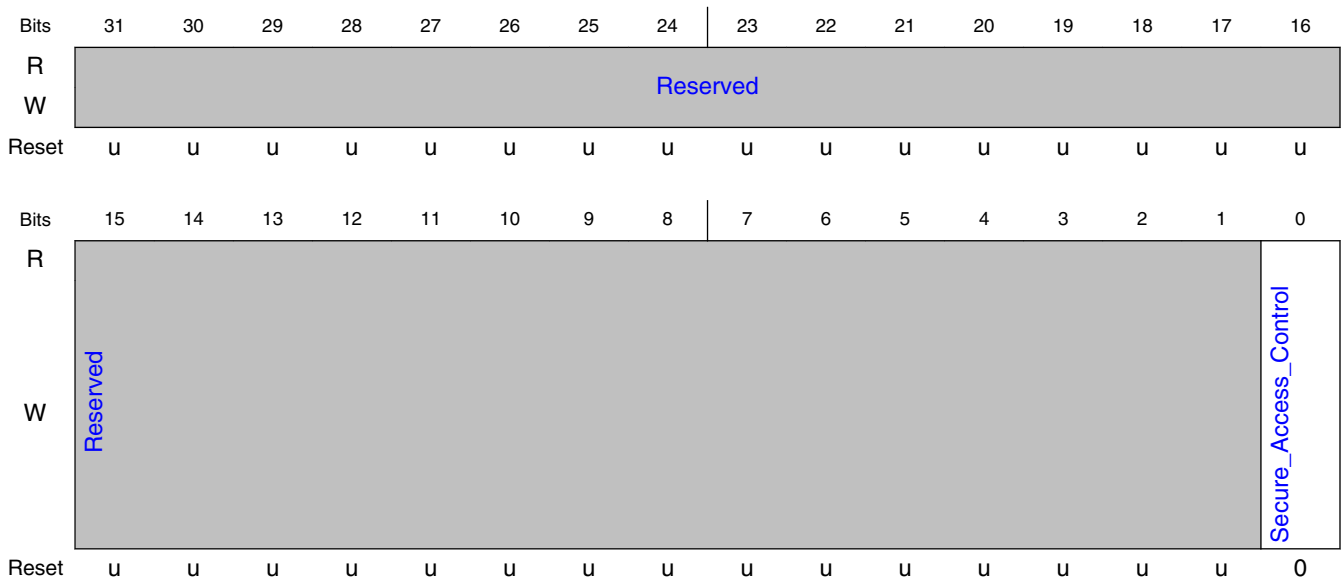
9.2.4.2 Function

The Secure Access register controls secure access. You can only write to this register using Secure transactions. Available in all CCI-400 configurations.

NOTE

This register enables Non-secure access to the CCI-400 registers for all masters. This compromises the security of your system.

9.2.4.3 Diagram



9.2.4.4 Fields

Field	Function
31-1 —	Reserved
0 Secure_Access_Control	Secure Access Control Non-secure register access override: 0b - Disable Non-secure access to CCI-400 registers. 1b - Enable Non-secure access to CCI-400 registers.

9.2.5 Status Register (Status_Register)

9.2.5.1 Offset

Register	Offset
Status_Register	Ch

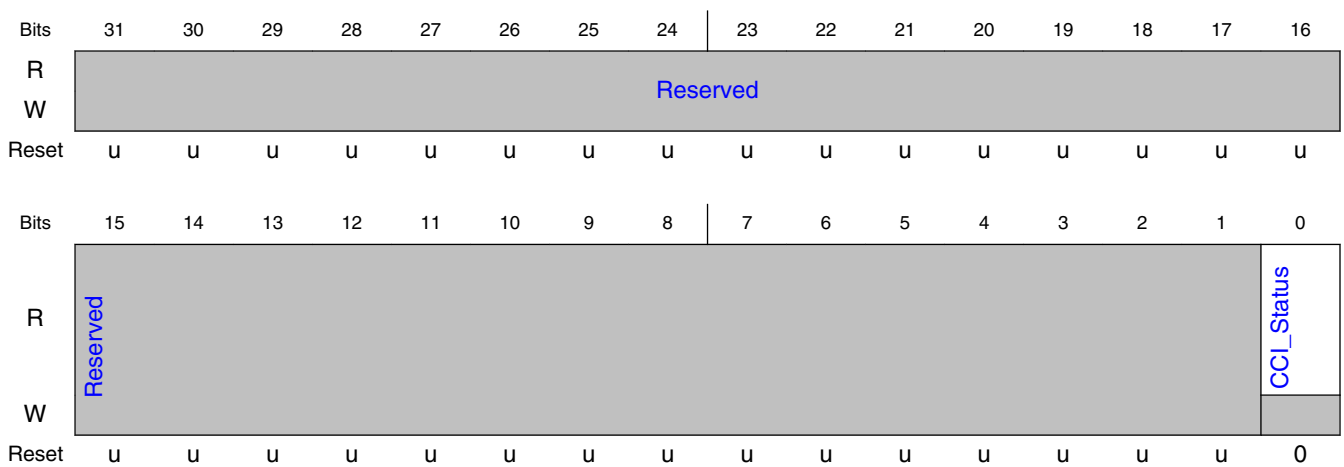
9.2.5.2 Function

The Status Register safely enables and disables snooping. When changing the snoop or DVM message enables using the Snoop Control Registers, see Snoop Control Registers, there is a delay until the changes are registered in all parts of the CCI-400. The change_pending bit in the Status Register indicates whether there are any changes to the enables that have not yet been applied, or whether a slave interface has been disabled for future snoop and DVM messages, but has outstanding AC requests. There are no usage constraints. Available in all CCI-400 configurations.

NOTE

After writing to the snoop or DVM enable bits, the controller must wait for the register write to complete, then test that the change_pending bit is LOW before it turns an attached device on or off.

9.2.5.3 Diagram



9.2.5.4 Fields

Field	Function
31-1 —	Reserved
0 CCI_Status	CCI_Status Indicates whether any changes to the snoop or DVM enables is pending in the CCI-400 0b - No change pending. 1b - Change pending.

9.2.6 Imprecise Error Register (Imprecise_Error_Register)

9.2.6.1 Offset

Register	Offset
Imprecise_Error_Register	10h

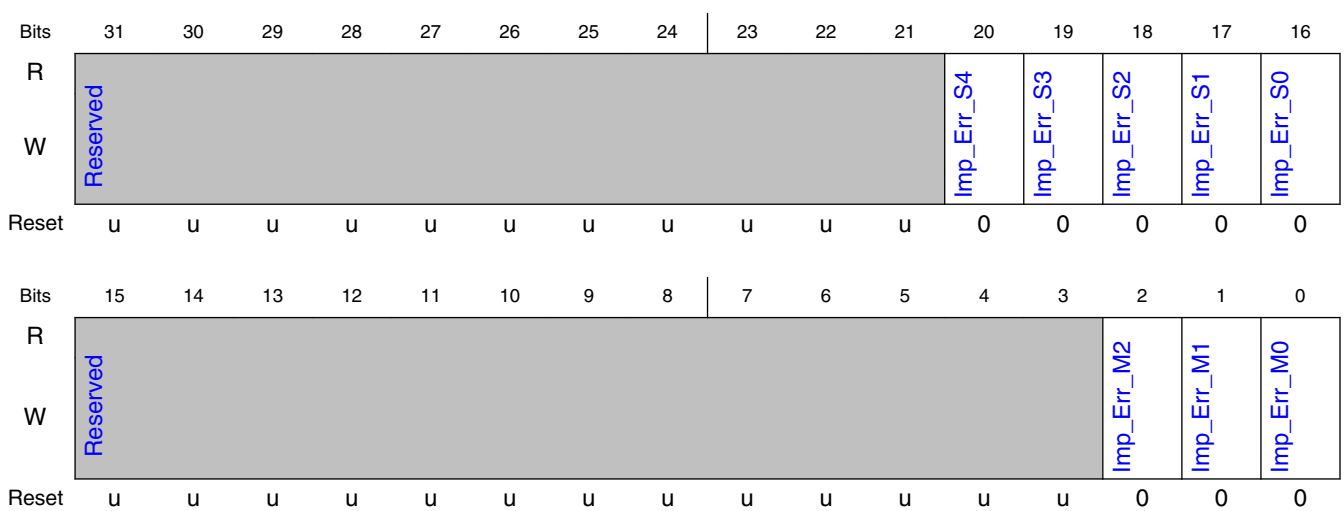
9.2.6.2 Function

The Imprecise Error register records the CCI-400 interfaces that received an error that is not signaled precisely. The appropriate bit is set, with respect to the interface on which the error was received. Bits are set when one or more error responses are detected, and they are reset on a write of 1 to the corresponding bit. Accessible using only Secure accesses, unless you set the Secure Access Register. See Secure Access Register bit assignments. There are no usage constraints. Available in all CCI-400 configurations.

NOTE

If any of the imprecise error indicator bits are set, the nERRORIRQ signal is asserted, active LOW.

9.2.6.3 Diagram



9.2.6.4 Fields

Field	Function
31-21 —	Reserved
20 Imp_Err_S4	Imprecise error indicator for slave interface S4
19 Imp_Err_S3	Imprecise error indicator for slave interface S3
18 Imp_Err_S2	Imprecise error indicator for slave interface S2
17 Imp_Err_S1	Imprecise error indicator for slave interface S1
16 Imp_Err_S0	Imprecise error indicator for slave interface S0
15-3 —	Reserved.
2 Imp_Err_M2	Imprecise error indicator for master interface M2
1 Imp_Err_M1	Imprecise error indicator for master interface M1
0 Imp_Err_M0	Imprecise error indicator for master interface M0 0b - No error from the time this bit was last reset. 1b - An error response has been received, but not signalled precisely.

9.2.7 Snoop Control Registers (Snoop_Control_Register_S0 - Snoop_Control_Register_S4)

9.2.7.1 Offset

For a = 0 to 4:

Register	Offset
Snoop_Control_Register_Sa	1000h + (a × 1000h)

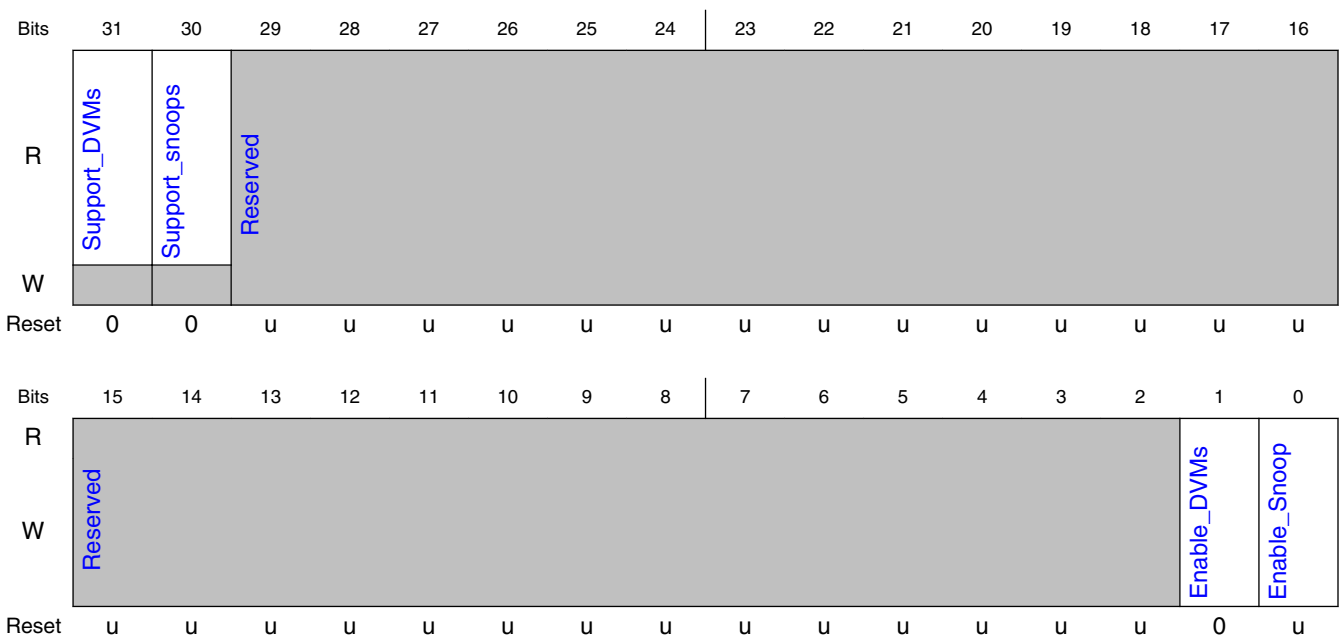
9.2.7.2 Function

The Snoop Control register controls the issuing of snoop and DVM requests on each slave interface. You can read the register to determine if the interface supports snoops or DVM messages. Enabling snoop or DVM requests on an interface that does not support them has no effect. One Snoop Control Register exists for each slave interface. Accessible using only Secure accesses, unless you set the Secure Access Register. See Secure Access Register bit assignments. Available in all CCI-400 configurations.

NOTE

- If the ACCHANNELEN input is LOW for this interface, write accesses to this register are ignored and snoop or DVM requests cannot be enabled.
- If snoops are disabled in the Control Override Register, write accesses to the snoop enable bit[0] are ignored.
- If DVM messages are disabled in the Control Override Register, write accesses to the DVM enable bit[1] are ignored.

9.2.7.3 Diagram



9.2.7.4 Fields

Field	Function
31 Support_DVMs	Slave interface supports DVM messages This is overridden to 0x0 if you set the Control Override Register [1]. See Control Override Register.
30 Support_snoops	Slave interface supports snoops This is overridden to 0x0 if you set the Control Override Register [0]. See Control Override Register.
29-2 —	Reserved
1 Enable_DVMs	Enable DVMs Enable issuing of DVM message requests from slave interface. RAZ/WI for interfaces not supporting DVM messages: 0b - Disable DVM message requests. 1b - Enable DVM message requests.
0 Enable_Snoop	Enable Snoop Enable issuing of snoop requests from this slave interface. RAZ/WI for interfaces not supporting snoops: **NOTE:** This bit is reserved for Snoop_Control_Register_S0, Snoop_Control_Register_S1, Snoop_Control_Register_S2, and Snoop_Control_Register_S3. 0b - Disable snoop requests. 1b - Enable snoop requests.

9.2.8 Shareable Override Registers (Shareable_Override_Register_S0 - Shareable_Override_Register_S4)

9.2.8.1 Offset

For a = 0 to 4:

Register	Offset
Shareable_Override_Register_Sa	1004h + (a × 1000h)

9.2.8.2 Function

The Shareable Override register overrides shareability of normal transactions through this interface. The following transaction types are unaffected by any override:

- FIXED-type bursts.

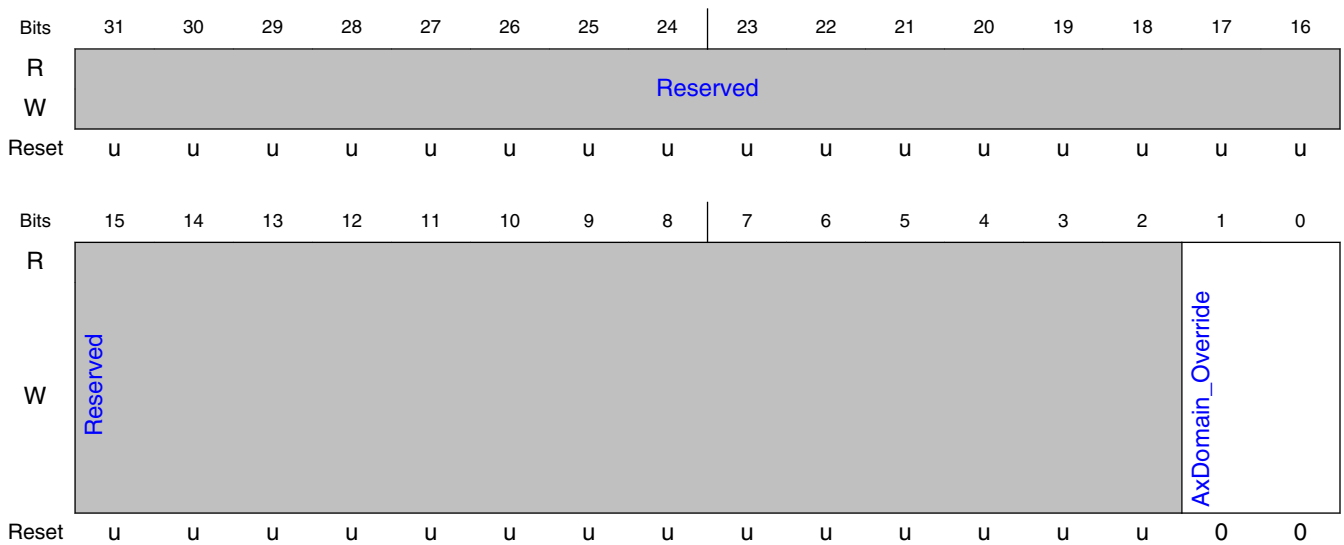
- Device transactions.
- Barrier.
- DVM message transactions.

Usage constraints This register is for ACE-Lite slave interfaces only. See the AMBA AXI and ACE Protocol Specification. Accessible using only Secure accesses, unless you set the Secure Access Register. See Secure Access Register bit assignments. Available in all CCI-400 configurations.

NOTE

Exclusive accesses must not be issued on an interface that is being overridden as shareable. If the CCI-400 is programmed to override transactions as shareable, Exclusive accesses are overridden to normal accesses. An exclusive write then receives an OKAY response to indicate that the slave does not support exclusive accesses.

9.2.8.3 Diagram



9.2.8.4 Fields

Field	Function
31-2 —	Reserved
1-0	AxDOMAIN override

Field	Function
AxDomain_Override	Shareable override for slave interface 00b - Do not override AxDOMAIN inputs. 01b - Do not override AxDOMAIN inputs. 10b - Override AxDOMAIN inputs to 0b00, all transactions are treated as non-shareable.ReadOnce becomes ReadNoSnoop.WriteUnique and WriteLineUnique become WriteNoSnoop. 11b - Override AxDOMAIN inputs to 0b01, normal transactions are treated as shareable.ReadNoSnoop becomes ReadOnce.WriteNoSnoop becomes WriteUnique.

9.2.9 Read Channel QoS Value Override Register (Read_Qos_Override_Register_S0 - Read_Qos_Override_Register_S4)

9.2.9.1 Offset

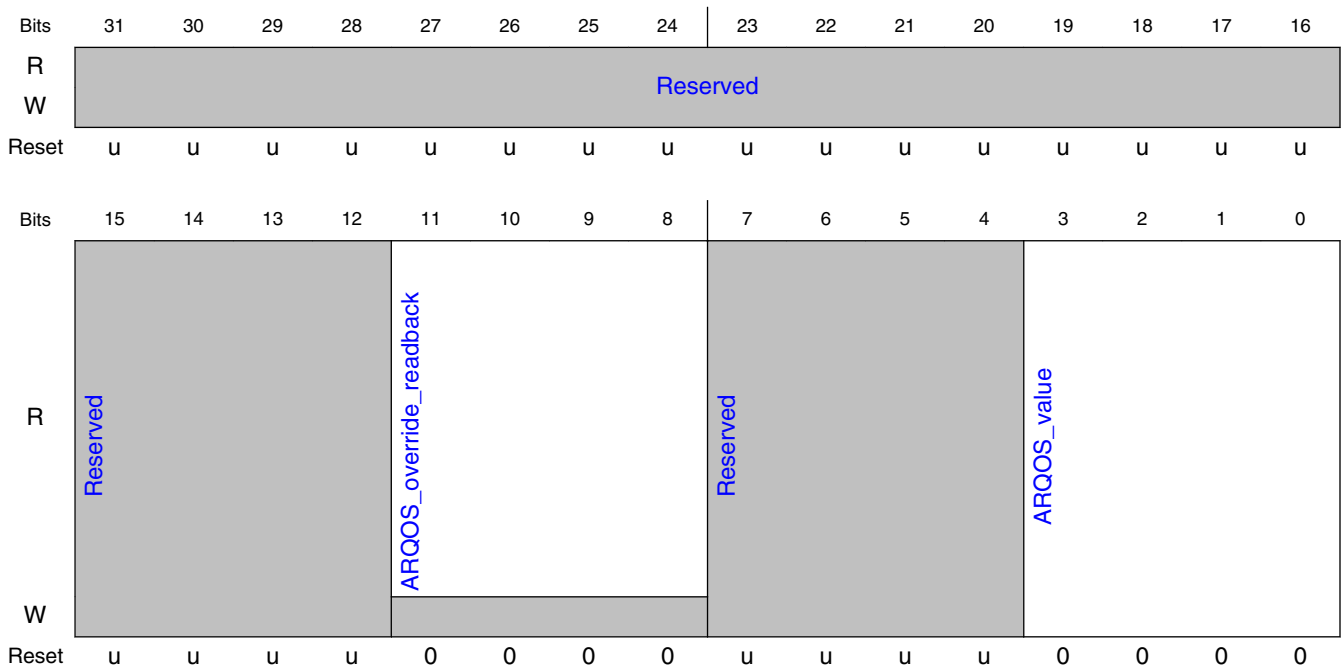
For a = 0 to 4:

Register	Offset
Read_Qos_Override_Register_Sa	1100h + (a × 1000h)

9.2.9.2 Function

The Read Channel QoS Value Override register contains override values for ARQOS, with a register for each slave interface. This value is used if you set the QOSOVERRIDE[4:0] input signal bit for this slave interface and the QoS value regulator is not enabled. You can also use this register to read the current value of the QoS value regulator for read accesses when the regulator is enabled. Accessible using only Secure accesses, unless you set the Secure Access Register. See Secure Access Register bit assignments. Available in all CCI-400 configurations.

9.2.9.3 Diagram



9.2.9.4 Fields

Field	Function
31-12 —	Reserved
11-8 ARQOS_override_readback	ARQOS override readback Reads what value is currently applied to transactions with ARQOS=0, provided QOSOVERRIDE is HIGH and the QoS value regulator is enabled.
7-4 —	Reserved
3-0 ARQOS_value	ARQOS value ARQOS value override for slave interface

9.2.10 Write Qos Override Register (Write_Qos_Override_Register_S0 - Write_Qos_Override_Register_S4)

9.2.10.1 Offset

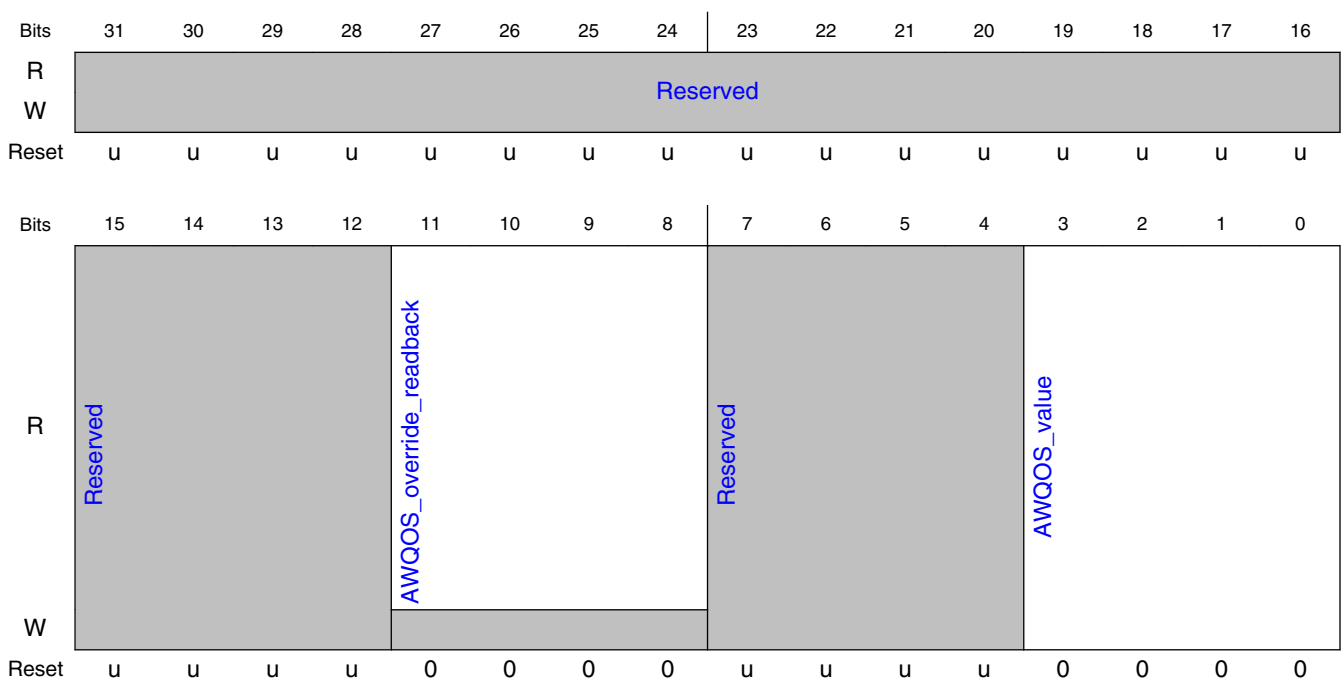
For a = 0 to 4:

Register	Offset
Write_Qos_Override_Register_Sa	1104h + (a × 1000h)

9.2.10.2 Function

The Write Channel QoS Value Override Register characteristics are: Purpose Contains override values for AWQOS, with a register for each slave interface. This value is used if you set the QOSOVERRIDE[4:0] input signal bit for this slave interface and the QoS value regulator is not enabled. You can also read the current value of the QoS value regulator for write accesses when the regulator is enabled. Accessible using only Secure accesses, unless you set the Secure Access Register. See Secure Access Register bit assignments. Available in all CCI-400 configurations.

9.2.10.3 Diagram



9.2.10.4 Fields

Field	Function
31-12 —	Reserved.
11-8 AWQOS_override_readback	AWQOS override readback Reads what value is currently applied to transactions with AWQOS=0, provided QOSOVERRIDE is HIGH and the QoS value regulator is enabled.
7-4 —	Reserved.
3-0 AWQOS_value	AWQOS value AWQOS value override for slave interface S0

9.2.11 Qos Control Register (Qos_Control_Register_S0 - Qos_Control_Register_S4)

9.2.11.1 Offset

For a = 0 to 4:

Register	Offset
Qos_Control_Register_Sa	110Ch + (a × 1000h)

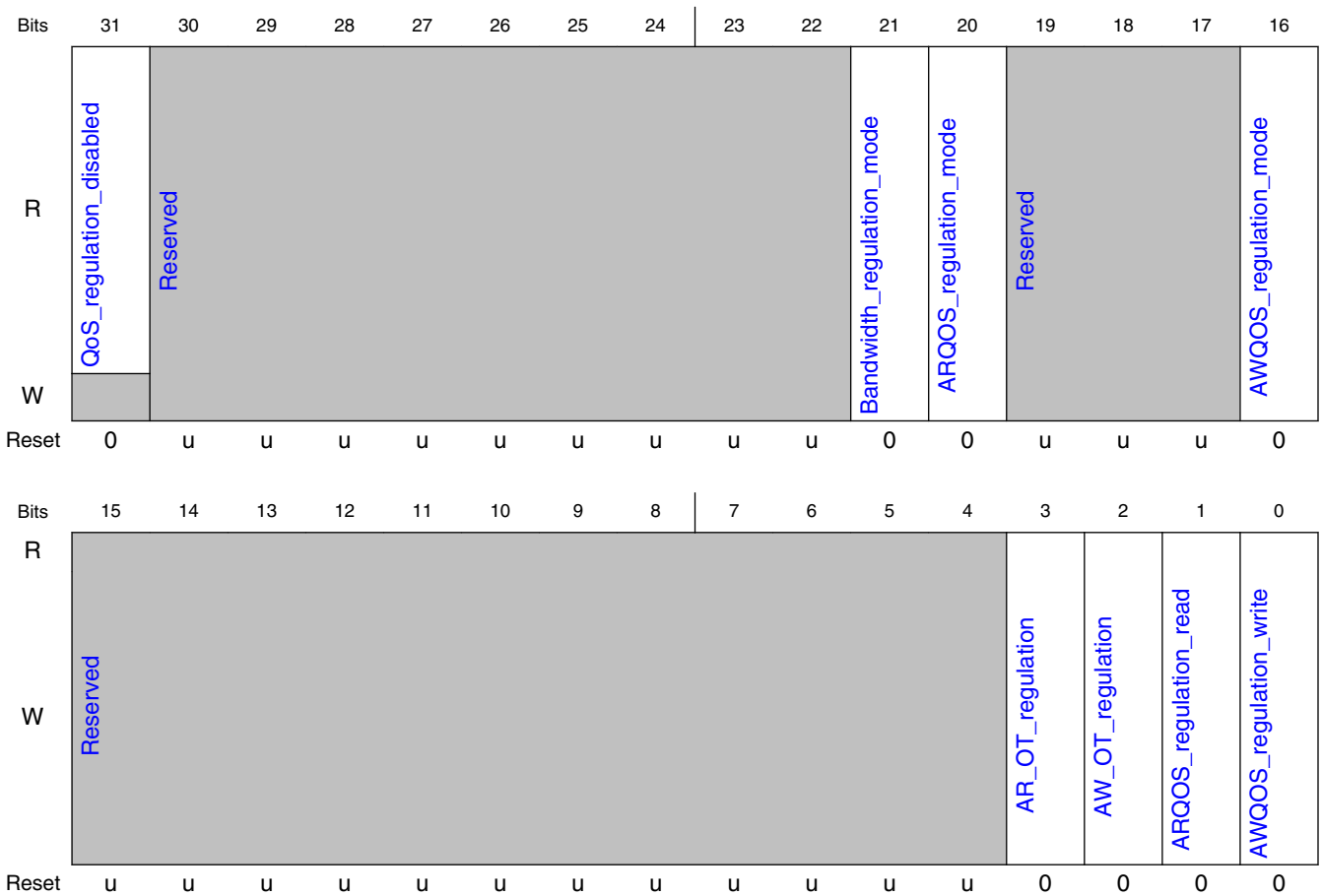
9.2.11.2 Function

The QoS Control register controls the regulators that are enabled on the slave interfaces. Accessible using only Secure accesses, unless you set the Secure Access Register. See Secure Access Register bit assignments on page 3-10. Available in all CCI-400 configurations.

NOTE

When outstanding transaction regulation is enabled or disabled for an interface, changes take effect only when there are no outstanding transactions in that interface.

9.2.11.3 Diagram



9.2.11.4 Fields

Field	Function
31 QoS_regulation_disabled	QoS regulation disabled Determines whether this CCI-400 implementation supports QoS regulation. 0b - QoS regulation fully supported as described in this document. See Quality of Service. 1b - QoS regulation not supported, reads and writes to this register have no effect.
30-22 —	Reserved
21 Bandwidth_regulation_mode	Bandwidth regulation mode Sets the mode for bandwidth regulation: 0b - Normal mode. The QoS value is stable when the master is idle. 1b - Quiesce High mode. The QoS value tends to the maximum when the master is idle.
20	ARQOS regulation mode Configures the mode of the QoS value regulator for read transactions:

Table continues on the next page...

Field	Function
ARQOS_regulation_mode	0b - Latency mode. 1b - Period mode, for bandwidth regulation.
19-17 —	Reserved
16 AWQOS_regulation_mode	AWQOS_regulation_mode Configures the mode of the QoS value regulator for write transactions: 0b - Latency mode. 1b - Period mode, for bandwidth regulation.
15-4 —	Reserved
3 AR_OT_regulation	AR_OT_regulation Enable regulation of outstanding read transactions for slave interfaces • ACE-Lite interfaces only, for example S0, S1, and S2. • RAZ/WI for ACE interfaces, for example S3 and S4.
2 AW_OT_regulation	AW_OT_regulation Enable regulation of outstanding write transactions for slave interfaces • ACE-Lite interfaces only, for example S0, S1, and S2. • RAZ/WI for ACE interfaces, for example S3 and S4.
1 ARQOS_regulation_read	ARQOS regulation read Enable QoS value regulation on reads for slave interfaces
0 AWQOS_regulation_write	AWQOS regulation write Enable QoS value regulation on writes for slave interfaces

9.2.12 Max OT Registers (Max_OT_Register_S0 - Max_OT_Register_S4)

9.2.12.1 Offset

For a = 0 to 4:

Register	Offset
Max_OT_Register_Sa	1110h + (a × 1000h)

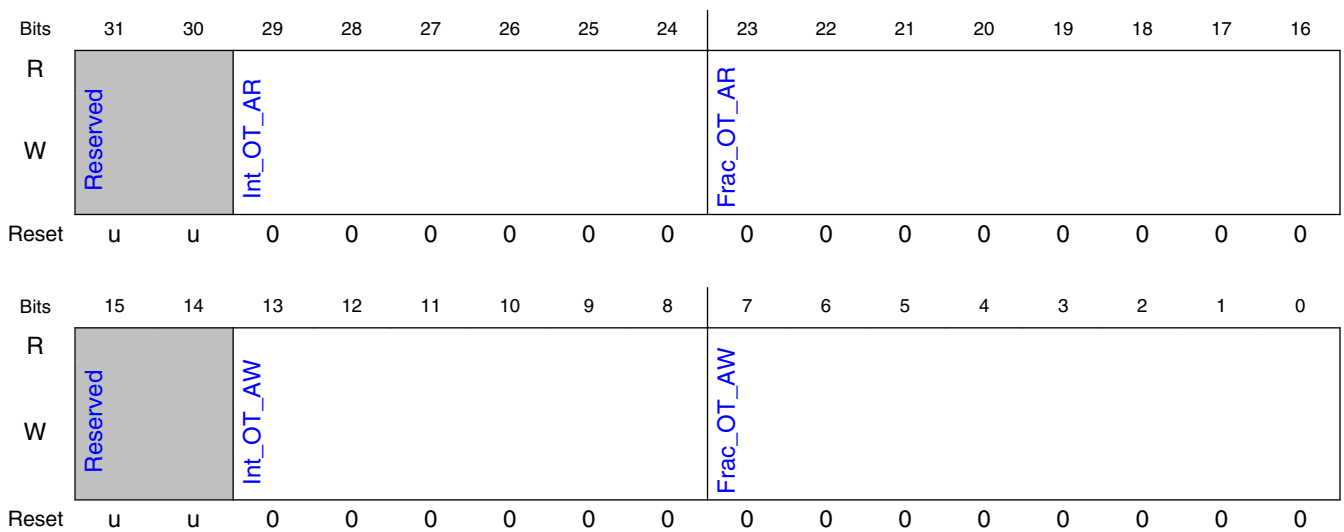
9.2.12.2 Function

The Max OT registers determine how many outstanding transactions are permitted when the OT regulator is enabled for each ACE-Lite slave interface. One register exists for each of the S0, S1, and S2 slave interfaces. A value of 0 for both the integer and fractional parts disables the programmable regulation so that the hardware limits apply. A value of 0 for the fractional part disables the regulation of fractional outstanding transactions. If int is the value of the integer part and frac is the value of the fractional part, then: Maximum mean number of outstanding transactions = int + frac/256.

Setting the maximum outstanding transaction size greater than that configured in the RTL, using the R_MAX or W_MAX parameters, has no effect. Accessible using only Secure accesses, unless you set the Secure Access Register. See Secure Access Register bit assignments.

Available in all CCI-400 configurations.

9.2.12.3 Diagram



9.2.12.4 Fields

Field	Function
31-30 —	Reserved
29-24	Int_OT_AR

Table continues on the next page...

Field	Function
Int_OT_AR	Integer part of the maximum outstanding AR addresses S0
23-16 Frac_OT_AR	Frac_OT_AR Fractional part of the maximum outstanding AR addresses S0
15-14 —	Reserved
13-8 Int_OT_AW	Int_OT_AW Integer part of the maximum outstanding AW addresses S0
7-0 Frac_OT_AW	Frac_OT_AW Fractional part of the maximum outstanding AW addresses S0

9.2.13 Regulator Target Registers (Target_Latency_Register_S0 - Target_Latency_Register_S4)

9.2.13.1 Offset

For a = 0 to 4:

Register	Offset
Target_Latency_Register_Sa	1130h + (a × 1000h)

9.2.13.2 Function

The Regulator Target registers determines the target, in cycles, for the regulation of reads and writes. The target is either transaction latency or inter-transaction period, depending on the programming of the QoS Control Register. A value of 0 corresponds to no regulation. One register exists for each slave interface.

Accessible using only Secure accesses, unless you set the Secure Access Register. See Secure Access Register bit assignments.

Only has an effect when QOSOVERRIDE is HIGH for the associated interface.

9.2.13.3 Diagram

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

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved				AW_Lat											
Reset	u	u	u	u	0	0	0	0	0	0	0	0	0	0	0	0

9.2.13.4 Fields

Field	Function
31-28 —	Reserved
27-16 AR_Lat	AR channel target latency
15-12 —	Reserved
11-0 AW_Lat	AW channel target latency

9.2.14 QoS Range Register (Qos_Range_Register_S0 - Qos_Range_Register_S4)

9.2.14.1 Offset

For a = 0 to 4:

Register	Offset
Qos_Range_Register_Sa	1138h + (a × 1000h)

9.2.14.2 Function

The QoS Range register enables you to program the minimum and maximum values for the ARQOS and AWQOS signals that the QV regulators generate. One register exists for each slave interface.

Accessible using only Secure accesses, unless you set the Secure Access Register. See Secure Access Register bit assignments.

Only has an effect when QOSOVERRIDE is HIGH for the associated interface.

9.2.14.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved				Max_ARQOS				Reserved				Min_ARQOS			
Reset	u	u	u	u	0	0	0	0	u	u	u	u	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved				Max_AWQOS				Reserved				Min_AWQOS			
Reset	u	u	u	u	0	0	0	0	u	u	u	u	0	0	0	0

9.2.14.4 Fields

Field	Function
31-28 —	Reserved
27-24 Max_ARQOS	Maximum ARQOS value
23-20 —	Reserved
19-16 Min_ARQOS	Minimum ARQOS value
15-12 —	Reserved
11-8 Max_AWQOS	Maximum AWQOS value
7-4	Reserved

Table continues on the next page...

Field	Function
—	
3-0 Min_AWQOS	Minimum AWQOS value

9.2.15 QoS Regulator Scale Factor Registers (Latency_Regulation_Register_S0 - Latency_Regulation_Register_S4)

9.2.15.1 Offset

For a = 0 to 4:

Register	Offset
Latency_Regulation_Register_Sa	2268h + (a × 1000h)

9.2.15.2 Function

The QoS Regulator Scale Factor Registers characteristics are: Purpose QoS regulation value, AWQOS or ARQOS, scale factor coded for powers of 2 in the range 2^{-5}-2^{-12}, to match a 16-bit integrator. One register exists for each slave interface.

Accessible using only Secure accesses, unless you set the Secure Access Register. See Secure Access Register bit assignments.

Only has an effect when QOSOVERRIDE is HIGH for the associated interface.

The table here shows the mapping of Scale Factor Register value to the scale factor.

Table 9-2. Mapping of Scale Factor Register value to Regulator scale factor

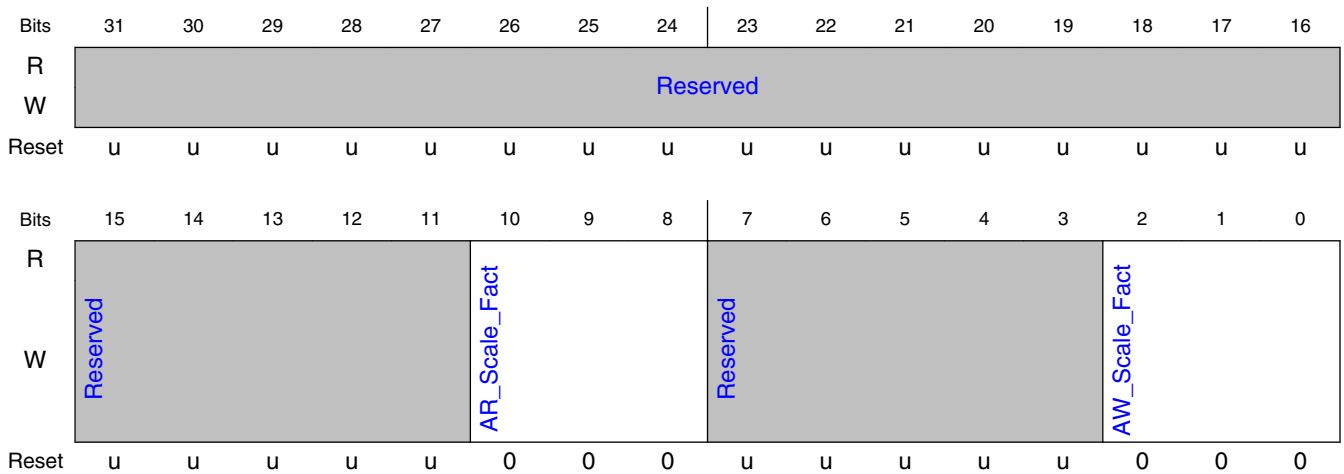
Scale Factor Register value	Scale factor
0x0	2^{-5}
0x1	2^{-6}
0x2	2^{-7}
0x3	2^{-8}
0x4	2^{-9}
0x5	2^{-10}
0x6	2^{-11}

Table continues on the next page...

Table 9-2. Mapping of Scale Factor Register value to Regulator scale factor (continued)

0x7	2^{-12}

9.2.15.3 Diagram



9.2.15.4 Fields

Field	Function
31-11 —	Reserved
10-8 AR_Scale_Fact	ARQOS Scale Factor ARQOS scale factor, power of 2 in the range 2^{-5}-2^{-12}.
7-3 —	Reserved
2-0 AW_Scale_Fact	AWQOS Scale Factor AWQOS scale factor, power of 2 in the range 2^{-5}-2^{-12}.

9.3 Functional Description

This chapter describes the functionality of the CoreLink CCI-400 Cache Coherent Interconnect.

9.3.1 About the functions

The CCI-400 combines interconnect and coherency functions into a single module.

9.3.2 Snoop connectivity and control

The CCI-400 has a fully-connected snoop topology, so if they are enabled:

- Each ACE slave interface snoops the other ACE slave interface.
- ACE-Lite slave interfaces snoop both ACE slave interfaces.
- DVM messages are broadcast through all enabled slave interfaces.

Snooping and DVM message broadcast are disabled at reset, so you must enable the appropriate masters for snooping and DVM messages using the Snoop Control Registers before shareable or DVM messages are sent to the CCI-400. See Snoop Control Registers (Snoop_Control_Register_S0 - Snoop_Control_Register_S4).

Each ACE slave interface has programmable bits in the Snoop Control Registers. These bits control the issuing of snoop and DVM message requests on that interface.

NOTE

ACE-Lite slave interfaces have programmable bits to enable DVM messages only.

These programmable bits of the Snoop Control Registers are tied LOW at reset so you must program them HIGH for each master in the shareable domain before shareable transactions or DVM messages are sent to the CCI-400. Before disabling a master, you must disable snoop and DVM messages to that master by programming the relevant Snoop Control Register bits LOW.

To avoid deadlock through having AC requests enabled to interfaces where masters are not present, or not able to process them, the CCI-400 has the following hardware and software override mechanisms:

- Each slave interface has an **ACCHANNELEN** input bit that, if you tie it LOW, prevents that interface from issuing any AC requests.

  NOTE

  These bits are only sampled at reset.

- There are bits in the Control Override Register to disable all snooping or all DVM message broadcast, irrespective of the programming of the Snoop Control Register.

9.3.2.1 Removing a master from the coherent domain

If you want to remove a master from the coherent domain, for example if a processor is being powered down, take the following actions:

1. Stop the processor from issuing shareable transactions. See the documentation of the processor.
2. Clean any shareable data in the processor cache. See the documentation of the processor.
3. Use the Snoop Control Register to prevent any more snoops or DVM messages being sent to the processor. See the Snoop Control Registers (Snoop_Control_Register_S0 - Snoop_Control_Register_S4).
4. Poll the CCI Status Register to confirm that the changes to the Snoop Control Register have completed. See the Status Register (Status_Register).

After you complete these actions, the master is no longer in the coherent domain and you can power it down or disable it. You must enable snoops to that master again before it allocates any shareable data in its cache.

9.3.3 Speculative fetch

For an application where the probability of a miss is high, then the snoop request and response time adds directly to the latency for each transaction labelled as shareable. To mitigate this, you can program each master interface to issue a fetch downstream in parallel with issuing a snoop request. This is known as a *speculative fetch*.

In the event that the snoop associated with a speculative fetch hits in a cache, then the data from the snoop is returned in preference to the data from the speculative fetch.

A speculative fetch is issued before all hazards that had arisen from the corresponding snoop have been resolved. Therefore, it is sometimes necessary to discard the data returned from memory and retry the fetch. These cases are:

- When a hazarding write transaction is detected. This hazarding write transaction must be ordered before the speculative fetch.
- When data from the speculative fetch returns before the snoop response for that transaction, and the read data buffer is already occupied by data waiting for a snoop response.

You can use the PMU to record the number of retry transactions for each master interface.

NOTE

Speculative fetches are only issued for these read-type transactions:

- ReadOnce .
- ReadClean .
- ReadNotSharedDirty .
- ReadUnique .
- ReadShared.

Although speculative fetches reduce the latency in the case of a snoop miss, there is a bandwidth and power penalty because of the additional transactions on a snoop hit. Therefore, you can disable speculative fetches where you expect a significant number of snoops to hit. You can use the Speculation Control Register to disable speculative fetches for a master or a slave interface. For example, you can disable speculative fetches for transactions from a master that is not latency sensitive. See Speculation Control Register (Speculation_Control_Register).

9.3.4 Security

If you are building a system based on the Secure and Non-secure capabilities that Arm TrustZone® technology provides, then you must consider security issues. This section describes:

- Internal programmers view.
- Security of master interfaces.

9.3.4.1 Internal programmers view

With the exception of the PMU registers, the programmers view defaults to Secure access only, as follows:

- Non-secure read requests to Secure registers receive a DECERR response, **RRESP[1:0]** == 0b11 , and zeroed data.
- Non-secure write requests to Secure registers receive a DECERR response, **BRESP[1:0]** == 0b11 and are *Write-Ignored* (WI).

NOTE

Some accesses might receive a response before they reach the CCI-400 registers and so do not receive a DECERR response nor affect the register values. An example of this is a cache maintenance operation that incorrectly addresses the CCI-400 register space.

You can override these security settings in the Secure Access Register. At reset, you can only access this using Secure requests, but if you write to it, this enables Non-secure access to all registers in the CCI-400 except for the Control Override Register and Secure Access Register. See Control Override Register (Control_Override_Register) and Secure Access Register (Secure_Access_Register).

9.3.4.2 Security of master interfaces

Transactions from the CCI-400 master interfaces always retain the security setting of the originating transactions. This applies to:

- Non-shareable transactions.
- Snoop misses.
- Speculative fetches.
- CCI-400-generated writes.

9.3.5 Error responses

The CCI-400 uses a mixture of precise and imprecise signaling of error responses, where:

- Precise errors are signalled back to the master on the R and B channels for the precise transaction that caused the error.
- Imprecise errors are not signalled on R and B channels but are instead signalled using the **nERRORIRQ** output pin (See Table "Interrupt Assignments" in the "Interrupt Assignments" chapter). You can identify the interface that received the error response by reading the Imprecise Error Register. See Imprecise Error Register (Imprecise_Error_Register).

9.3.5.1 Imprecise errors

Table 9-3 shows the errors that are signalled as imprecise. All other sources of error are signalled precisely.

NOTE

An error is signalled either precisely or imprecisely, but never both.

Table 9-3. Imprecise errors

Transaction causing error	Channel receiving error	Imprecise error indicator from
A ReadX snoop that misses in the cache and fetches data from downstream	CR	Slave interface receiving the CR response
Distributed Virtual Memory message	CR	Slave interface receiving the CR response
Speculative fetch that returns an error, but the snoop returns data	R	Master interface receiving the R response
Speculative fetch that must be retried	R	Master interface receiving the R response
Write that the CCI-400 generated	B	Master interface receiving the B response
Snoop error generated by a WriteLineUnique or WriteUnique transaction	CR	Master interface receiving the CR response
Write error for WriteUnique transactions that have been split but not the last transaction in the split sequence	B	Slave interface that received the transaction that was split

The CCI-400 generates a precise DECERR response in the case of a security violation on a CCI-400 register access. See Imprecise Error Register (Imprecise_Error_Register) and Security .

9.3.6 Barriers

The CCI-400 supports all types of AMBA 4 barrier transactions. Each slave interface broadcasts these to every master interface, ensuring that intermediate transaction source and sink points observe the barrier correctly.

NOTE

Only MMDC supports barrier transactions in the chip.

9.3.7 DVM messages

The ACE slave interfaces support DVM messages through their regular AC and CR channels. The ACE-Lite interfaces all contain AC and CR channels to support DVM messages only. Each slave interface has a programmable enable bit to determine whether it supports the issuing of AC requests for DVM messages. DVM messages are handled as regular transactions in the CCI-400, except that they are decoded based on the DVM message indicator, instead of the address, to ensure that multi-transaction DVM messages are correctly ordered.

The Snoop Control Registers and Control Override Register control the issuing of DVM message requests. See Snoop Control Registers (Snoop_Control_Register_S0 - Snoop_Control_Register_S4) and Control Override Register (Control_Override_Register).

ACE and ACE-Lite, plus DVM slave interfaces support all types of DVM transaction. These are:

- DVM Operation.
- DVM Synchronization.
- DVM Complete.

The R channel response to a DVM messages is sent immediately by the CCI-400. If the DVM message results in an error response, this is signaled imprecisely. For more information, see Error responses .

NOTE

- A master that issues DVM messages must also be able to receive DVM messages. The slave interface through which the master connects must have DVM messages enabled.
- No other bus master except A53 supprots DVM messaging in the chip.

9.3.8 Quality of Service

The CCI-400 supports QoS with the following independent mechanisms:

- QoS value
- Regulation based on outstanding transactions .

9.3.8.1 QoS value

Each CCI-400 slave interface has **ARQOS** and **AWQOS** input signals that transport a transaction-based QoS value. This determines the relative priority between transactions on that interface, or between interfaces. The CCI-400 uses the QoS value when it chooses between transaction requests at arbitration points and within queues. Transaction requests with the highest QoS value are prioritized. The CCI-400 uses a *Least Recently Granted* (LRG) scheme when two or more transactions share the highest value.

QoS values are propagated by CCI-400.

NOTE

Ensure that you balance the relative priorities of all slave interfaces. For example, setting each to the highest QoS value reduces the arbitration to LRG and no advantage is gained from having a QoS value.

You can override the **ARQOS** and **AWQOS** input signals from each slave interface by using a programmable register if the relevant static input signal, **QOSOVERRIDE[4:0]**, with one bit for each of slave interfaces 4-0, is HIGH. The QoS override is either based on a programmable value or uses performance feedback to set the value within a programmable range. Transactions that the CCI-400 generates use the same QoS value as the instigating transaction or the override value if **QOSOVERRIDE** is set.

NOTE

QOSOVERRIDE[4:0] input signal is not controllable in this device and set to zero.

NOTE

QOSOVERRIDE only applies to transactions that have a **ARQOS** or **AWQOS** value of 0. Therefore, each interface can have a mixture of traffic that is overridden or regulated and other traffic, with non-zero QoS value, that is unaffected. For example, high priority MMU page table requests might be mixed with high-bandwidth media requests that require regulation.

QoS value regulation

CCI-400 regulation mechanisms vary the transaction QoS value depending on latency or bandwidth achieved through that slave interface. You can program target latency or bandwidth and a QoS value range for each regulator. Arm recommends that achievable targets are set so that the regulator uses the minimum QoS value in most cases and only increases the QoS value, up to the programmed maximum, under worst case conditions. The maximum value for each regulator is 0 at reset, so you must program a maximum value before the regulator can be used.

You can control the rate of change of the regulator integrator by using a programmable scale factor. There are two types of QoS value regulation:

- Regulation based on latency.
- Regulation based on bandwidth.

Regulation based on latency

In this regulation mode, QoS values change based on measured latency. The value tends to increase if the latency is greater than the target and decrease if the latency is lower than the target.

Regulation based on bandwidth

For bandwidth regulation, the target used for feedback is the period between successive request handshakes, in cycles. The value tends to increase if the period is greater than the target and decrease if the period is lower than the target. If the average number of bytes per request is known, this is equivalent to a bandwidth measure. Shareable transactions in the CCI-400 are 64 bytes in size, so this is usually a good approximation to use.

There are two modes of operation available when you are using this type of regulation:

Normal mode

In this mode the QoS value remains stable when the master is idle, this is equivalent to measuring the average bandwidth only when the master is active. This is the default mode.

Quiesce High mode

In this mode, the QoS value tends to the maximum programmed value when the master is idle, so when it becomes active, the initial transactions have a high QoS value. This mode is suitable for latency sensitive masters because it allows the master to be serviced with high priority while the bandwidth requirement is below that set. If the master starts to exceed its programmed bandwidth, the priority is reduced. You can use this mechanism to ensure that other masters are not excluded when latency sensitive masters take significant bandwidth.

You enable QoS value regulation by setting the appropriate control bits. See Qos Control Register (Qos_Control_Register_S0 - Qos_Control_Register_S4). When you enable QoS value regulation, **ARQOS** and **AWQOS** values are driven by those generated by the regulators, if the original transaction has a zero QoS value and the **QOSOVERRIDE** configuration input is HIGH.

You can program the regulator mode using the QoS Control Registers.

NOTE

- Turning QoS value regulation on when **QOSOVERRIDE[x]** is set LOW for a specific interface has no effect.
- Transactions that do not transfer data are not counted for QoS value regulation and do not have their QoS value overridden. These transactions are:
  - CleanUnique .
  - MakeUnique .
  - CleanShared .
  - CleanInvalid .
  - MakeInvalid .
  - Evict .
  - Barriers.
  - DVM transactions.

9.3.8.2 Regulation based on outstanding transactions

The CCI-400 offers an additional mechanism for regulating traffic flows for the benefit of overall performance. Each ACE-Lite slave interface has an optional, programmable mechanism for limiting the number of outstanding read and write transactions, where an *Outstanding Transaction* (OT) is a read request without read data, or a write request without a response. This can be used where QoS value regulation is not effective because the system is not sensitive to QoS value. The disadvantage of this form of regulation is that it might stall the master even when the system is idle and traffic from this master is not affecting the performance of other masters.

You can characterize a sequence of transactions, with periods when there are no outstanding transactions, by using a fractional outstanding transaction number. For example, if requests occur every 100 cycles, but it only takes 50 cycles for the last response to arrive, then this corresponds to 0.5 OTs. The sum of the integer and fractional values represents a maximum of the mean number of OTs in a sliding window and, consequently, also over all time. If the fractional part is 0, the number of OTs is never permitted to exceed the integer part. If the fractional part is not 0, the number of OTs is not permitted to exceed one more than the integer part. This mean value is only achieved if the attached master maintains the maximum number of transactions it is able to issue at all times. Therefore, if the integer part is 0 and the fractional part is 0.5, and transactions have a lifespan of 50 cycles, then a master can issue a transaction. It finishes after 50 cycles and it cannot issue the next transaction until 100 cycles, maintaining a mean number of outstanding transactions as 0.5.

If you enable regulation, the following programmable values set the permitted number of outstanding transactions:

- OT integer.
- OT fraction.

NOTE

- Outstanding transaction regulation only counts transactions with a zero QoS value.
- Outstanding transaction regulation does not count or override transactions that have no data associated with them. These transactions are:
  - CleanUnique .
  - MakeUnique .
  - CleanShared .
  - CleanInvalid .
  - MakeInvalid .
  - Evict .
  - Barriers.
  - DVM transactions.

Read Queue Slot Reservation

Each master interface of the CCI-400 has a queue that stores read requests. If this becomes full with low priority requests, higher priority requests are blocked. To avoid this, the CCI-400 reserves a number of slots for high priority requests and a number of slots for high or medium priority requests. Table Regulation based on outstanding transactions shows a summary of the possible configurations.

Table 9-4. Slots reserved for high and medium priority traffic in each master interface

Read tracker size, as determined by the MIx_R_MAX parameter	Number of reserved slots for medium or high priority requests	Number of reserved slots for only high priority requests
2-4	0	0
5-7	0	1
8-15	1	1
16-128	3	1

For example, assume the read tracker size is 32, implying that three slots are reserved for medium to high priority requests and one slot is reserved for high priority requests. In this case:

- The maximum number of slots available for high priority requests is 32.
- The maximum number of slots available for medium priority requests is 32 - 1 = 31.
- The maximum number of slots available for low priority requests is 32 - 3 - 1 = 28.

The QoS values that are considered as high and medium priority can be configured at the design time using the R_THRESHOLD_UPPER and R_THRESHOLD_LOWER parameters however the values configured for these two parameters in this chip are 15 and 11.

9.3.8.3 QoS programmable registers

Programmers Model describes the following registers:

- Read Channel QoS Value Override Register.
- Write Channel QoS Value Override Register.
- QoS Control Register.
- Max OT Registers.
- Regulator Target Registers.
- QoS Regulator Scale Factor Registers.

Chapter 10
Arm CoreLink™ TrustZone Address Space Controller TZC-380

10.1 Introduction

This chapter introduces the CoreLink TrustZone Address Space Controller (TZC-380).

10.1.1 Overview

The TZASC is an advanced microcontroller bus architecture (AMBA) compliant system-on-chip (SoC) peripheral. It is a high-performance, area-optimized address space controller with on-chip AMBA bus interfaces that conform to the AMBA advanced eXtensible interface (AXI) protocol and the AMBA advanced peripheral bus (APB) protocol.

The TZASC can be configured to provide the optimum security address region control functions required for intended application. See Features for a summary of the configurable features supported.

NOTE

The software should configure the TZASC bypass mux (csu_sa1[2-3]) to disable the bypass operation and use the TZASC default region before any transaction goes to DDR.

The figure below shows the TZASC in an example system:

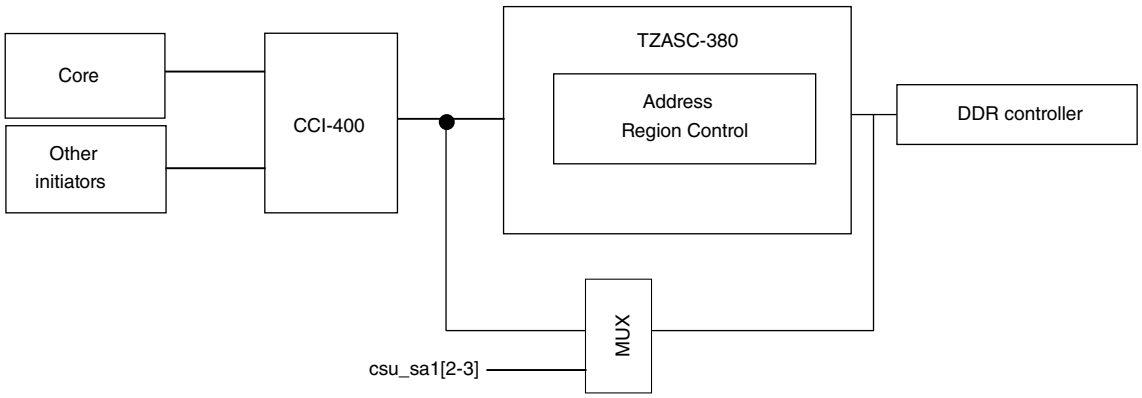


Figure 10-1. Example system

10.1.1.1 Features

The TZASC provides the following features:

- Programs security access permissions for each address region
- Transfers data between master and slave only if the security status of the AXI transaction matches the security settings of the memory region it addresses
- Prevents write access to various registers after assertion of secure_boot_lock, which is controlled by the CSU_SA1[4:5] bit. Refer CSU Memory Map/Register Definition for details.

The number of address regions can be configured to:

- 2 regions
- 4 regions
- 8 regions
- 16 regions.

10.2 Miscellaneous signal descriptions

There are two miscellaneous signals provided by TZASC:

- **secure_boot_lock**. The assertion of this signal is controlled by the CSU_SA1[4:5] bit. Refer CSU Memory Map/Register Definition for details.
- **tzasc_int**. The assertion of this signals is controlled by the TZASC interrupt (#125). Refer Internal interrupt sources for details.

The figure below shows the signals provided by the TZASC:

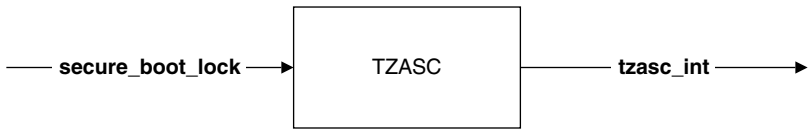


Figure 10-2. Miscellaneous signals

Asserting secure_boot_lock enhances the security of the TZASC. See Preventing writes to registers and using secure_boot_lock.

You can program the TZASC to assert tzasc_int when it denies an AXI master access to a region. See Denied AXI transactions.

10.3 Register Descriptions

The following information applies to the TZASC registers:

- The base address of the TZASC is not fixed, and can be different for any particular system implementation. The offset of each register from the base address is fixed.
- Do not attempt to access reserved or unused address locations. Attempting to access these location can result in Unpredictable behavior of the TZASC.
- Unless otherwise stated in the accompanying text:
  - do not modify undefined register bits
  - ignore undefined register bits on reads
  - all register bits are reset to a logic 0 by a system or power-on reset.
- For programming the registers, the TZASC supports data in word-invariant endianness.
- System designers must ensure that only processors in Secure state can access the registers, otherwise it can compromise the security of the system.

NOTE

See Constraints of use for more information about considerations relating to change in programmers view on an active system.

The TZASC register map consists of the following regions:

- Configuration, lockdown, and interrupt : Use these registers to determine the global configuration of the TZASC, and control its operating state.
- Fail status : These registers provide information about an access that failed because of insufficient permissions.
- Control : Use these registers to enable the TZASC to perform security inversion or speculative accesses.

- Region control : Use these registers to control the operating state of each region.
- Integration test : Use these registers when testing the integration of the TZASC in a System-on-Chip (SoC).
- Component configuration : These registers enable the identification of system components by software.

10.3.1 TZASC Memory map

TZASC base address: 150_0000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Configuration Register (configuration)	32	RO	0000_270Fh
4h	Action register (action)	32	RW	0000_0001h
8h	Lockdown Range Register (lockdown_range)	32	RW	0000_0000h
Ch	Lockdown Select Register (lockdown_select)	32	RW	0000_0000h
10h	Interrupt Status Register (int_status)	32	RO	0000_0000h
14h	Interrupt Clear Register (int_clear)	32	WO	0000_0000h
20h	Fail Address Low Register (fail_address_low)	32	RO	0000_0000h
24h	Fail Address High Register (fail_address_high)	32	RO	0000_0000h
28h	Fail Control Register (fail_control)	32	RO	0000_0000h
2Ch	Fail ID Register (fail_id)	32	RO	0000_0000h
30h	Speculation Control Register (speculation_control)	32	RW	0000_0000h
34h	Security Inversion Register (security_inversion_en)	32	RW	0000_0000h
100h	Region Setup Low 0 Register (region_setup_low_0)	32	RO	0000_0000h
104h	Region Setup High 0 Register (region_setup_high_0)	32	RO	0000_0000h
108h	Region Attributes 0 Register (region_attributes_0)	32	RW	C000_0000h
110h	Region Setup Low 1 Register (region_setup_low_1)	32	RO	0000_0000h
114h	Region Setup High 1 Register (region_setup_high_1)	32	RO	0000_0000h
118h	Region Attributes 1 Register (region_attributes_1)	32	RW	0000_001Ch
120h	Region Setup Low 2 Register (region_setup_low_2)	32	RO	0000_0000h
124h	Region Setup High 2 Register (region_setup_high_2)	32	RO	0000_0000h
128h	Region Attributes 2 Register (region_attributes_2)	32	RW	0000_001Ch
130h	Region Setup Low 3 Register (region_setup_low_3)	32	RO	0000_0000h
134h	Region Setup High 3 Register (region_setup_high_3)	32	RO	0000_0000h
138h	Region Attributes 3 Register (region_attributes_3)	32	RW	0000_001Ch
140h	Region Setup Low 4 Register (region_setup_low_4)	32	RO	0000_0000h
144h	Region Setup High 4 Register (region_setup_high_4)	32	RO	0000_0000h
148h	Region Attributes 4 Register (region_attributes_4)	32	RW	0000_001Ch
150h	Region Setup Low 5 Register (region_setup_low_5)	32	RO	0000_0000h

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
154h	Region Setup High 5 Register (region_setup_high_5)	32	RO	0000_0000h
158h	Region Attributes 5 Register (region_attributes_5)	32	RW	0000_001Ch
160h	Region Setup Low 6 Register (region_setup_low_6)	32	RO	0000_0000h
164h	Region Setup High 6 Register (region_setup_high_6)	32	RO	0000_0000h
168h	Region Attributes 6 Register (region_attributes_6)	32	RW	0000_001Ch
170h	Region Setup Low 7 Register (region_setup_low_7)	32	RO	0000_0000h
174h	Region Setup High 7 Register (region_setup_high_7)	32	RO	0000_0000h
178h	Region Attributes 7 Register (region_attributes_7)	32	RW	0000_001Ch
180h	Region Setup Low 8 Register (region_setup_low_8)	32	RO	0000_0000h
184h	Region Setup High 8 Register (region_setup_high_8)	32	RO	0000_0000h
188h	Region Attributes 8 Register (region_attributes_8)	32	RW	0000_001Ch
190h	Region Setup Low 9 Register (region_setup_low_9)	32	RO	0000_0000h
194h	Region Setup High 9 Register (region_setup_high_9)	32	RO	0000_0000h
198h	Region Attributes 9 Register (region_attributes_9)	32	RW	0000_001Ch
1A0h	Region Setup Low 10 Register (region_setup_low_10)	32	RO	0000_0000h
1A4h	Region Setup High 10 Register (region_setup_high_10)	32	RO	0000_0000h
1A8h	Region Attributes 10 Register (region_attributes_10)	32	RW	0000_001Ch
1B0h	Region Setup Low 11 Register (region_setup_low_11)	32	RO	0000_0000h
1B4h	Region Setup High 11 Register (region_setup_high_11)	32	RO	0000_0000h
1B8h	Region Attributes 11 Register (region_attributes_11)	32	RW	0000_001Ch
1C0h	Region Setup Low 12 Register (region_setup_low_12)	32	RO	0000_0000h
1C4h	Region Setup High 12 Register (region_setup_high_12)	32	RO	0000_0000h
1C8h	Region Attributes 12 Register (region_attributes_12)	32	RW	0000_001Ch
1D0h	Region Setup Low 13 Register (region_setup_low_13)	32	RO	0000_0000h
1D4h	Region Setup High 13 Register (region_setup_high_13)	32	RO	0000_0000h
1D8h	Region Attributes 13 Register (region_attributes_13)	32	RW	0000_001Ch
1E0h	Region Setup Low 14 Register (region_setup_low_14)	32	RO	0000_0000h
1E4h	Region Setup High 14 Register (region_setup_high_14)	32	RO	0000_0000h
1E8h	Region Attributes 14 Register (region_attributes_14)	32	RW	0000_001Ch
1F0h	Region Setup Low 15 Register (region_setup_low_15)	32	RO	0000_0000h
1F4h	Region Setup High 15 Register (region_setup_high_15)	32	RO	0000_0000h
1F8h	Region Attributes 15 Register (region_attributes_15)	32	RW	0000_001Ch
E00h	Integration Test Control Register (itcrg)	32	RW	0000_0000h
E04h	Integration Test Input Register (itip)	32	RO	0000_0000h
E08h	Integration Test Output Register (itop)	32	RW	0000_0000h

10.3.2 Configuration Register (configuration)

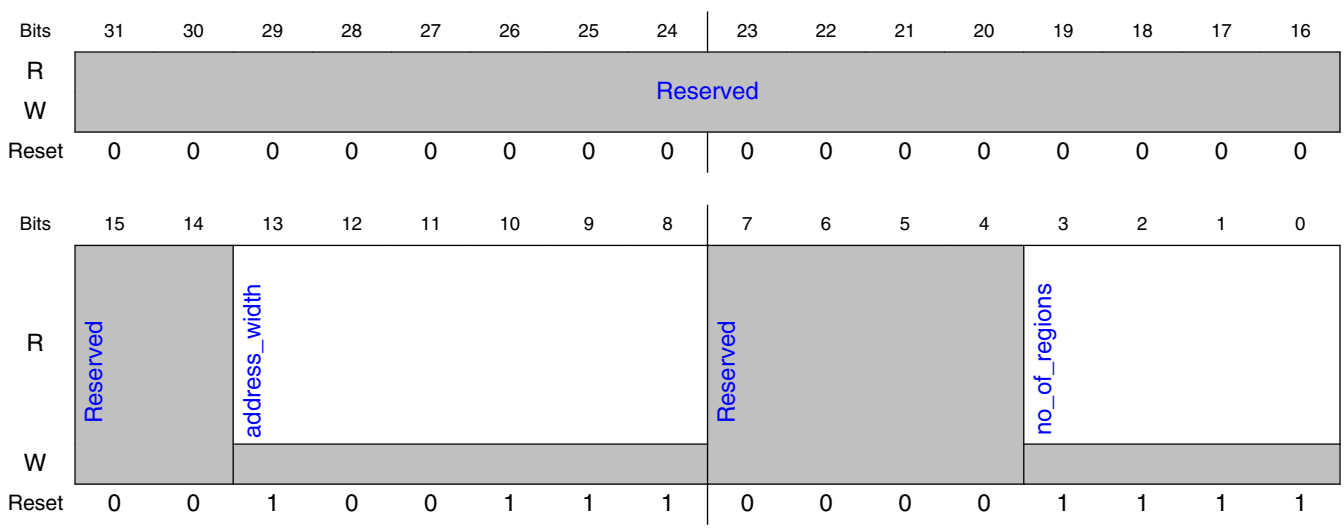
10.3.2.1 Offset

Register	Offset
configuration	0h

10.3.2.2 Function

The configuration Register provides information about the configuration of the TZASC. There are no usage constraints. Available in all configurations of the TZASC.

10.3.2.3 Diagram



10.3.2.4 Fields

Field	Function
31-14 —	Reserved, Should be Zero (SBZ).
13-8 address_width	address_width Returns the width of the AXI address bus. Read as:

Table continues on the next page...

Field	Function
	b000000-b011110 = reserved b011111 = 32-bit b100000 = 33-bit b100001 = 34-bit ... b111110 = 63-bit b111111 = 64-bit.
7-4 —	Reserved, Should be Zero (SBZ).
3-0 no_of_regions	no_of_regions Returns the number of regions that the TZASC provides: b0000 = reserved b0001 = 2 regions b0010 = 3 regions b0011 = 4 regions ... b1111 = 16 regions.

10.3.3 Action register (action)

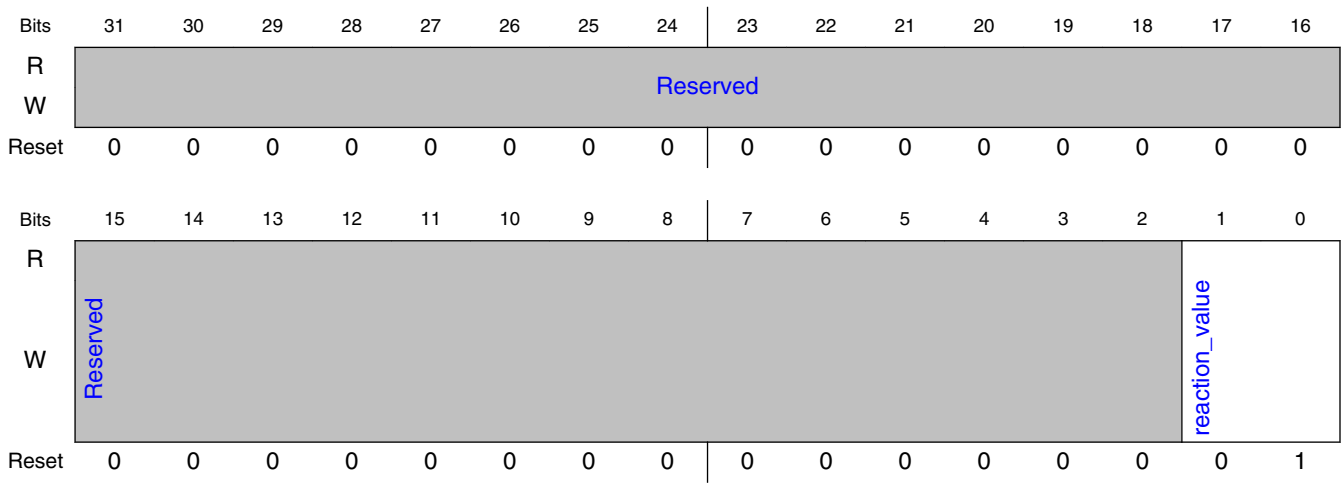
10.3.3.1 Offset

Register	Offset
action	4h

10.3.3.2 Function

The action Register controls the response signaling behavior of the TZASC to region permission failures. There are no usage constraints.Available in all configurations of the TZASC.

10.3.3.3 Diagram



10.3.3.4 Fields

Field	Function
31-2 —	Reserved, Should be Zero (SBZ).
1-0 reaction_value	reaction_value Controls how the TZASC uses the bresps[1:0], rresps[1:0], and tzasc_int signals when a region permission failure occurs: b00 = sets tzasc_int LOW and issues an OKAY response b01 = sets tzasc_int LOW and issues a DECERR response b10 = sets tzasc_int HIGH and issues an OKAY response b11 = sets tzasc_int HIGH and issues a DECERR response.

10.3.4 Lockdown Range Register (lockdown_range)

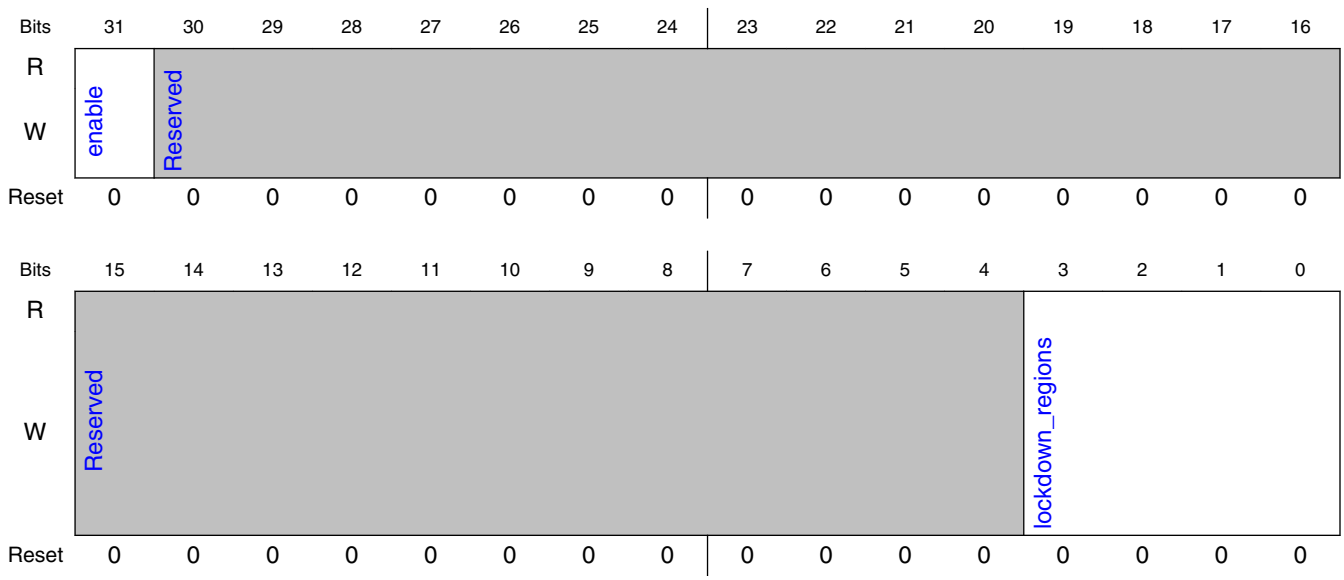
10.3.4.1 Offset

Register	Offset
lockdown_range	8h

10.3.4.2 Function

The lockdown_range Register controls the range of regions that are locked down.The lockdown_select Register can restrict the access type of this register to RO. Available in all configurations of the TZASC.

10.3.4.3 Diagram



10.3.4.4 Fields

Field	Function
31 enable	enable When set to 1, it enables the lockdown_regions field to control the regions that are to be locked.
30-4 —	Reserved, Should be Zero (SBZ).
3-0 lockdown_regions	lockdown_regions Controls the number of regions to lockdown when the enable bit is set to 1: b0000 = *region no_of_regions*–1 is locked b0001 = *region no_of_regions*–1 to region no_of_regions–2 are locked b0010 = *region no_of_regions*–1 to region no_of_regions–3 are locked b0011 = *region no_of_regions*–1 to region no_of_regions–4 are locked ... b1111 = region no_of_regions–1 to region no_of_regions–16 are locked.

Field	Function
	no_of_regions is the value of the no_of_regions field in the configuration Register. See Configuration Register. **NOTE:** The value programmed in lockdown_range Register must not be greater than no_of_regions-1 else all regions are locked.

10.3.5 Lockdown Select Register (lockdown_select)

10.3.5.1 Offset

Register	Offset
lockdown_select	Ch

10.3.5.2 Function

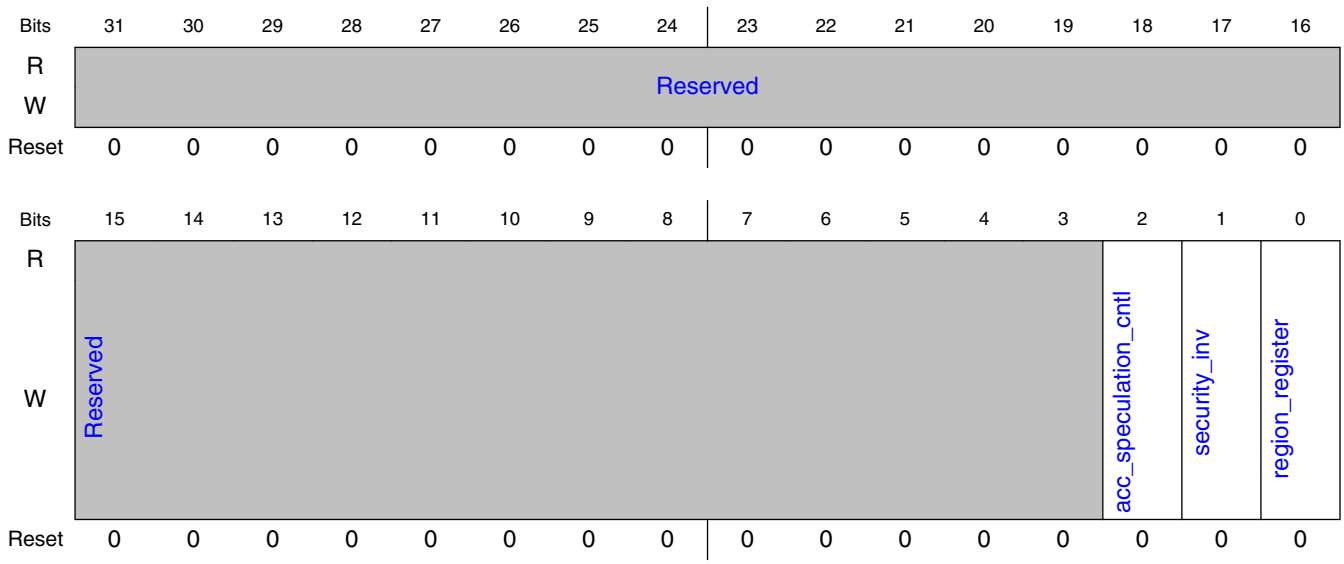
The lockdown_select Register controls whether the TZASC permits write accesses to the following registers:

- Lockdown Range Register
- Speculation Control Register
- Security Inversion Enable Register

After aresetn goes HIGH, the TZASC only permits write access to this register, if secure_boot_lock remains LOW. When secure_boot_lock is HIGH for one aclk period, or more then the TZASC ignores writes to this register.

This register is available in all configurations of the TZASC.

10.3.5.3 Diagram



10.3.5.4 Fields

Field	Function
31-3 —	Reserved, Should be Zero (SBZ).
2 acc_speculation_cntl	acc_speculation_cntl Modifies the access type of the speculation_control Register: 0 = no effect. speculation_control Register remains RW. 1 = speculation_control Register is RO. See Speculation Control Register for more information.
1 security_inv	security_inv Modifies the access type of the security_inversion_en Register: 0 = no effect. security_inversion_en Register remains RW. 1 = security_inversion_en Register is RO. See Security Inversion Enable Register for more information.
0 region_register	region_register Modifies the access type of the lockdown_range Register: 0 = no effect. lockdown_range Register remains RW. 1 = lockdown_range Register is RO. See Lockdown Range Register for more information.

10.3.6 Interrupt Status Register (int_status)

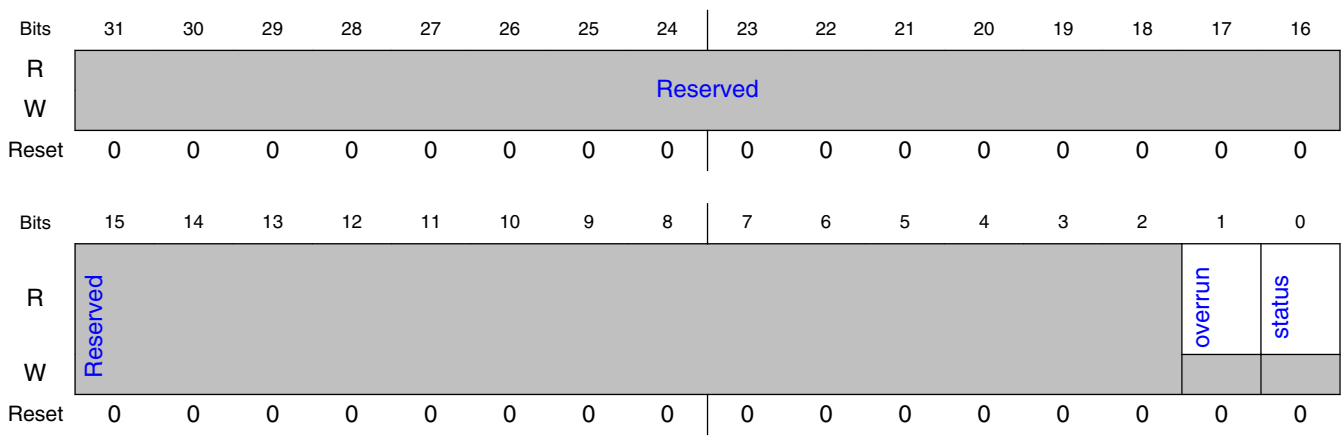
10.3.6.1 Offset

Register	Offset
int_status	10h

10.3.6.2 Function

The int_status register characteristics are: Returns the status of the interrupt. There are no usage constraints. Available in all configurations of the TZASC.

10.3.6.3 Diagram



10.3.6.4 Fields

Field	Function
31-2 —	Reserved, Should be Zero (SBZ).
1 overrun	overrun When set to 1, it indicates the occurrence of two or more region permission failures since the interrupt was last cleared

Table continues on the next page...

Field	Function
0 status	status Returns the status of the interrupt: 0 = interrupt is inactive 1 = interrupt is active.

10.3.7 Interrupt Clear Register (int_clear)

10.3.7.1 Offset

Register	Offset
int_clear	14h

10.3.7.2 Function

The int_clear Register clears the interrupt. There are no usage constraints. Available in all configurations of the TZASC.

10.3.7.3 Diagram

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

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

10.3.7.4 Fields

Field	Function
31-0 int_clear	int_clear Writing any value to the int_clear Register sets the: • status bit to 0 in the int_status Register • overrun bit to 0 in the int_status Register. See Interrupt Status register for details.

10.3.8 Fail Address Low Register (fail_address_low)

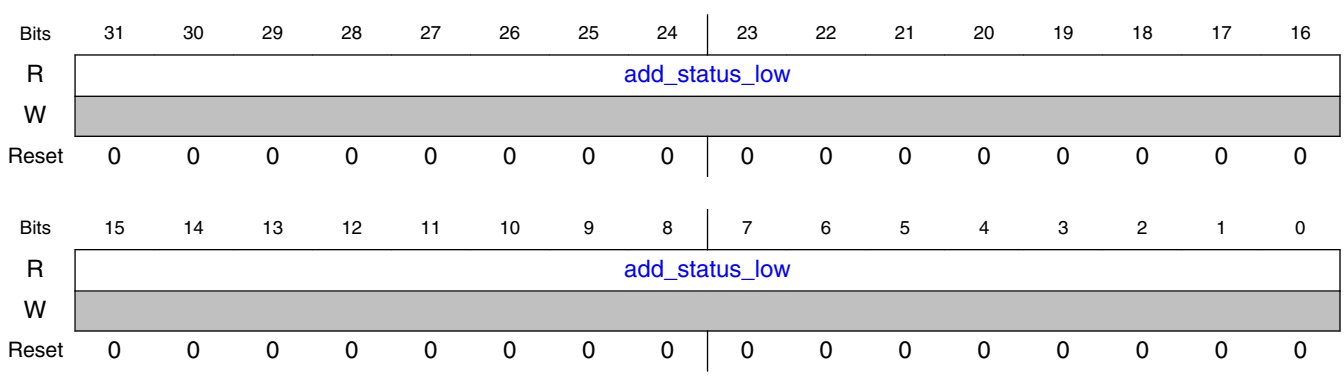
10.3.8.1 Offset

Register	Offset
fail_address_low	20h

10.3.8.2 Function

The fail_address_low Register returns the address, the lower 32-bits, of the first access that failed a region permission, after the interrupt was cleared. There are no usage constraints. Available in all configurations of the TZASC.

10.3.8.3 Diagram



10.3.8.4 Fields

Field	Function
31-0 add_status_low	add_status_low Returns the AXI address bits [31:0] of the first access to fail a region permission check after the interrupt was cleared.

10.3.9 Fail Address High Register (fail_address_high)

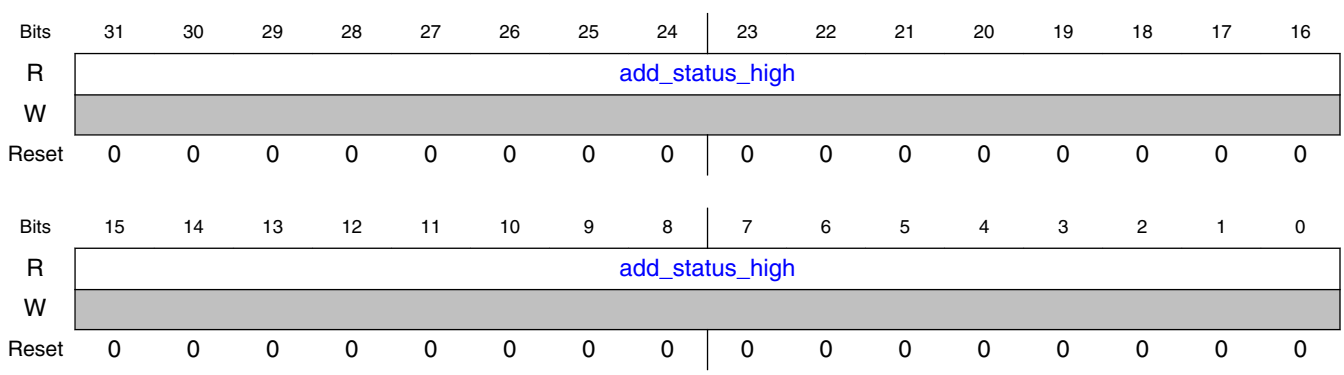
10.3.9.1 Offset

Register	Offset
fail_address_high	24h

10.3.9.2 Function

The fail_address_high Register returns the address, the upper 32-bits, of the first access that failed a region permission, after the interrupt was cleared. There are no usage constraints. Only available when the TZASC has an AXI address width of greater than 32 bits.

10.3.9.3 Diagram



10.3.9.4 Fields

Field	Function
31-0 add_status_high	add_status_high Returns the address bits [AXI_ADDRESS_MSB:32] of the first access to fail a region permission check after the interrupt was cleared. The size of this bitfield varies as [n:0} where n = AXI_ADDRESS_MSB–32..

10.3.10 Fail Control Register (fail_control)

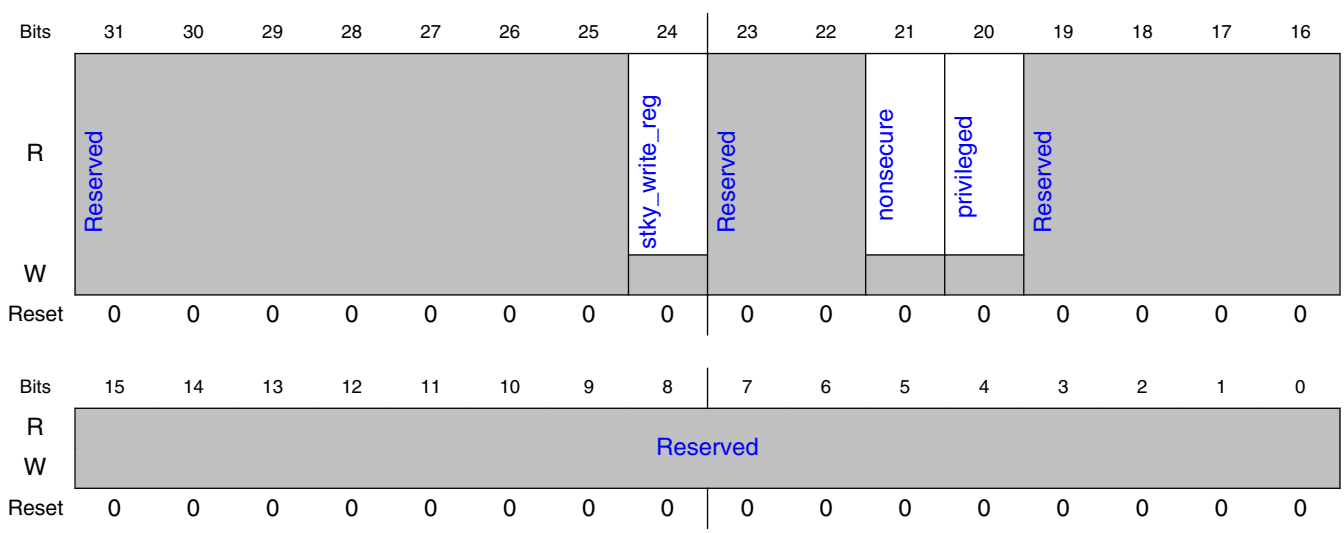
10.3.10.1 Offset

Register	Offset
fail_control	28h

10.3.10.2 Function

The fail_control Register returns the control status information of the first access that failed a region permission, after the interrupt was cleared. There are no usage constraints. Available in all configurations of the TZASC.

10.3.10.3 Diagram



10.3.10.4 Fields

Field	Function
31-25 —	Reserved, Should be Zero (SBZ).
24 stky_write_reg	stky_write_reg This bit indicates whether the first access to fail a region permission check was a write or read as: 0 = read access 1 = write access.
23-22 —	Reserved, Should be Zero (SBZ).
21 nonsecure	nonsecure After clearing the interrupt status, this bit indicates whether the first access to fail a region permission check was non-secure. Read as: 0 = secure access 1 = non-secure access.
20 privileged	privileged After clearing the interrupt status, this bit indicates whether the first access to fail a region permission check was privileged. Read as: 0 = unprivileged access 1 = privileged access.
19-0 —	Reserved, Should be Zero (SBZ).

10.3.11 Fail ID Register (fail_id)

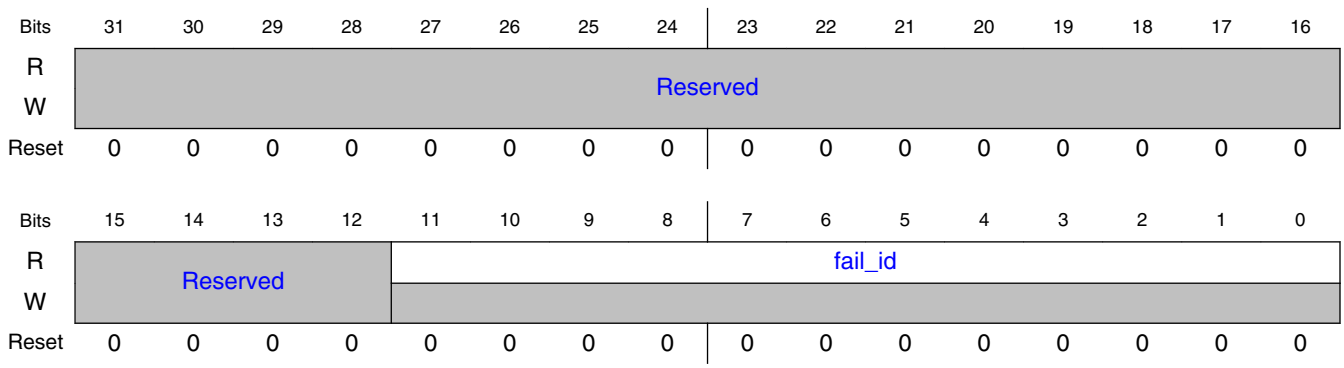
10.3.11.1 Offset

Register	Offset
fail_id	2Ch

10.3.11.2 Function

The fail_id Register returns the master AXI ID of the first access that failed a region permission, after the interrupt was cleared. There are no usage constraints. Available in all configurations of the TZASC.

10.3.11.3 Diagram



10.3.11.4 Fields

Field	Function
31-12 —	Reserved, Should be Zero (SBZ).
11-0 fail_id	fail_id Returns the master AXI ID of the first access to fail a region permission check after the interrupt was cleared. The size of this bit field is [n:0] where n = AID_WIDTH-1.

10.3.12 Speculation Control Register (speculation_control)

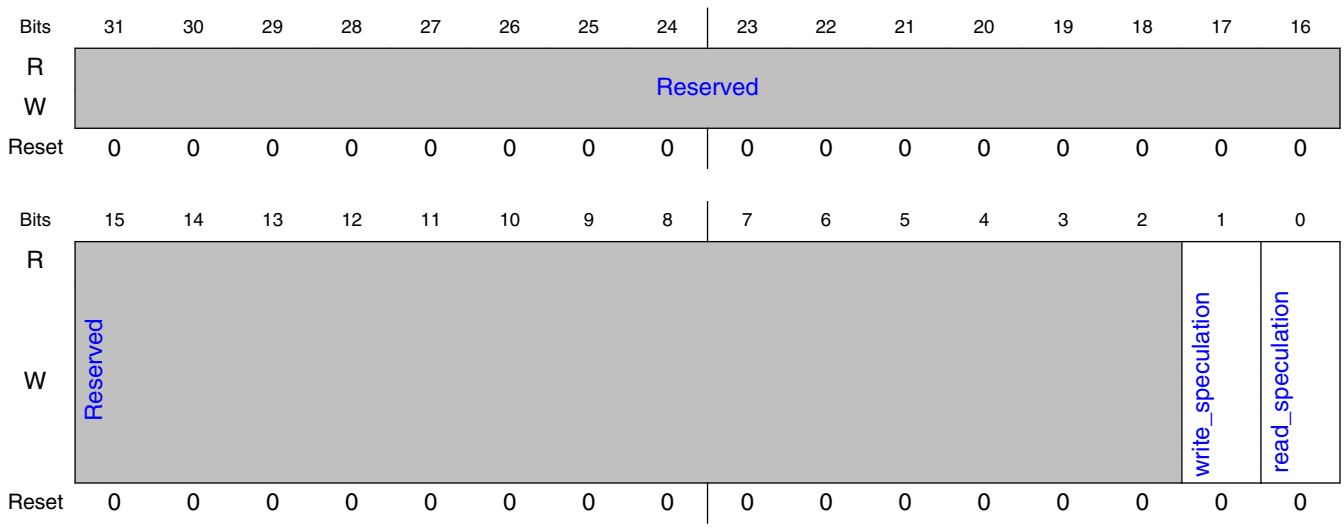
10.3.12.1 Offset

Register	Offset
speculation_control	30h

10.3.12.2 Function

The speculation_control Register controls the read access speculation and write access speculation. The lockdown_select Register can restrict the access type of this register to RO. See Lockdown Select Register for more details. Available in all configurations of the TZASC.

10.3.12.3 Diagram



10.3.12.4 Fields

Field	Function
31-2 —	Reserved, Should be Zero (SBZ).
1 write_speculation	write_speculation Controls the write access speculation: 0 = write access speculation is enabled. This is the default. 1 = write access speculation is disabled.
0 read_speculation	read_speculation Controls the read access speculation: 0 = read access speculation is enabled. This is the default. 1 = read access speculation is disabled.

10.3.13 Security Inversion Register (security_inversion_en)

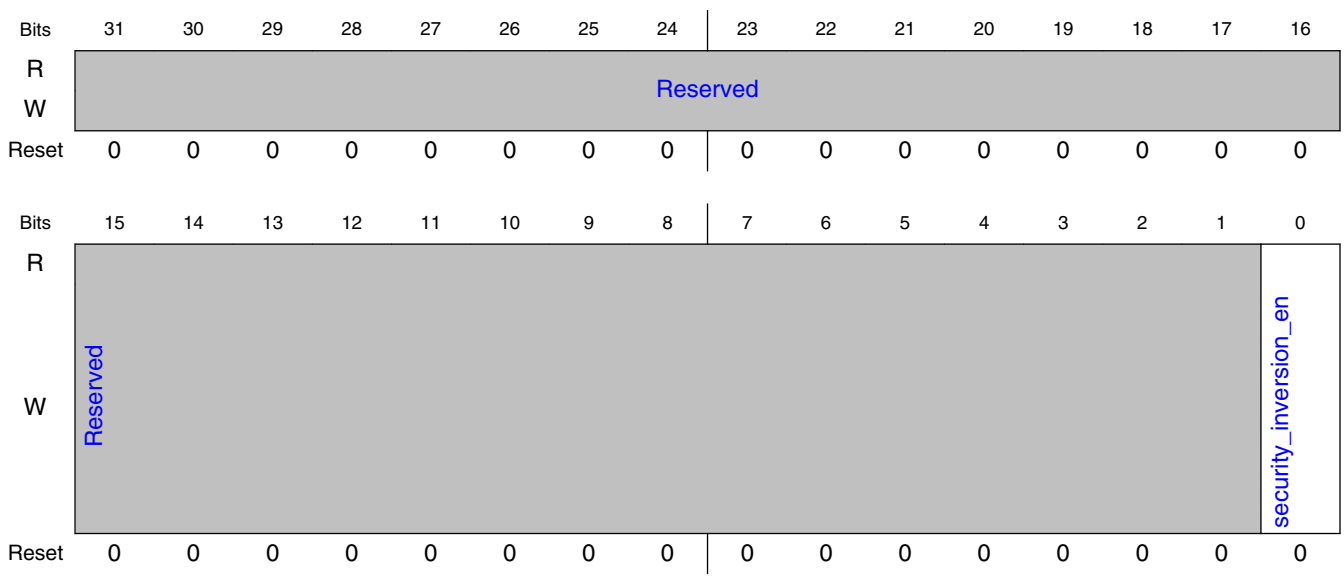
10.3.13.1 Offset

Register	Offset
security_inversion_en	34h

10.3.13.2 Function

The security_inversion_en Register controls whether the TZASC enables security inversion to occur. Usage constraints The lockdown_select Register can restrict the access type of this register to RO. See Lockdown Select Register for more details. Available in all configurations of the TZASC.

10.3.13.3 Diagram



10.3.13.4 Fields

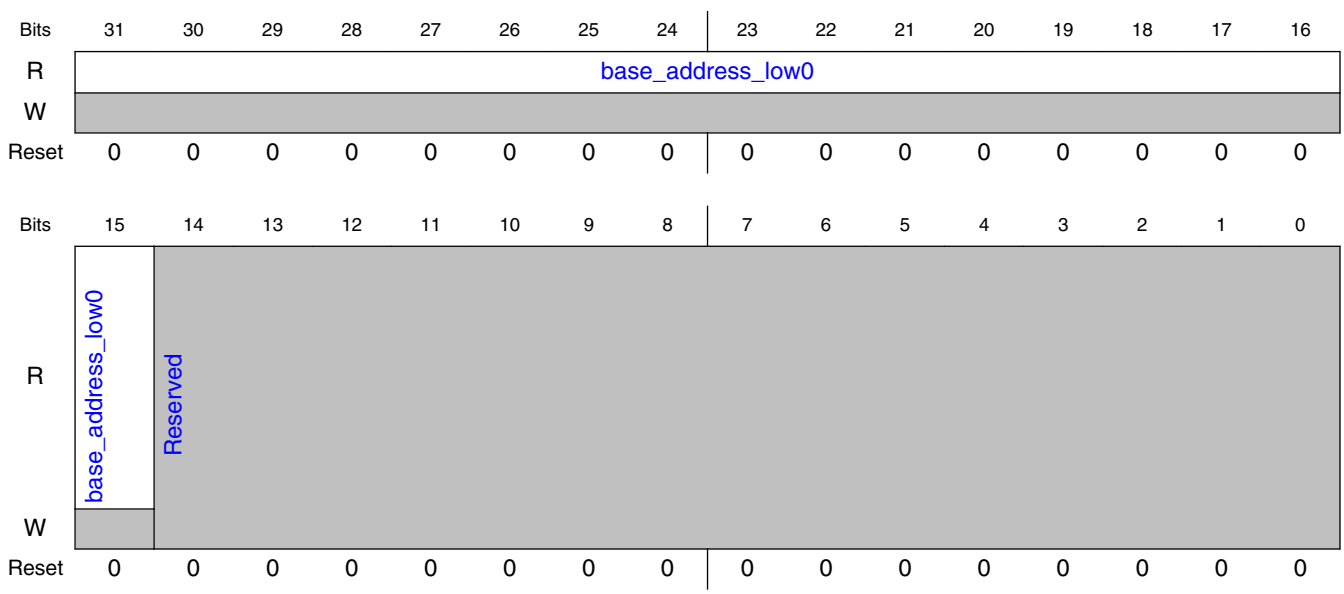
Field	Function
31-1 —	Reserved, Should be Zero (SBZ).
0 security_inversion_en	security_inversion_en Controls whether the TZASC permits security inversion to occur: 0 = security inversion is not permitted. This is the default. 1 = security inversion is permitted. This enables a region to be accessible to masters in Non-secure state but not accessible to masters in Secure state.

10.3.14 Region Setup Low 0 Register (region_setup_low_0)

10.3.14.1 Offset

Register	Offset
region_setup_low_0	100h

10.3.14.2 Diagram



10.3.14.3 Fields

Field	Function
31-15 base_address_low0	base_address_low0 Controls the base address [31:15] of region n. For region 0, this field is Read Only (RO). The TZASC sets the base address of region 0 to 0x0. The TZASC only permits a region to start at address 0x0, or at a multiple of its region size. For example, if the size of a region is 512MB, and it is not at address 0x0, the only valid settings for this field are: b00100000000000000 b01000000000000000 b01100000000000000 b10000000000000000 b10100000000000000 b11000000000000000 b11100000000000000 If you attempt to set an inappropriate base address for the size of the region, the TZASC ignores certain bits depending on the region size. See Table on register size for more information.
14-0 —	Reserved, Should be Zero (SBZ).

10.3.15 Region Setup High 0 Register (region_setup_high_0)

10.3.15.1 Offset

Register	Offset
region_setup_high_0	104h

10.3.15.2 Function

The region_setup_high register is none for region 0, for other regions it controls the base address [63:32] of region n.There are no usage constraints. Available in all configurations of the TZASC.

10.3.15.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	base_address_high0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	base_address_high0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

10.3.15.4 Fields

Field	Function
31-0 base_address_high0	base_address_high0 Controls the base address [63:32] of region n. For region 0, this field is RO. The TZASC sets the base address of region 0 to 0x0. The TZASC only permits a region to start at address 0x0, or at a multiple of its region size. If you program a region size to be 8GB or more, then the TZASC might ignore certain bits depending on the region size.

10.3.16 Region Attributes 0 Register (region_attributes_0)

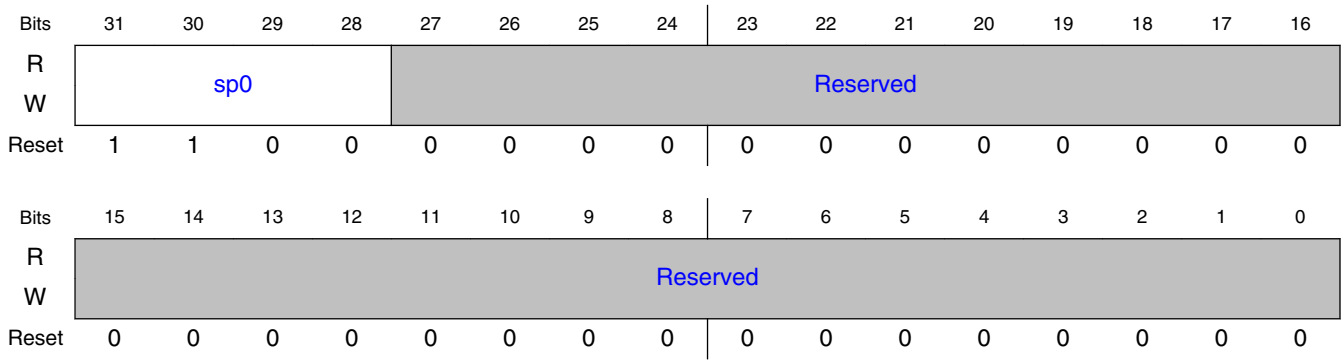
10.3.16.1 Offset

Register	Offset
region_attributes_0	108h

10.3.16.2 Function

The region_attributes_0 register controls the permissions for region 0. There are no usage constraints. Available in all configurations of the TZASC.

10.3.16.3 Diagram



10.3.16.4 Fields

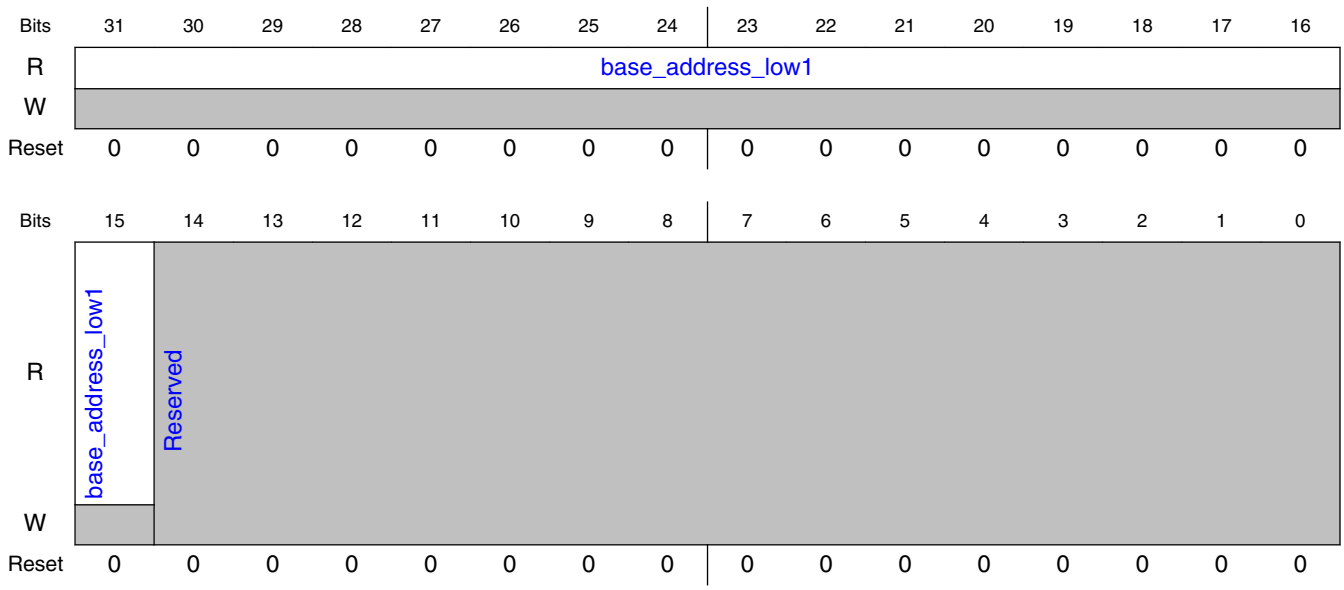
Field	Function
31-28 sp0	sp0 Permissions setting for region *n*. If an AXI transaction occurs to region *n*, the value in the sp*n* field controls whether the TZASC permits the transaction to proceed.
27-0 —	Reserved, Should be Zero (SBZ).

10.3.17 Region Setup Low 1 Register (region_setup_low_1)

10.3.17.1 Offset

Register	Offset
region_setup_low_1	110h

10.3.17.2 Diagram



10.3.17.3 Fields

Field	Function
31-15 base_address_low1	base_address_low1 Controls the base address [31:15] of region n. For region 0, this field is Read Only (RO). The TZASC sets the base address of region 0 to 0x0. The TZASC only permits a region to start at address 0x0, or at a multiple of its region size. For example, if the size of a region is 512MB, and it is not at address 0x0, the only valid settings for this field are: b00100000000000000 b01000000000000000 b01100000000000000 b10000000000000000 b10100000000000000 b11000000000000000 b11100000000000000 If you attempt to set an inappropriate base address for the size of the region, the TZASC ignores certain bits depending on the region size. See Table on register size for more information.
14-0 —	Reserved, Should be Zero (SBZ).

10.3.18 Region Setup High 1 Register (region_setup_high_1)

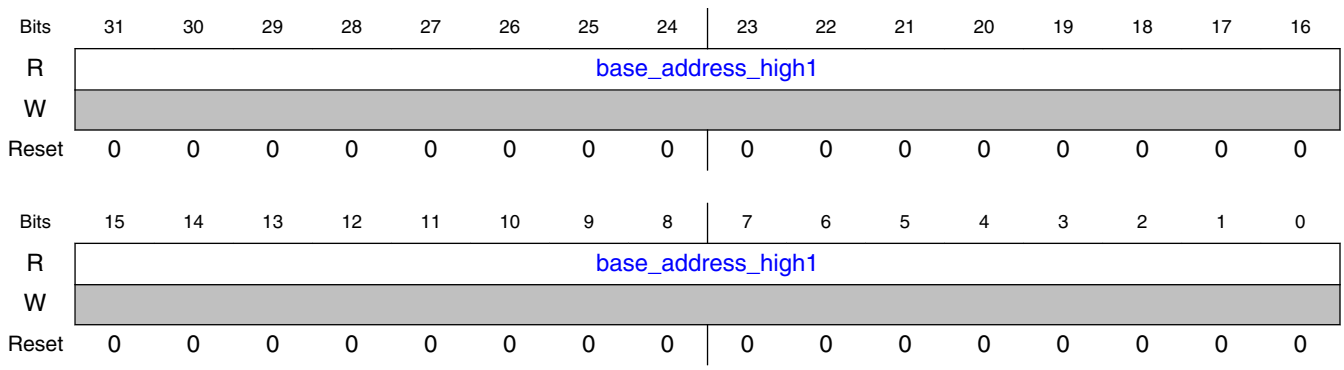
10.3.18.1 Offset

Register	Offset
region_setup_high_1	114h

10.3.18.2 Function

The region_setup_high register is none for region 0, for other regions it controls the base address [63:32] of region n.There are no usage constraints. Available in all configurations of the TZASC.

10.3.18.3 Diagram



10.3.18.4 Fields

Field	Function
31-0 base_address_high1	base_address_high1 Controls the base address [63:32] of region n. For region 0, this field is RO. The TZASC sets the base address of region 0 to 0x0. The TZASC only permits a region to start at address 0x0, or at a multiple of its region size. If you program a region size to be 8GB or more, then the TZASC might ignore certain bits depending on the region size.

10.3.19 Region Attributes 1 Register (region_attributes_1)

10.3.19.1 Offset

Register	Offset
region_attributes_1	118h

10.3.19.2 Function

The region_attributes_n register controls the permissions, region size, subregion disable, and region enable. There are no usage constraints. Available in all configurations of the TZASC.

Table 10-1. Region size

size<n> field	Size of region <n>	Base address [1] constraints
b000000-b001101	Reserved	-
b001110	32KB	-
b001111	64KB	Bit [15] must be zero
b010000	128KB	Bits [16:15] must be zero
b010001	256KB	Bits [17:15] must be zero
b010010	512KB	Bits [18:15] must be zero
b010011	1MB	Bits [19:15] must be zero
b010100	2MB	Bits [20:15] must be zero
b010101	4MB	Bits [21:15] must be zero
b010110	8MB	Bits [22:15] must be zero
b010111	16MB	Bits [23:15] must be zero
b011000	32MB	Bits [24:15] must be zero
b011001	64MB	Bits [25:15] must be zero
b011010	128MB	Bits [26:15] must be zero
b011011	256MB	Bits [27:15] must be zero
b011100	512MB	Bits [28:15] must be zero
b011101	1GB	Bits [29:15] must be zero
b011110	2GB	Bits [30:15] must be zero
b011111	4GB	Bits [31:15] must be zero
b100000	8GB	Bits [32:15] must be zero
b100001	16GB	Bits [33:15] must be zero
b100010	32GB	Bits [34:15] must be zero
b100011	64GB	Bits [35:15] must be zero

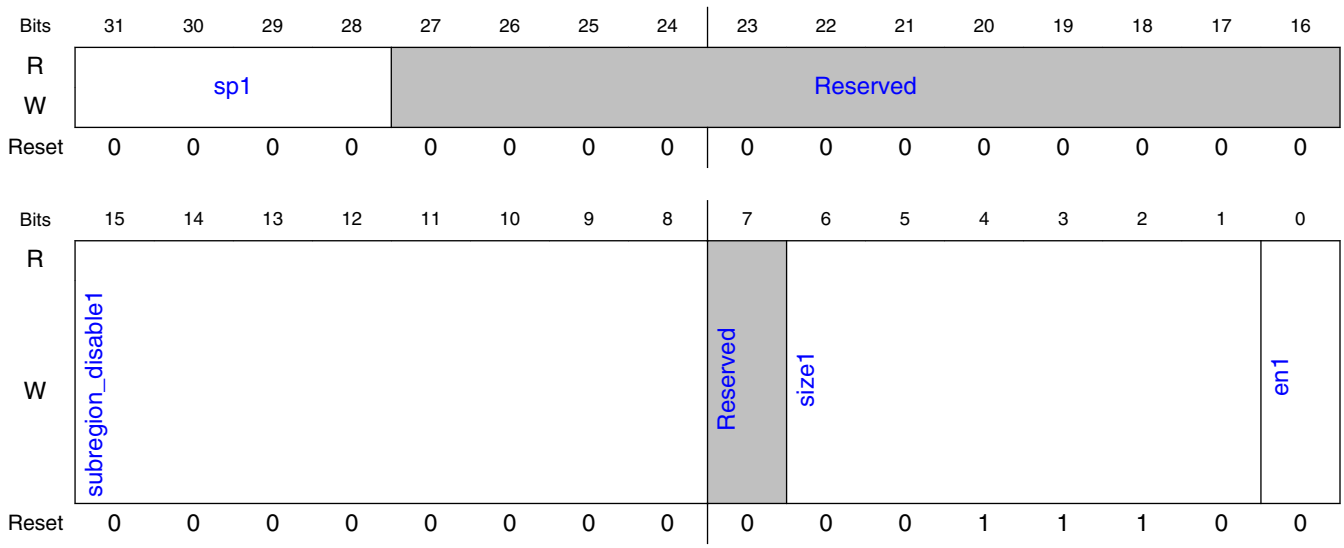
Table continues on the next page...

Table 10-1. Region size (continued)

b100100	128GB	Bits [36:15] must be zero
b100101	256GB	Bits [37:15] must be zero
b100110	512GB	Bits [38:15] must be zero
b100111	1TB	Bits [39:15] must be zero
b101000	2TB	Bits [40:15] must be zero
b101001	4TB	Bits [41:15] must be zero
b101010	8TB	Bits [42:15] must be zero
b101011	16TB	Bits [43:15] must be zero
b101100	32TB	Bits [44:15] must be zero
b101101	64TB	Bits [45:15] must be zero
b101110	128TB	Bits [46:15] must be zero
b101111	256TB	Bits [47:15] must be zero
b110000	512TB	Bits [48:15] must be zero
b110001	1PB	Bits [49:15] must be zero
b110010	2PB	Bits [50:15] must be zero
b110011	4PB	Bits [51:15] must be zero
b110100	8PB	Bits [52:15] must be zero
b110101	16PB	Bits [53:15] must be zero
b110110	32PB	Bits [54:15] must be zero
b110111	64PB	Bits [55:15] must be zero
b111000	128PB	Bits [56:15] must be zero
b111001	256PB	Bits [57:15] must be zero
b111010	512PB	Bits [58:15] must be zero
b111011	1EB	Bits [59:15] must be zero
b111100	2EB	Bits [60:15] must be zero
b111101	4EB	Bits [61:15] must be zero
b111110	8EB	Bits [62:15] must be zero
b111111	16EB	Bits [63:15] must be zero

[1]The region_setup_low_<n> Register and region_setup_high_<n> Register contain the base address. See Table 313 on page 318 and Table 314 on page 319.

10.3.19.3 Diagram



10.3.19.4 Fields

Field	Function
31-28 sp1	sp1 Permissions setting for region *n*. If an AXI transaction occurs to region *n*, the value in the sp*n* field controls whether the TZASC permits the transaction to proceed.
27-16 —	Reserved, Should be Zero (SBZ).
15-8 subregion_disable1	subregion_disable1 Regions are split into eight equal-sized sub-regions, and each bit enables the corresponding subregion to be disabled: Bit [15] = 1 Subregion 7 is disabled. Bit [14] = 1 Subregion 6 is disabled. Bit [13] = 1 Subregion 5 is disabled. Bit [12] = 1 Subregion 4 is disabled. Bit [11] = 1 Subregion 3 is disabled. Bit [10] = 1 Subregion 2 is disabled. Bit [9] = 1 Subregion 1 is disabled. Bit [8] = 1 Subregion 0 is disabled.
7 —	Reserved, Should be Zero (SBZ).
6-1 size1	size1

Table continues on the next page...

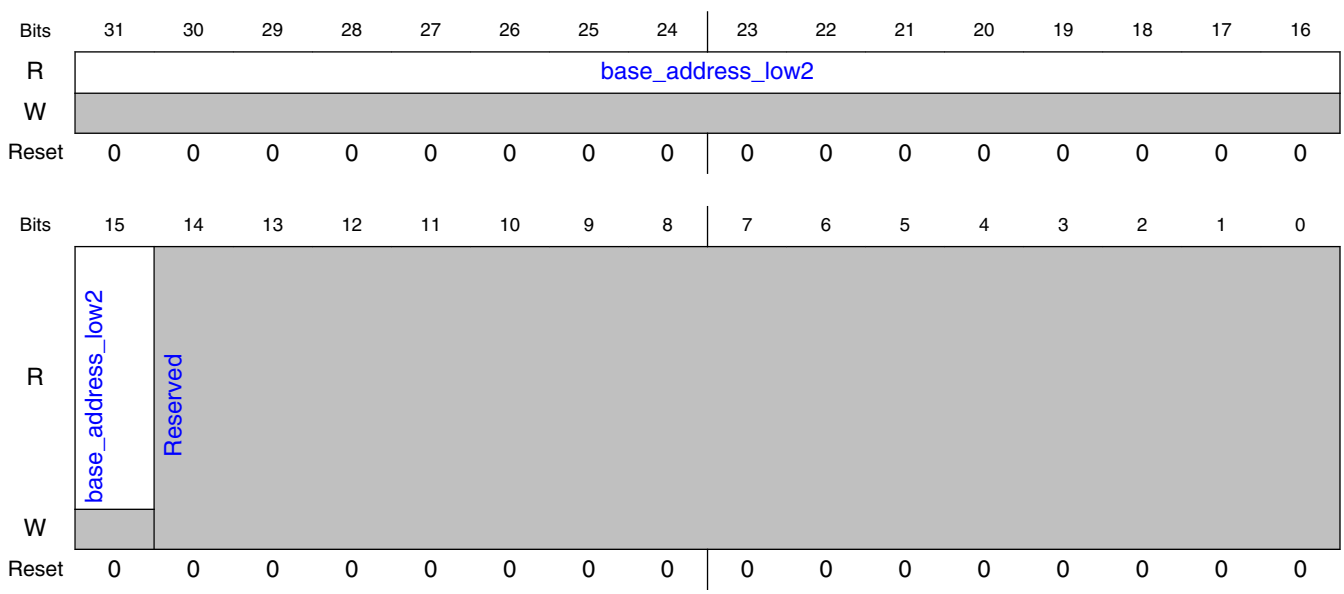
Field	Function
	Size of region *n*.The AXI address width, that is AXI_ADDRESS_MSB+1, controls the upper limit value of this field. The table below shows how the size *n* field controls the region size and what constraints, if any, the TZASC applies to the base address to ensure that a region starts on the boundary of the region size.
0 en1	en1 Enable for region *n*: 0 = region *n* is disabled 1 = region *n* is enabled.

10.3.20 Region Setup Low 2 Register (region_setup_low_2)

10.3.20.1 Offset

Register	Offset
region_setup_low_2	120h

10.3.20.2 Diagram



10.3.20.3 Fields

Field	Function
31-15 base_address_low2	base_address_low2 Controls the base address [31:15] of region n. For region 0, this field is Read Only (RO). The TZASC sets the base address of region 0 to 0x0. The TZASC only permits a region to start at address 0x0, or at a multiple of its region size. For example, if the size of a region is 512MB, and it is not at address 0x0, the only valid settings for this field are: b00100000000000000 b01000000000000000 b01100000000000000 b10000000000000000 b10100000000000000 b11000000000000000 b11100000000000000 If you attempt to set an inappropriate base address for the size of the region, the TZASC ignores certain bits depending on the region size. See Table on register size for more information.
14-0 —	Reserved, Should be Zero (SBZ).

10.3.21 Region Setup High 2 Register (region_setup_high_2)

10.3.21.1 Offset

Register	Offset
region_setup_high_2	124h

10.3.21.2 Function

The region_setup_high register is none for region 0, for other regions it controls the base address [63:32] of region n.There are no usage constraints. Available in all configurations of the TZASC.

10.3.21.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	base_address_high2															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	base_address_high2															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

10.3.21.4 Fields

Field	Function
31-0 base_address_high2	base_address_high2 Controls the base address [63:32] of region n. For region 0, this field is RO. The TZASC sets the base address of region 0 to 0x0. The TZASC only permits a region to start at address 0x0, or at a multiple of its region size. If you program a region size to be 8GB or more, then the TZASC might ignore certain bits depending on the region size.

10.3.22 Region Attributes 2 Register (region_attributes_2)

10.3.22.1 Offset

Register	Offset
region_attributes_2	128h

10.3.22.2 Function

The region_attributes_n register controls the permissions, region size, subregion disable, and region enable. There are no usage constraints. Available in all configurations of the TZASC.

Table 10-2. Region size

size<n> field	Size of region <n>	Base address [1] constraints
b000000-b001101	Reserved	-
b001110	32KB	-
b001111	64KB	Bit [15] must be zero
b010000	128KB	Bits [16:15] must be zero
b010001	256KB	Bits [17:15] must be zero
b010010	512KB	Bits [18:15] must be zero
b010011	1MB	Bits [19:15] must be zero
b010100	2MB	Bits [20:15] must be zero
b010101	4MB	Bits [21:15] must be zero
b010110	8MB	Bits [22:15] must be zero
b010111	16MB	Bits [23:15] must be zero
b011000	32MB	Bits [24:15] must be zero
b011001	64MB	Bits [25:15] must be zero
b011010	128MB	Bits [26:15] must be zero
b011011	256MB	Bits [27:15] must be zero
b011100	512MB	Bits [28:15] must be zero
b011101	1GB	Bits [29:15] must be zero
b011110	2GB	Bits [30:15] must be zero
b011111	4GB	Bits [31:15] must be zero
b100000	8GB	Bits [32:15] must be zero
b100001	16GB	Bits [33:15] must be zero
b100010	32GB	Bits [34:15] must be zero
b100011	64GB	Bits [35:15] must be zero
b100100	128GB	Bits [36:15] must be zero
b100101	256GB	Bits [37:15] must be zero
b100110	512GB	Bits [38:15] must be zero
b100111	1TB	Bits [39:15] must be zero
b101000	2TB	Bits [40:15] must be zero
b101001	4TB	Bits [41:15] must be zero
b101010	8TB	Bits [42:15] must be zero
b101011	16TB	Bits [43:15] must be zero
b101100	32TB	Bits [44:15] must be zero
b101101	64TB	Bits [45:15] must be zero
b101110	128TB	Bits [46:15] must be zero
b101111	256TB	Bits [47:15] must be zero
b110000	512TB	Bits [48:15] must be zero
b110001	1PB	Bits [49:15] must be zero
b110010	2PB	Bits [50:15] must be zero
b110011	4PB	Bits [51:15] must be zero

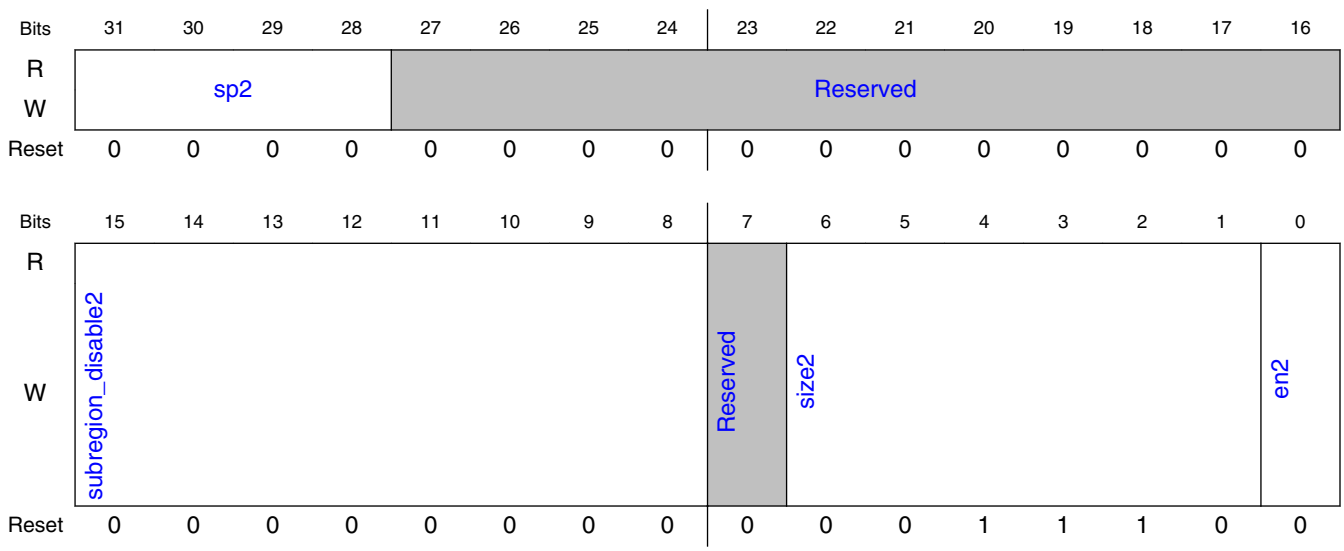
Table continues on the next page...

Table 10-2. Region size (continued)

b110100	8PB	Bits [52:15] must be zero
b110101	16PB	Bits [53:15] must be zero
b110110	32PB	Bits [54:15] must be zero
b110111	64PB	Bits [55:15] must be zero
b111000	128PB	Bits [56:15] must be zero
b111001	256PB	Bits [57:15] must be zero
b111010	512PB	Bits [58:15] must be zero
b111011	1EB	Bits [59:15] must be zero
b111100	2EB	Bits [60:15] must be zero
b111101	4EB	Bits [61:15] must be zero
b111110	8EB	Bits [62:15] must be zero
b111111	16EB	Bits [63:15] must be zero

[1]The region_setup_low_<n> Register and region_setup_high_<n> Register contain the base address. See Table 313 on page 318 and Table 314 on page 319.

10.3.22.3 Diagram



10.3.22.4 Fields

Field	Function
31-28 sp2	sp2

Table continues on the next page...

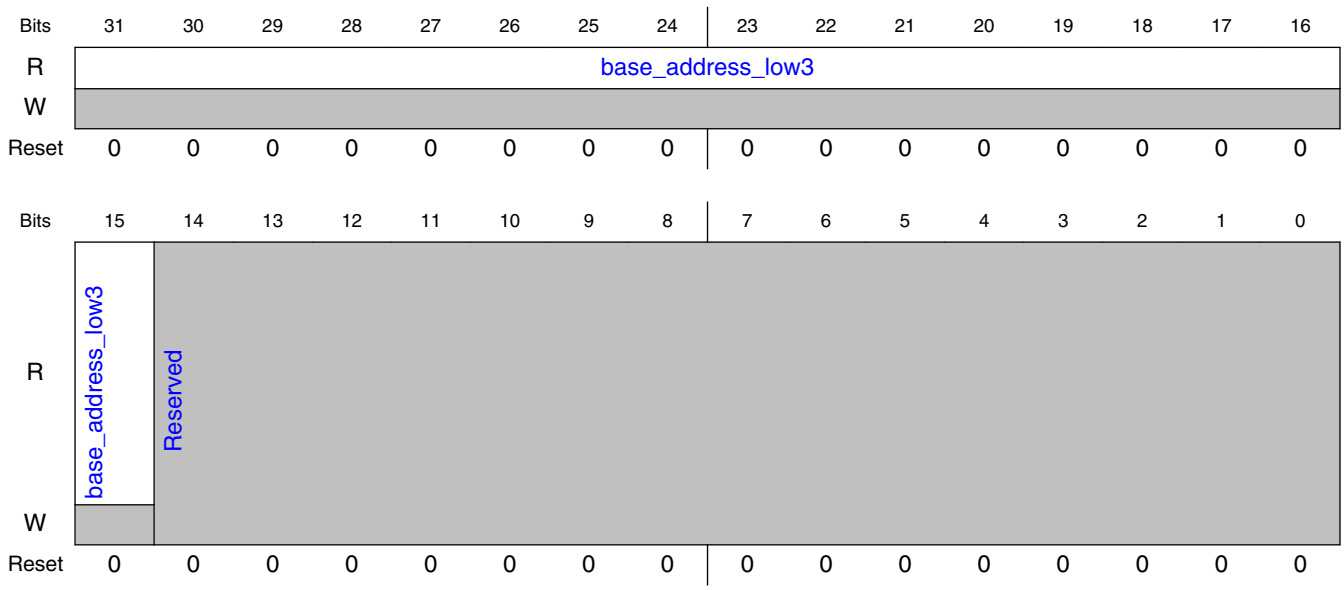
Field	Function
	Permissions setting for region *n*. If an AXI transaction occurs to region *n*, the value in the sp*n* field controls whether the TZASC permits the transaction to proceed.
27-16 —	Reserved, Should be Zero (SBZ).
15-8 subregion_disable2	subregion_disable2 Regions are split into eight equal-sized sub-regions, and each bit enables the corresponding subregion to be disabled: Bit [15] = 1 Subregion 7 is disabled. Bit [14] = 1 Subregion 6 is disabled. Bit [13] = 1 Subregion 5 is disabled. Bit [12] = 1 Subregion 4 is disabled. Bit [11] = 1 Subregion 3 is disabled. Bit [10] = 1 Subregion 2 is disabled. Bit [9] = 1 Subregion 1 is disabled. Bit [8] = 1 Subregion 0 is disabled.
7 —	Reserved, Should be Zero (SBZ).
6-1 size2	size2 Size of region *n*.The AXI address width, that is AXI_ADDRESS_MSB+1, controls the upper limit value of this field. The table below shows how the size *n* field controls the region size and what constraints, if any, the TZASC applies to the base address to ensure that a region starts on the boundary of the region size.
0 en2	en2 Enable for region *n*: 0 = region *n* is disabled 1 = region *n* is enabled.

10.3.23 Region Setup Low 3 Register (region_setup_low_3)

10.3.23.1 Offset

Register	Offset
region_setup_low_3	130h

10.3.23.2 Diagram



10.3.23.3 Fields

Field	Function
31-15 base_address_low3	base_address_low3 Controls the base address [31:15] of region n. For region 0, this field is Read Only (RO). The TZASC sets the base address of region 0 to 0x0. The TZASC only permits a region to start at address 0x0, or at a multiple of its region size. For example, if the size of a region is 512MB, and it is not at address 0x0, the only valid settings for this field are: b00100000000000000 b01000000000000000 b01100000000000000 b10000000000000000 b10100000000000000 b11000000000000000 b11100000000000000 If you attempt to set an inappropriate base address for the size of the region, the TZASC ignores certain bits depending on the region size. See Table on register size for more information.
14-0 —	Reserved, Should be Zero (SBZ).

10.3.24 Region Setup High 3 Register (region_setup_high_3)

10.3.24.1 Offset

Register	Offset
region_setup_high_3	134h

10.3.24.2 Function

The region_setup_high register is none for region 0, for other regions it controls the base address [63:32] of region n.There are no usage constraints. Available in all configurations of the TZASC.

10.3.24.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	base_address_high3															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	base_address_high3															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

10.3.24.4 Fields

Field	Function
31-0 base_address_high3	base_address_high3 Controls the base address [63:32] of region n. For region 0, this field is RO. The TZASC sets the base address of region 0 to 0x0. The TZASC only permits a region to start at address 0x0, or at a multiple of its region size. If you program a region size to be 8GB or more, then the TZASC might ignore certain bits depending on the region size.

10.3.25 Region Attributes 3 Register (region_attributes_3)

10.3.25.1 Offset

Register	Offset
region_attributes_3	138h

10.3.25.2 Function

The region_attributes_n register controls the permissions, region size, subregion disable, and region enable. There are no usage constraints. Available in all configurations of the TZASC.

Table 10-3. Region size

size<n> field	Size of region <n>	Base address [1] constraints
b000000-b001101	Reserved	-
b001110	32KB	-
b001111	64KB	Bit [15] must be zero
b010000	128KB	Bits [16:15] must be zero
b010001	256KB	Bits [17:15] must be zero
b010010	512KB	Bits [18:15] must be zero
b010011	1MB	Bits [19:15] must be zero
b010100	2MB	Bits [20:15] must be zero
b010101	4MB	Bits [21:15] must be zero
b010110	8MB	Bits [22:15] must be zero
b010111	16MB	Bits [23:15] must be zero
b011000	32MB	Bits [24:15] must be zero
b011001	64MB	Bits [25:15] must be zero
b011010	128MB	Bits [26:15] must be zero
b011011	256MB	Bits [27:15] must be zero
b011100	512MB	Bits [28:15] must be zero
b011101	1GB	Bits [29:15] must be zero
b011110	2GB	Bits [30:15] must be zero
b011111	4GB	Bits [31:15] must be zero
b100000	8GB	Bits [32:15] must be zero
b100001	16GB	Bits [33:15] must be zero
b100010	32GB	Bits [34:15] must be zero
b100011	64GB	Bits [35:15] must be zero

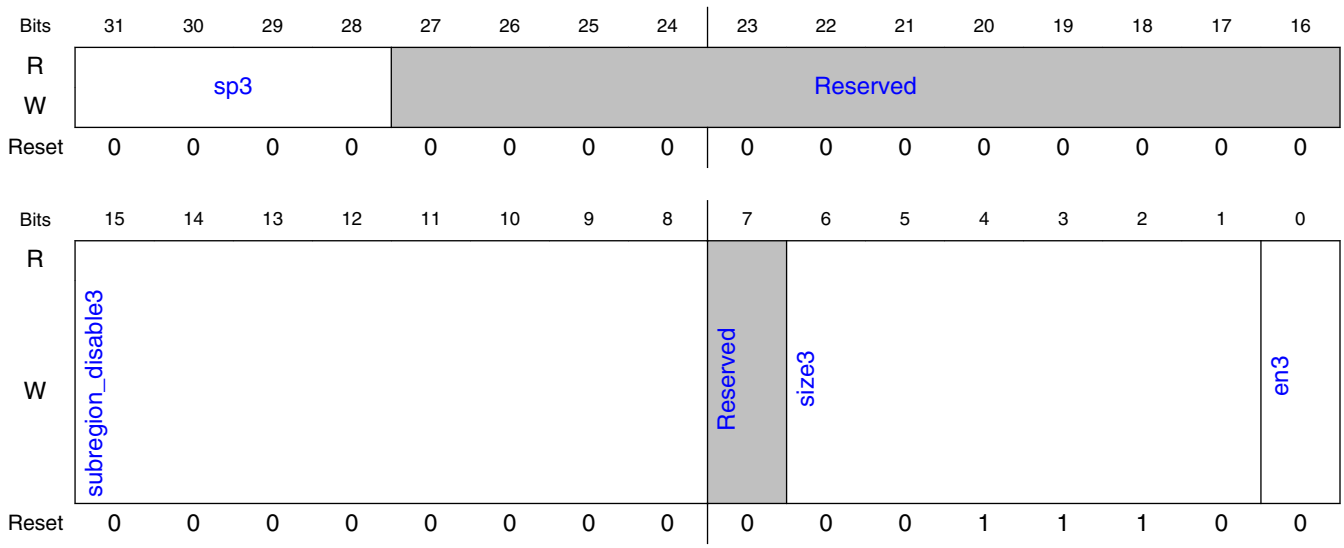
Table continues on the next page...

Table 10-3. Region size (continued)

b100100	128GB	Bits [36:15] must be zero
b100101	256GB	Bits [37:15] must be zero
b100110	512GB	Bits [38:15] must be zero
b100111	1TB	Bits [39:15] must be zero
b101000	2TB	Bits [40:15] must be zero
b101001	4TB	Bits [41:15] must be zero
b101010	8TB	Bits [42:15] must be zero
b101011	16TB	Bits [43:15] must be zero
b101100	32TB	Bits [44:15] must be zero
b101101	64TB	Bits [45:15] must be zero
b101110	128TB	Bits [46:15] must be zero
b101111	256TB	Bits [47:15] must be zero
b110000	512TB	Bits [48:15] must be zero
b110001	1PB	Bits [49:15] must be zero
b110010	2PB	Bits [50:15] must be zero
b110011	4PB	Bits [51:15] must be zero
b110100	8PB	Bits [52:15] must be zero
b110101	16PB	Bits [53:15] must be zero
b110110	32PB	Bits [54:15] must be zero
b110111	64PB	Bits [55:15] must be zero
b111000	128PB	Bits [56:15] must be zero
b111001	256PB	Bits [57:15] must be zero
b111010	512PB	Bits [58:15] must be zero
b111011	1EB	Bits [59:15] must be zero
b111100	2EB	Bits [60:15] must be zero
b111101	4EB	Bits [61:15] must be zero
b111110	8EB	Bits [62:15] must be zero
b111111	16EB	Bits [63:15] must be zero

[1]The region_setup_low_<n> Register and region_setup_high_<n> Register contain the base address. See Table 313 on page 318 and Table 314 on page 319.

10.3.25.3 Diagram



10.3.25.4 Fields

Field	Function
31-28 sp3	sp3 Permissions setting for region *n*. If an AXI transaction occurs to region *n*, the value in the sp*n* field controls whether the TZASC permits the transaction to proceed.
27-16 —	Reserved, Should be Zero (SBZ).
15-8 subregion_disable3	subregion_disable3 Regions are split into eight equal-sized sub-regions, and each bit enables the corresponding subregion to be disabled: Bit [15] = 1 Subregion 7 is disabled. Bit [14] = 1 Subregion 6 is disabled. Bit [13] = 1 Subregion 5 is disabled. Bit [12] = 1 Subregion 4 is disabled. Bit [11] = 1 Subregion 3 is disabled. Bit [10] = 1 Subregion 2 is disabled. Bit [9] = 1 Subregion 1 is disabled. Bit [8] = 1 Subregion 0 is disabled.
7 —	Reserved, Should be Zero (SBZ).
6-1 size3	size3

Table continues on the next page...

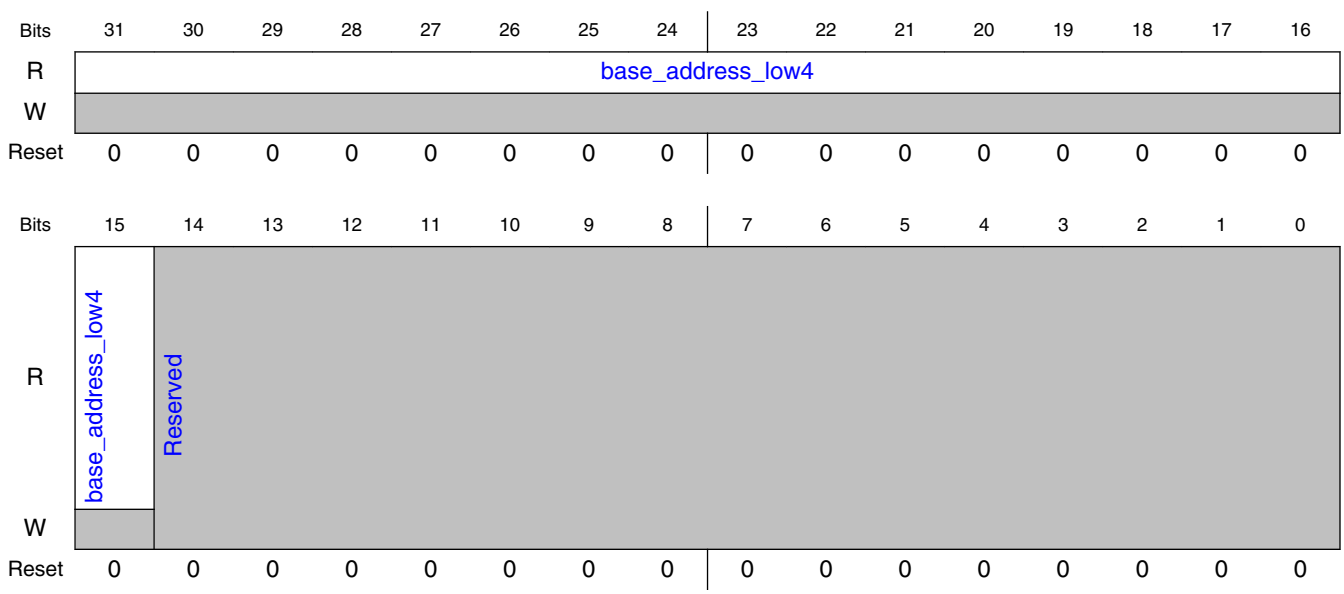
Field	Function
	Size of region *n*.The AXI address width, that is AXI_ADDRESS_MSB+1, controls the upper limit value of this field. The table below shows how the size *n* field controls the region size and what constraints, if any, the TZASC applies to the base address to ensure that a region starts on the boundary of the region size.
0 en3	en3 Enable for region *n*: 0 = region *n* is disabled 1 = region *n* is enabled.

10.3.26 Region Setup Low 4 Register (region_setup_low_4)

10.3.26.1 Offset

Register	Offset
region_setup_low_4	140h

10.3.26.2 Diagram



10.3.26.3 Fields

Field	Function
31-15 base_address_low4	base_address_low4 Controls the base address [31:15] of region n. For region 0, this field is Read Only (RO). The TZASC sets the base address of region 0 to 0x0. The TZASC only permits a region to start at address 0x0, or at a multiple of its region size. For example, if the size of a region is 512MB, and it is not at address 0x0, the only valid settings for this field are: b00100000000000000 b01000000000000000 b01100000000000000 b10000000000000000 b10100000000000000 b11000000000000000 b11100000000000000 If you attempt to set an inappropriate base address for the size of the region, the TZASC ignores certain bits depending on the region size. See Table on register size for more information.
14-0 —	Reserved, Should be Zero (SBZ).

10.3.27 Region Setup High 4 Register (region_setup_high_4)

10.3.27.1 Offset

Register	Offset
region_setup_high_4	144h

10.3.27.2 Function

The region_setup_high register is none for region 0, for other regions it controls the base address [63:32] of region n.There are no usage constraints. Available in all configurations of the TZASC.

10.3.27.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	base_address_high4															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	base_address_high4															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

10.3.27.4 Fields

Field	Function
31-0 base_address_high4	base_address_high4 Controls the base address [63:32] of region n. For region 0, this field is RO. The TZASC sets the base address of region 0 to 0x0. The TZASC only permits a region to start at address 0x0, or at a multiple of its region size. If you program a region size to be 8GB or more, then the TZASC might ignore certain bits depending on the region size.

10.3.28 Region Attributes 4 Register (region_attributes_4)

10.3.28.1 Offset

Register	Offset
region_attributes_4	148h

10.3.28.2 Function

The region_attributes_n register controls the permissions, region size, subregion disable, and region enable. There are no usage constraints. Available in all configurations of the TZASC.

Table 10-4. Region size

size<n> field	Size of region <n>	Base address [1] constraints
b000000-b001101	Reserved	-
b001110	32KB	-
b001111	64KB	Bit [15] must be zero
b010000	128KB	Bits [16:15] must be zero
b010001	256KB	Bits [17:15] must be zero
b010010	512KB	Bits [18:15] must be zero
b010011	1MB	Bits [19:15] must be zero
b010100	2MB	Bits [20:15] must be zero
b010101	4MB	Bits [21:15] must be zero
b010110	8MB	Bits [22:15] must be zero
b010111	16MB	Bits [23:15] must be zero
b011000	32MB	Bits [24:15] must be zero
b011001	64MB	Bits [25:15] must be zero
b011010	128MB	Bits [26:15] must be zero
b011011	256MB	Bits [27:15] must be zero
b011100	512MB	Bits [28:15] must be zero
b011101	1GB	Bits [29:15] must be zero
b011110	2GB	Bits [30:15] must be zero
b011111	4GB	Bits [31:15] must be zero
b100000	8GB	Bits [32:15] must be zero
b100001	16GB	Bits [33:15] must be zero
b100010	32GB	Bits [34:15] must be zero
b100011	64GB	Bits [35:15] must be zero
b100100	128GB	Bits [36:15] must be zero
b100101	256GB	Bits [37:15] must be zero
b100110	512GB	Bits [38:15] must be zero
b100111	1TB	Bits [39:15] must be zero
b101000	2TB	Bits [40:15] must be zero
b101001	4TB	Bits [41:15] must be zero
b101010	8TB	Bits [42:15] must be zero
b101011	16TB	Bits [43:15] must be zero
b101100	32TB	Bits [44:15] must be zero
b101101	64TB	Bits [45:15] must be zero
b101110	128TB	Bits [46:15] must be zero
b101111	256TB	Bits [47:15] must be zero
b110000	512TB	Bits [48:15] must be zero
b110001	1PB	Bits [49:15] must be zero
b110010	2PB	Bits [50:15] must be zero
b110011	4PB	Bits [51:15] must be zero

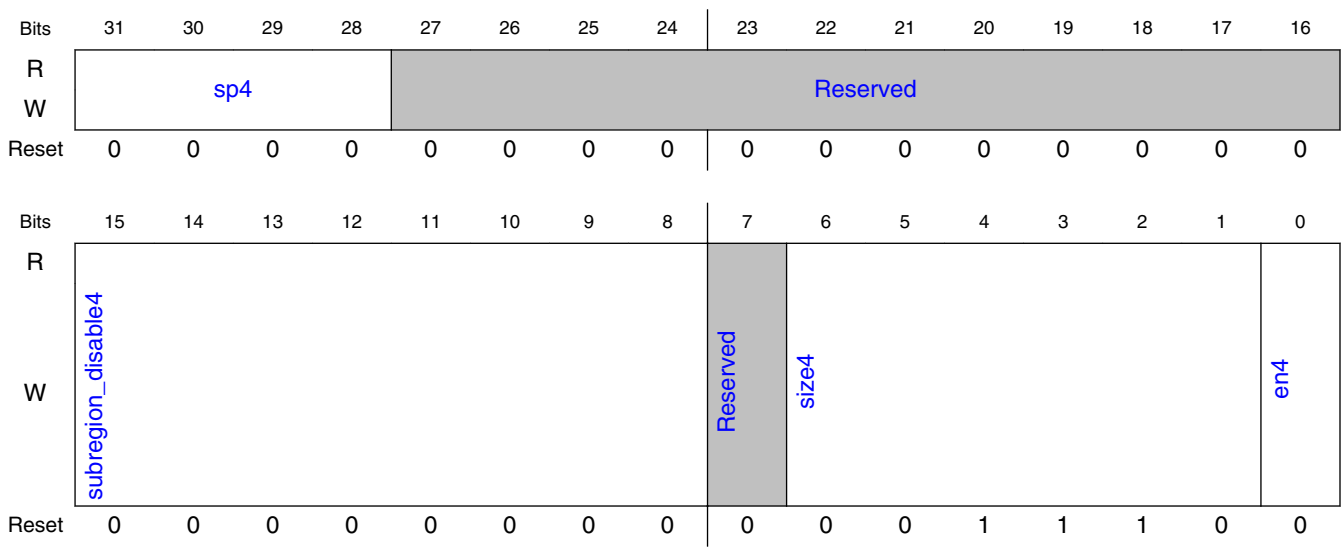
Table continues on the next page...

Table 10-4. Region size (continued)

b110100	8PB	Bits [52:15] must be zero
b110101	16PB	Bits [53:15] must be zero
b110110	32PB	Bits [54:15] must be zero
b110111	64PB	Bits [55:15] must be zero
b111000	128PB	Bits [56:15] must be zero
b111001	256PB	Bits [57:15] must be zero
b111010	512PB	Bits [58:15] must be zero
b111011	1EB	Bits [59:15] must be zero
b111100	2EB	Bits [60:15] must be zero
b111101	4EB	Bits [61:15] must be zero
b111110	8EB	Bits [62:15] must be zero
b111111	16EB	Bits [63:15] must be zero

[1]The region_setup_low_<n> Register and region_setup_high_<n> Register contain the base address. See Table 313 on page 318 and Table 314 on page 319.

10.3.28.3 Diagram



10.3.28.4 Fields

Field	Function
31-28 sp4	sp4

Table continues on the next page...

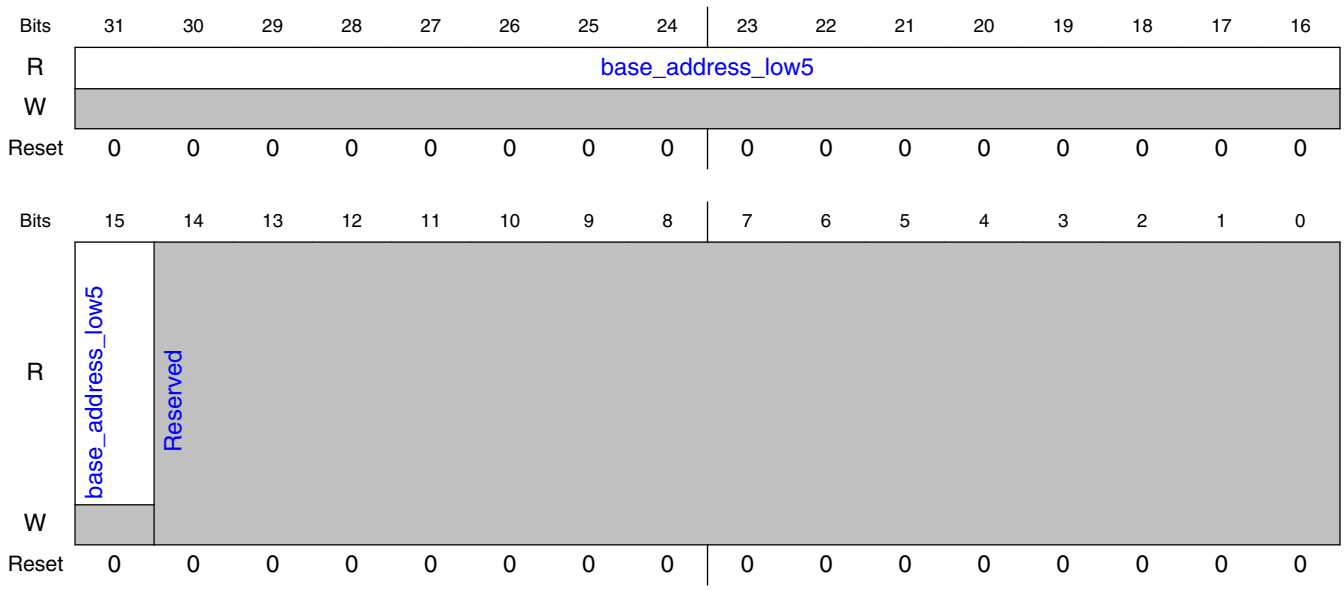
Field	Function
	Permissions setting for region *n*. If an AXI transaction occurs to region *n*, the value in the sp*n* field controls whether the TZASC permits the transaction to proceed.
27-16 —	Reserved, Should be Zero (SBZ).
15-8 subregion_disable4	subregion_disable4 Regions are split into eight equal-sized sub-regions, and each bit enables the corresponding subregion to be disabled: Bit [15] = 1 Subregion 7 is disabled. Bit [14] = 1 Subregion 6 is disabled. Bit [13] = 1 Subregion 5 is disabled. Bit [12] = 1 Subregion 4 is disabled. Bit [11] = 1 Subregion 3 is disabled. Bit [10] = 1 Subregion 2 is disabled. Bit [9] = 1 Subregion 1 is disabled. Bit [8] = 1 Subregion 0 is disabled.
7 —	Reserved, Should be Zero (SBZ).
6-1 size4	size4 Size of region *n*.The AXI address width, that is AXI_ADDRESS_MSB+1, controls the upper limit value of this field. The table below shows how the size *n* field controls the region size and what constraints, if any, the TZASC applies to the base address to ensure that a region starts on the boundary of the region size.
0 en4	en4 Enable for region *n*: 0 = region *n* is disabled 1 = region *n* is enabled.

10.3.29 Region Setup Low 5 Register (region_setup_low_5)

10.3.29.1 Offset

Register	Offset
region_setup_low_5	150h

10.3.29.2 Diagram



10.3.29.3 Fields

Field	Function
31-15 base_address_low5	base_address_low5 Controls the base address [31:15] of region n. For region 0, this field is Read Only (RO). The TZASC sets the base address of region 0 to 0x0. The TZASC only permits a region to start at address 0x0, or at a multiple of its region size. For example, if the size of a region is 512MB, and it is not at address 0x0, the only valid settings for this field are: b00100000000000000 b01000000000000000 b01100000000000000 b10000000000000000 b10100000000000000 b11000000000000000 b11100000000000000 If you attempt to set an inappropriate base address for the size of the region, the TZASC ignores certain bits depending on the region size. See Table on register size for more information.
14-0 —	Reserved, Should be Zero (SBZ).

10.3.30 Region Setup High 5 Register (region_setup_high_5)

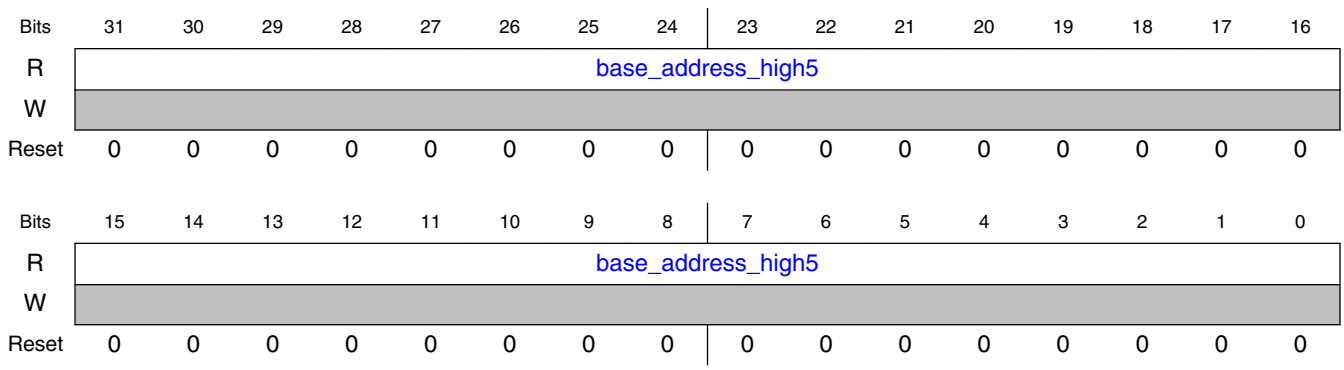
10.3.30.1 Offset

Register	Offset
region_setup_high_5	154h

10.3.30.2 Function

The region_setup_high register is none for region 0, for other regions it controls the base address [63:32] of region n.There are no usage constraints. Available in all configurations of the TZASC.

10.3.30.3 Diagram



10.3.30.4 Fields

Field	Function
31-0 base_address_high5	base_address_high5 Controls the base address [63:32] of region n. For region 0, this field is RO. The TZASC sets the base address of region 0 to 0x0. The TZASC only permits a region to start at address 0x0, or at a multiple of its region size. If you program a region size to be 8GB or more, then the TZASC might ignore certain bits depending on the region size.

10.3.31 Region Attributes 5 Register (region_attributes_5)

10.3.31.1 Offset

Register	Offset
region_attributes_5	158h

10.3.31.2 Function

The region_attributes_n register controls the permissions, region size, subregion disable, and region enable. There are no usage constraints. Available in all configurations of the TZASC.

Table 10-5. Region size

size<n> field	Size of region <n>	Base address [1] constraints
b000000-b001101	Reserved	-
b001110	32KB	-
b001111	64KB	Bit [15] must be zero
b010000	128KB	Bits [16:15] must be zero
b010001	256KB	Bits [17:15] must be zero
b010010	512KB	Bits [18:15] must be zero
b010011	1MB	Bits [19:15] must be zero
b010100	2MB	Bits [20:15] must be zero
b010101	4MB	Bits [21:15] must be zero
b010110	8MB	Bits [22:15] must be zero
b010111	16MB	Bits [23:15] must be zero
b011000	32MB	Bits [24:15] must be zero
b011001	64MB	Bits [25:15] must be zero
b011010	128MB	Bits [26:15] must be zero
b011011	256MB	Bits [27:15] must be zero
b011100	512MB	Bits [28:15] must be zero
b011101	1GB	Bits [29:15] must be zero
b011110	2GB	Bits [30:15] must be zero
b011111	4GB	Bits [31:15] must be zero
b100000	8GB	Bits [32:15] must be zero
b100001	16GB	Bits [33:15] must be zero
b100010	32GB	Bits [34:15] must be zero
b100011	64GB	Bits [35:15] must be zero

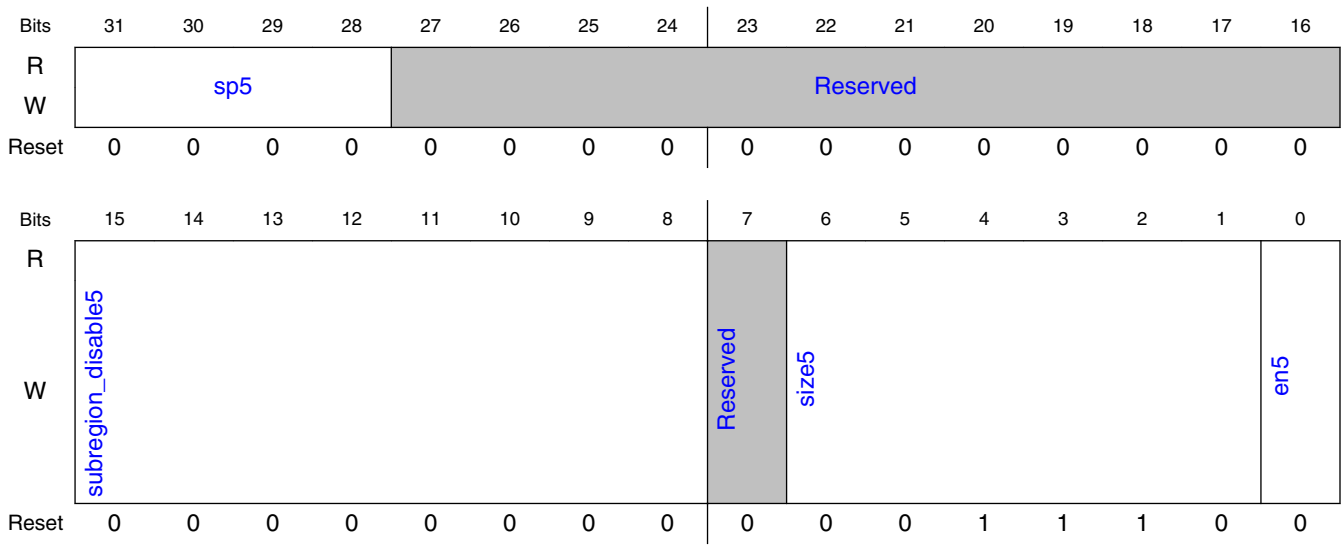
Table continues on the next page...

Table 10-5. Region size (continued)

b100100	128GB	Bits [36:15] must be zero
b100101	256GB	Bits [37:15] must be zero
b100110	512GB	Bits [38:15] must be zero
b100111	1TB	Bits [39:15] must be zero
b101000	2TB	Bits [40:15] must be zero
b101001	4TB	Bits [41:15] must be zero
b101010	8TB	Bits [42:15] must be zero
b101011	16TB	Bits [43:15] must be zero
b101100	32TB	Bits [44:15] must be zero
b101101	64TB	Bits [45:15] must be zero
b101110	128TB	Bits [46:15] must be zero
b101111	256TB	Bits [47:15] must be zero
b110000	512TB	Bits [48:15] must be zero
b110001	1PB	Bits [49:15] must be zero
b110010	2PB	Bits [50:15] must be zero
b110011	4PB	Bits [51:15] must be zero
b110100	8PB	Bits [52:15] must be zero
b110101	16PB	Bits [53:15] must be zero
b110110	32PB	Bits [54:15] must be zero
b110111	64PB	Bits [55:15] must be zero
b111000	128PB	Bits [56:15] must be zero
b111001	256PB	Bits [57:15] must be zero
b111010	512PB	Bits [58:15] must be zero
b111011	1EB	Bits [59:15] must be zero
b111100	2EB	Bits [60:15] must be zero
b111101	4EB	Bits [61:15] must be zero
b111110	8EB	Bits [62:15] must be zero
b111111	16EB	Bits [63:15] must be zero

[1]The region_setup_low_<n> Register and region_setup_high_<n> Register contain the base address. See Table 313 on page 318 and Table 314 on page 319.

10.3.31.3 Diagram



10.3.31.4 Fields

Field	Function
31-28 sp5	sp5 Permissions setting for region *n*. If an AXI transaction occurs to region *n*, the value in the sp*n* field controls whether the TZASC permits the transaction to proceed.
27-16 —	Reserved, Should be Zero (SBZ).
15-8 subregion_disable5	subregion_disable5 Regions are split into eight equal-sized sub-regions, and each bit enables the corresponding subregion to be disabled: Bit [15] = 1 Subregion 7 is disabled. Bit [14] = 1 Subregion 6 is disabled. Bit [13] = 1 Subregion 5 is disabled. Bit [12] = 1 Subregion 4 is disabled. Bit [11] = 1 Subregion 3 is disabled. Bit [10] = 1 Subregion 2 is disabled. Bit [9] = 1 Subregion 1 is disabled. Bit [8] = 1 Subregion 0 is disabled.
7 —	Reserved, Should be Zero (SBZ).
6-1 size5	size5

Table continues on the next page...

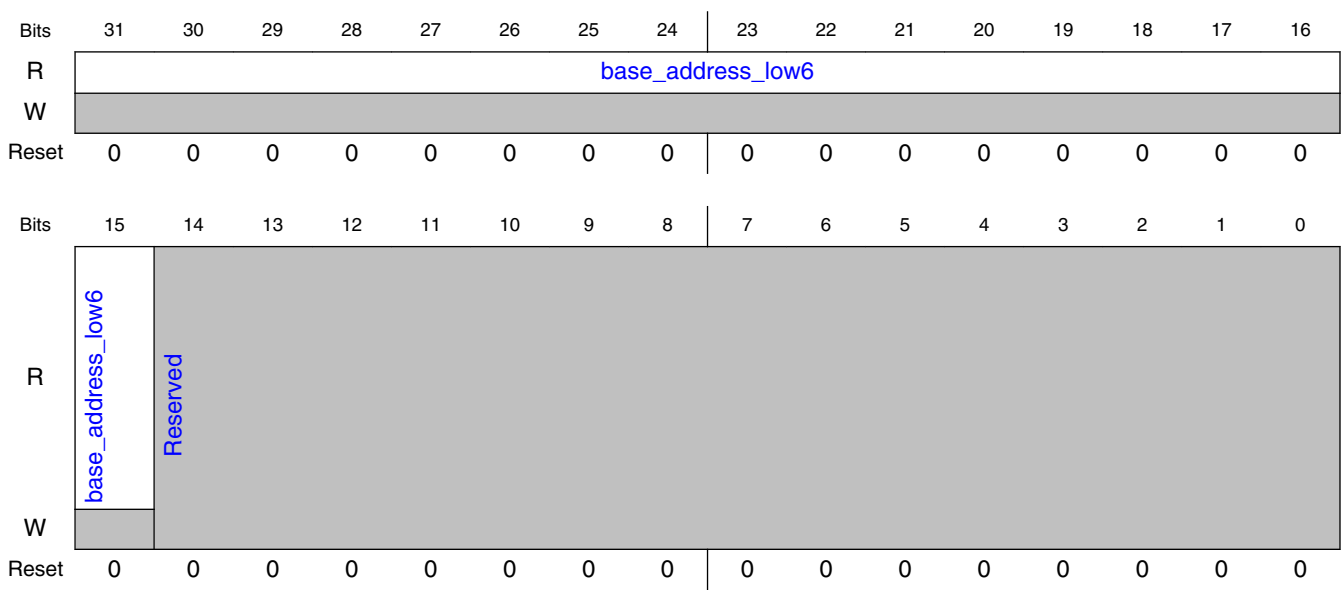
Field	Function
	Size of region *n*.The AXI address width, that is AXI_ADDRESS_MSB+1, controls the upper limit value of this field. The table below shows how the size *n* field controls the region size and what constraints, if any, the TZASC applies to the base address to ensure that a region starts on the boundary of the region size.
0 en5	en5 Enable for region *n*: 0 = region *n* is disabled 1 = region *n* is enabled.

10.3.32 Region Setup Low 6 Register (region_setup_low_6)

10.3.32.1 Offset

Register	Offset
region_setup_low_6	160h

10.3.32.2 Diagram



10.3.32.3 Fields

Field	Function
31-15 base_address_low6	base_address_low6 Controls the base address [31:15] of region n. For region 0, this field is Read Only (RO). The TZASC sets the base address of region 0 to 0x0. The TZASC only permits a region to start at address 0x0, or at a multiple of its region size. For example, if the size of a region is 512MB, and it is not at address 0x0, the only valid settings for this field are: b00100000000000000 b01000000000000000 b01100000000000000 b10000000000000000 b10100000000000000 b11000000000000000 b11100000000000000 If you attempt to set an inappropriate base address for the size of the region, the TZASC ignores certain bits depending on the region size. See Table on register size for more information.
14-0 —	Reserved, Should be Zero (SBZ).

10.3.33 Region Setup High 6 Register (region_setup_high_6)

10.3.33.1 Offset

Register	Offset
region_setup_high_6	164h

10.3.33.2 Function

The region_setup_high register is none for region 0, for other regions it controls the base address [63:32] of region n.There are no usage constraints. Available in all configurations of the TZASC.

10.3.33.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	base_address_high6															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	base_address_high6															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

10.3.33.4 Fields

Field	Function
31-0 base_address_high6	base_address_high6 Controls the base address [63:32] of region n. For region 0, this field is RO. The TZASC sets the base address of region 0 to 0x0. The TZASC only permits a region to start at address 0x0, or at a multiple of its region size. If you program a region size to be 8GB or more, then the TZASC might ignore certain bits depending on the region size.

10.3.34 Region Attributes 6 Register (region_attributes_6)

10.3.34.1 Offset

Register	Offset
region_attributes_6	168h

10.3.34.2 Function

The region_attributes_n register controls the permissions, region size, subregion disable, and region enable. There are no usage constraints. Available in all configurations of the TZASC.

Table 10-6. Region size

size<n> field	Size of region <n>	Base address [1] constraints
b000000-b001101	Reserved	-
b001110	32KB	-
b001111	64KB	Bit [15] must be zero
b010000	128KB	Bits [16:15] must be zero
b010001	256KB	Bits [17:15] must be zero
b010010	512KB	Bits [18:15] must be zero
b010011	1MB	Bits [19:15] must be zero
b010100	2MB	Bits [20:15] must be zero
b010101	4MB	Bits [21:15] must be zero
b010110	8MB	Bits [22:15] must be zero
b010111	16MB	Bits [23:15] must be zero
b011000	32MB	Bits [24:15] must be zero
b011001	64MB	Bits [25:15] must be zero
b011010	128MB	Bits [26:15] must be zero
b011011	256MB	Bits [27:15] must be zero
b011100	512MB	Bits [28:15] must be zero
b011101	1GB	Bits [29:15] must be zero
b011110	2GB	Bits [30:15] must be zero
b011111	4GB	Bits [31:15] must be zero
b100000	8GB	Bits [32:15] must be zero
b100001	16GB	Bits [33:15] must be zero
b100010	32GB	Bits [34:15] must be zero
b100011	64GB	Bits [35:15] must be zero
b100100	128GB	Bits [36:15] must be zero
b100101	256GB	Bits [37:15] must be zero
b100110	512GB	Bits [38:15] must be zero
b100111	1TB	Bits [39:15] must be zero
b101000	2TB	Bits [40:15] must be zero
b101001	4TB	Bits [41:15] must be zero
b101010	8TB	Bits [42:15] must be zero
b101011	16TB	Bits [43:15] must be zero
b101100	32TB	Bits [44:15] must be zero
b101101	64TB	Bits [45:15] must be zero
b101110	128TB	Bits [46:15] must be zero
b101111	256TB	Bits [47:15] must be zero
b110000	512TB	Bits [48:15] must be zero
b110001	1PB	Bits [49:15] must be zero
b110010	2PB	Bits [50:15] must be zero
b110011	4PB	Bits [51:15] must be zero

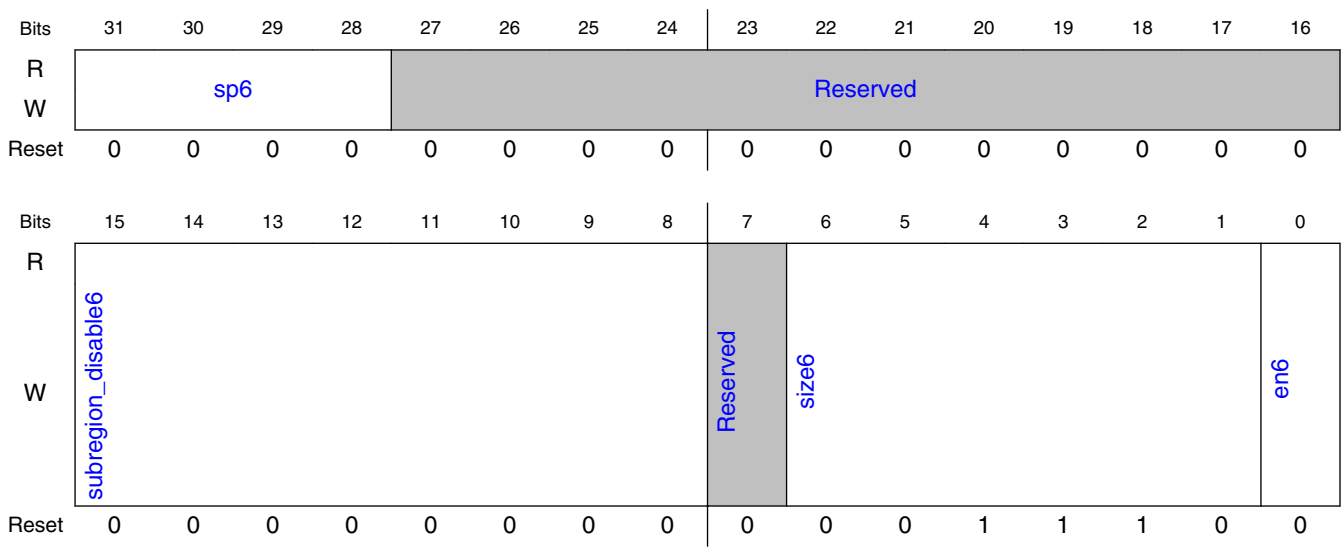
Table continues on the next page...

Table 10-6. Region size (continued)

b110100	8PB	Bits [52:15] must be zero
b110101	16PB	Bits [53:15] must be zero
b110110	32PB	Bits [54:15] must be zero
b110111	64PB	Bits [55:15] must be zero
b111000	128PB	Bits [56:15] must be zero
b111001	256PB	Bits [57:15] must be zero
b111010	512PB	Bits [58:15] must be zero
b111011	1EB	Bits [59:15] must be zero
b111100	2EB	Bits [60:15] must be zero
b111101	4EB	Bits [61:15] must be zero
b111110	8EB	Bits [62:15] must be zero
b111111	16EB	Bits [63:15] must be zero

[1]The region_setup_low_<n> Register and region_setup_high_<n> Register contain the base address. See Table 313 on page 318 and Table 314 on page 319.

10.3.34.3 Diagram



10.3.34.4 Fields

Field	Function
31-28 sp6	sp6

Table continues on the next page...

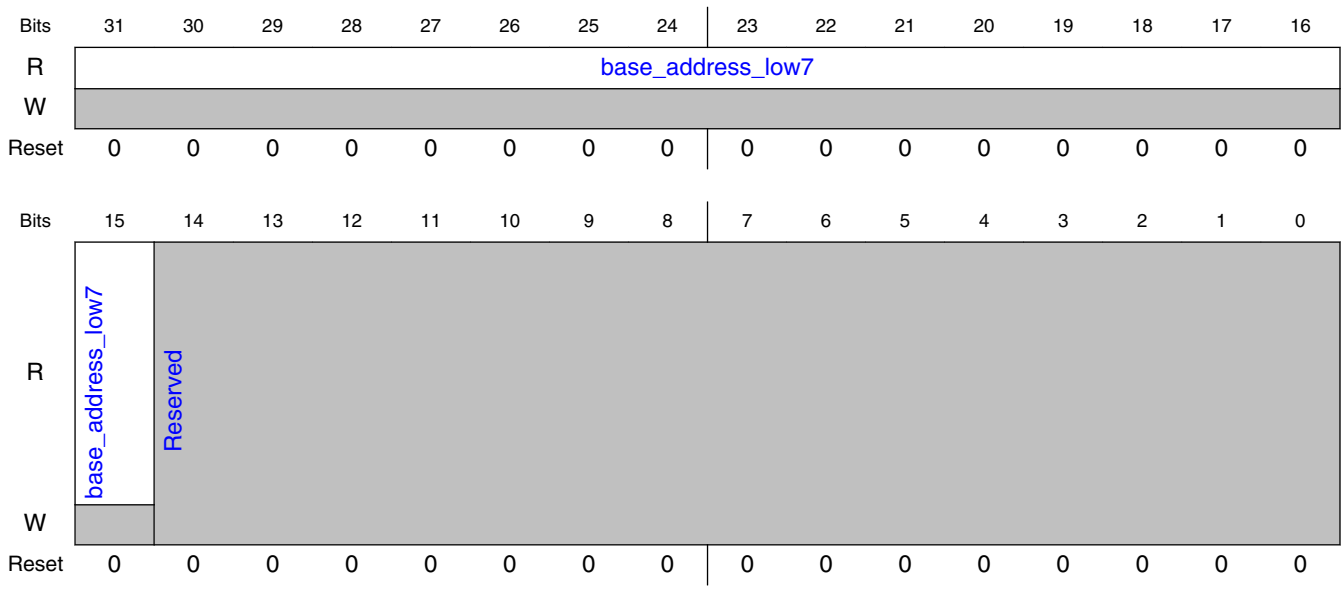
Field	Function
	Permissions setting for region *n*. If an AXI transaction occurs to region *n*, the value in the sp*n* field controls whether the TZASC permits the transaction to proceed.
27-16 —	Reserved, Should be Zero (SBZ).
15-8 subregion_disable6	subregion_disable6 Regions are split into eight equal-sized sub-regions, and each bit enables the corresponding subregion to be disabled: Bit [15] = 1 Subregion 7 is disabled. Bit [14] = 1 Subregion 6 is disabled. Bit [13] = 1 Subregion 5 is disabled. Bit [12] = 1 Subregion 4 is disabled. Bit [11] = 1 Subregion 3 is disabled. Bit [10] = 1 Subregion 2 is disabled. Bit [9] = 1 Subregion 1 is disabled. Bit [8] = 1 Subregion 0 is disabled.
7 —	Reserved, Should be Zero (SBZ).
6-1 size6	size6 Size of region *n*.The AXI address width, that is AXI_ADDRESS_MSB+1, controls the upper limit value of this field. The table below shows how the size *n* field controls the region size and what constraints, if any, the TZASC applies to the base address to ensure that a region starts on the boundary of the region size.
0 en6	en6 Enable for region *n*: 0 = region *n* is disabled 1 = region *n* is enabled.

10.3.35 Region Setup Low 7 Register (region_setup_low_7)

10.3.35.1 Offset

Register	Offset
region_setup_low_7	170h

10.3.35.2 Diagram



10.3.35.3 Fields

Field	Function
31-15 base_address_low7	base_address_low7 Controls the base address [31:15] of region n. For region 0, this field is Read Only (RO). The TZASC sets the base address of region 0 to 0x0. The TZASC only permits a region to start at address 0x0, or at a multiple of its region size. For example, if the size of a region is 512MB, and it is not at address 0x0, the only valid settings for this field are: b00100000000000000 b01000000000000000 b01100000000000000 b10000000000000000 b10100000000000000 b11000000000000000 b11100000000000000 If you attempt to set an inappropriate base address for the size of the region, the TZASC ignores certain bits depending on the region size. See Table on register size for more information.
14-0 —	Reserved, Should be Zero (SBZ).

10.3.36 Region Setup High 7 Register (region_setup_high_7)

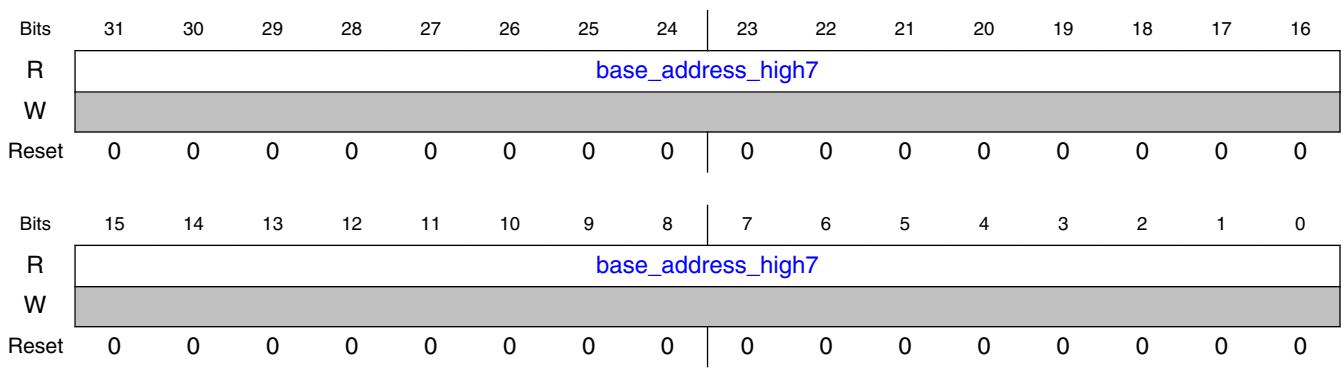
10.3.36.1 Offset

Register	Offset
region_setup_high_7	174h

10.3.36.2 Function

The region_setup_high register is none for region 0, for other regions it controls the base address [63:32] of region n.There are no usage constraints. Available in all configurations of the TZASC.

10.3.36.3 Diagram



10.3.36.4 Fields

Field	Function
31-0 base_address_high7	base_address_high7 Controls the base address [63:32] of region n. For region 0, this field is RO. The TZASC sets the base address of region 0 to 0x0. The TZASC only permits a region to start at address 0x0, or at a multiple of its region size. If you program a region size to be 8GB or more, then the TZASC might ignore certain bits depending on the region size.

10.3.37 Region Attributes 7 Register (region_attributes_7)

10.3.37.1 Offset

Register	Offset
region_attributes_7	178h

10.3.37.2 Function

The region_attributes_n register controls the permissions, region size, subregion disable, and region enable. There are no usage constraints. Available in all configurations of the TZASC.

Table 10-7. Region size

size<n> field	Size of region <n>	Base address [1] constraints
b000000-b001101	Reserved	-
b001110	32KB	-
b001111	64KB	Bit [15] must be zero
b010000	128KB	Bits [16:15] must be zero
b010001	256KB	Bits [17:15] must be zero
b010010	512KB	Bits [18:15] must be zero
b010011	1MB	Bits [19:15] must be zero
b010100	2MB	Bits [20:15] must be zero
b010101	4MB	Bits [21:15] must be zero
b010110	8MB	Bits [22:15] must be zero
b010111	16MB	Bits [23:15] must be zero
b011000	32MB	Bits [24:15] must be zero
b011001	64MB	Bits [25:15] must be zero
b011010	128MB	Bits [26:15] must be zero
b011011	256MB	Bits [27:15] must be zero
b011100	512MB	Bits [28:15] must be zero
b011101	1GB	Bits [29:15] must be zero
b011110	2GB	Bits [30:15] must be zero
b011111	4GB	Bits [31:15] must be zero
b100000	8GB	Bits [32:15] must be zero
b100001	16GB	Bits [33:15] must be zero
b100010	32GB	Bits [34:15] must be zero
b100011	64GB	Bits [35:15] must be zero

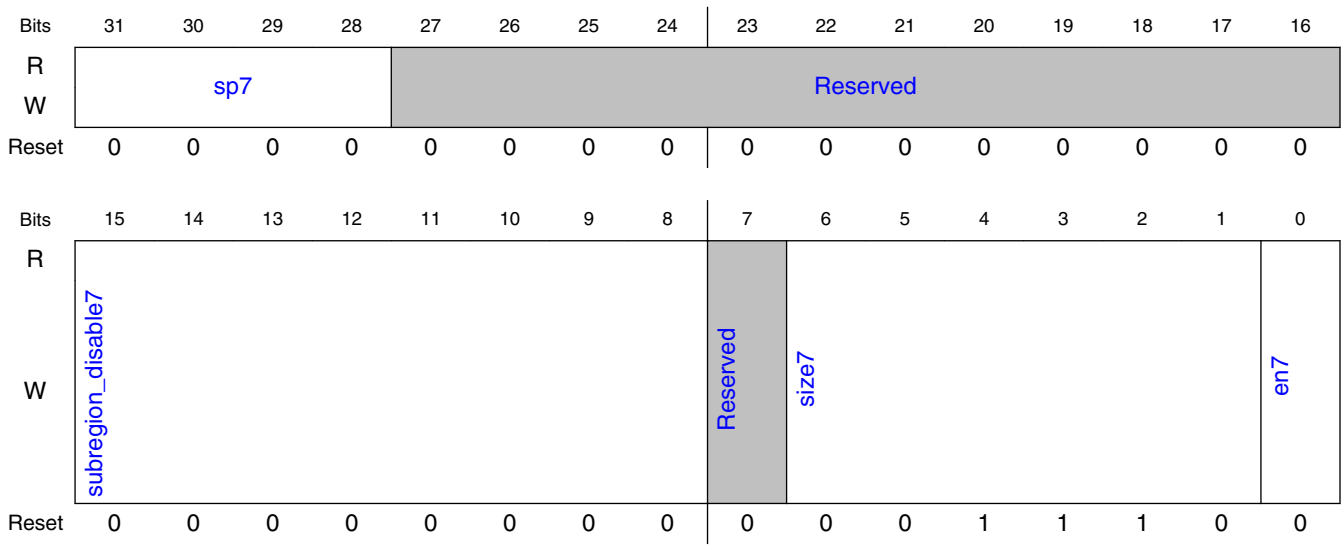
Table continues on the next page...

Table 10-7. Region size (continued)

b100100	128GB	Bits [36:15] must be zero
b100101	256GB	Bits [37:15] must be zero
b100110	512GB	Bits [38:15] must be zero
b100111	1TB	Bits [39:15] must be zero
b101000	2TB	Bits [40:15] must be zero
b101001	4TB	Bits [41:15] must be zero
b101010	8TB	Bits [42:15] must be zero
b101011	16TB	Bits [43:15] must be zero
b101100	32TB	Bits [44:15] must be zero
b101101	64TB	Bits [45:15] must be zero
b101110	128TB	Bits [46:15] must be zero
b101111	256TB	Bits [47:15] must be zero
b110000	512TB	Bits [48:15] must be zero
b110001	1PB	Bits [49:15] must be zero
b110010	2PB	Bits [50:15] must be zero
b110011	4PB	Bits [51:15] must be zero
b110100	8PB	Bits [52:15] must be zero
b110101	16PB	Bits [53:15] must be zero
b110110	32PB	Bits [54:15] must be zero
b110111	64PB	Bits [55:15] must be zero
b111000	128PB	Bits [56:15] must be zero
b111001	256PB	Bits [57:15] must be zero
b111010	512PB	Bits [58:15] must be zero
b111011	1EB	Bits [59:15] must be zero
b111100	2EB	Bits [60:15] must be zero
b111101	4EB	Bits [61:15] must be zero
b111110	8EB	Bits [62:15] must be zero
b111111	16EB	Bits [63:15] must be zero

[1]The region_setup_low_<n> Register and region_setup_high_<n> Register contain the base address. See Table 313 on page 318 and Table 314 on page 319.

10.3.37.3 Diagram



10.3.37.4 Fields

Field	Function
31-28 sp7	sp7 Permissions setting for region *n*. If an AXI transaction occurs to region *n*, the value in the sp*n* field controls whether the TZASC permits the transaction to proceed.
27-16 —	Reserved, Should be Zero (SBZ).
15-8 subregion_disable7	subregion_disable7 Regions are split into eight equal-sized sub-regions, and each bit enables the corresponding subregion to be disabled: Bit [15] = 1 Subregion 7 is disabled. Bit [14] = 1 Subregion 6 is disabled. Bit [13] = 1 Subregion 5 is disabled. Bit [12] = 1 Subregion 4 is disabled. Bit [11] = 1 Subregion 3 is disabled. Bit [10] = 1 Subregion 2 is disabled. Bit [9] = 1 Subregion 1 is disabled. Bit [8] = 1 Subregion 0 is disabled.
7 —	Reserved, Should be Zero (SBZ).
6-1 size7	size7

Table continues on the next page...

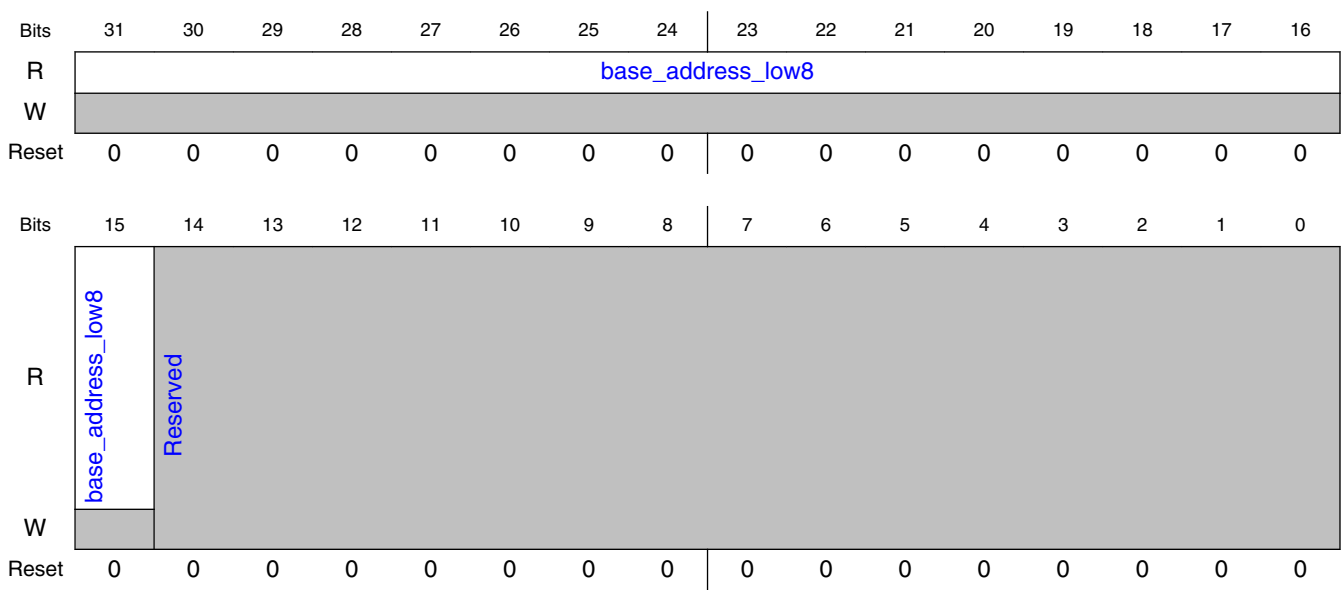
Field	Function
	Size of region *n*.The AXI address width, that is AXI_ADDRESS_MSB+1, controls the upper limit value of this field. The table below shows how the size *n* field controls the region size and what constraints, if any, the TZASC applies to the base address to ensure that a region starts on the boundary of the region size.
0 en7	en7 Enable for region *n*: 0 = region *n* is disabled 1 = region *n* is enabled.

10.3.38 Region Setup Low 8 Register (region_setup_low_8)

10.3.38.1 Offset

Register	Offset
region_setup_low_8	180h

10.3.38.2 Diagram



10.3.38.3 Fields

Field	Function
31-15 base_address_low8	base_address_low8 Controls the base address [31:15] of region n. For region 0, this field is Read Only (RO). The TZASC sets the base address of region 0 to 0x0. The TZASC only permits a region to start at address 0x0, or at a multiple of its region size. For example, if the size of a region is 512MB, and it is not at address 0x0, the only valid settings for this field are: b00100000000000000 b01000000000000000 b01100000000000000 b10000000000000000 b10100000000000000 b11000000000000000 b11100000000000000 If you attempt to set an inappropriate base address for the size of the region, the TZASC ignores certain bits depending on the region size. See Table on register size for more information.
14-0 —	Reserved, Should be Zero (SBZ).

10.3.39 Region Setup High 8 Register (region_setup_high_8)

10.3.39.1 Offset

Register	Offset
region_setup_high_8	184h

10.3.39.2 Function

The region_setup_high register is none for region 0, for other regions it controls the base address [63:32] of region n.There are no usage constraints. Available in all configurations of the TZASC.

10.3.39.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	base_address_high8															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	base_address_high8															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

10.3.39.4 Fields

Field	Function
31-0 base_address_high8	base_address_high8 Controls the base address [63:32] of region n. For region 0, this field is RO. The TZASC sets the base address of region 0 to 0x0. The TZASC only permits a region to start at address 0x0, or at a multiple of its region size. If you program a region size to be 8GB or more, then the TZASC might ignore certain bits depending on the region size.

10.3.40 Region Attributes 8 Register (region_attributes_8)

10.3.40.1 Offset

Register	Offset
region_attributes_8	188h

10.3.40.2 Function

The region_attributes_n register controls the permissions, region size, subregion disable, and region enable. There are no usage constraints. Available in all configurations of the TZASC.

Table 10-8. Region size

size<n> field	Size of region <n>	Base address [1] constraints
b000000-b001101	Reserved	-
b001110	32KB	-
b001111	64KB	Bit [15] must be zero
b010000	128KB	Bits [16:15] must be zero
b010001	256KB	Bits [17:15] must be zero
b010010	512KB	Bits [18:15] must be zero
b010011	1MB	Bits [19:15] must be zero
b010100	2MB	Bits [20:15] must be zero
b010101	4MB	Bits [21:15] must be zero
b010110	8MB	Bits [22:15] must be zero
b010111	16MB	Bits [23:15] must be zero
b011000	32MB	Bits [24:15] must be zero
b011001	64MB	Bits [25:15] must be zero
b011010	128MB	Bits [26:15] must be zero
b011011	256MB	Bits [27:15] must be zero
b011100	512MB	Bits [28:15] must be zero
b011101	1GB	Bits [29:15] must be zero
b011110	2GB	Bits [30:15] must be zero
b011111	4GB	Bits [31:15] must be zero
b100000	8GB	Bits [32:15] must be zero
b100001	16GB	Bits [33:15] must be zero
b100010	32GB	Bits [34:15] must be zero
b100011	64GB	Bits [35:15] must be zero
b100100	128GB	Bits [36:15] must be zero
b100101	256GB	Bits [37:15] must be zero
b100110	512GB	Bits [38:15] must be zero
b100111	1TB	Bits [39:15] must be zero
b101000	2TB	Bits [40:15] must be zero
b101001	4TB	Bits [41:15] must be zero
b101010	8TB	Bits [42:15] must be zero
b101011	16TB	Bits [43:15] must be zero
b101100	32TB	Bits [44:15] must be zero
b101101	64TB	Bits [45:15] must be zero
b101110	128TB	Bits [46:15] must be zero
b101111	256TB	Bits [47:15] must be zero
b110000	512TB	Bits [48:15] must be zero
b110001	1PB	Bits [49:15] must be zero
b110010	2PB	Bits [50:15] must be zero
b110011	4PB	Bits [51:15] must be zero

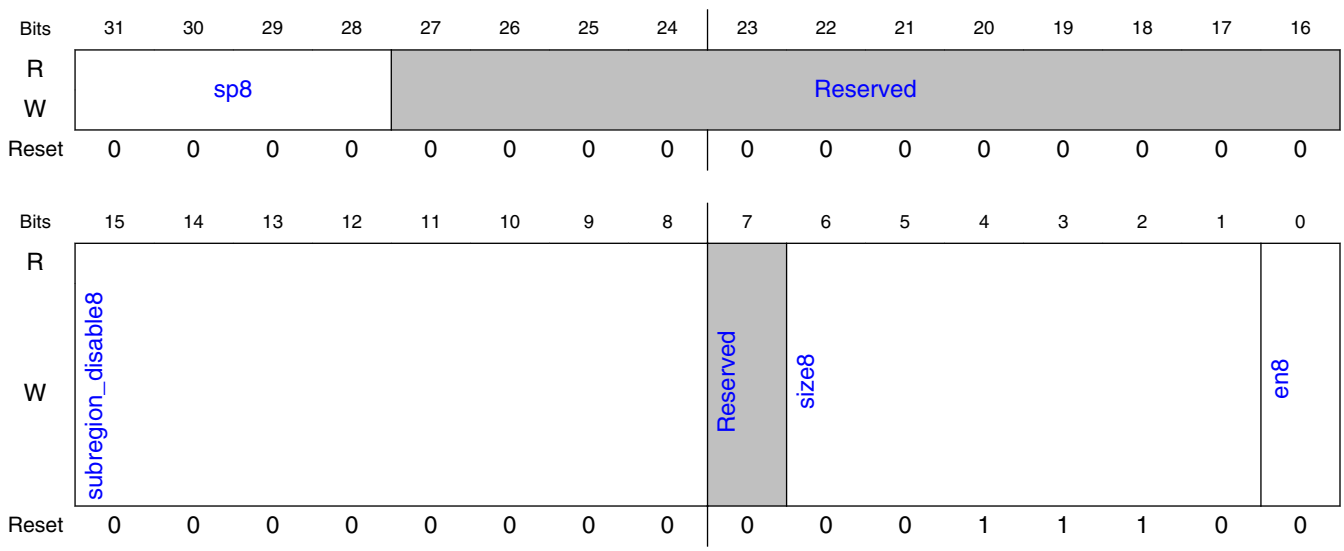
Table continues on the next page...

Table 10-8. Region size (continued)

b110100	8PB	Bits [52:15] must be zero
b110101	16PB	Bits [53:15] must be zero
b110110	32PB	Bits [54:15] must be zero
b110111	64PB	Bits [55:15] must be zero
b111000	128PB	Bits [56:15] must be zero
b111001	256PB	Bits [57:15] must be zero
b111010	512PB	Bits [58:15] must be zero
b111011	1EB	Bits [59:15] must be zero
b111100	2EB	Bits [60:15] must be zero
b111101	4EB	Bits [61:15] must be zero
b111110	8EB	Bits [62:15] must be zero
b111111	16EB	Bits [63:15] must be zero

[1]The region_setup_low_<n> Register and region_setup_high_<n> Register contain the base address. See Table 313 on page 318 and Table 314 on page 319.

10.3.40.3 Diagram



10.3.40.4 Fields

Field	Function
31-28 sp8	sp8

Table continues on the next page...

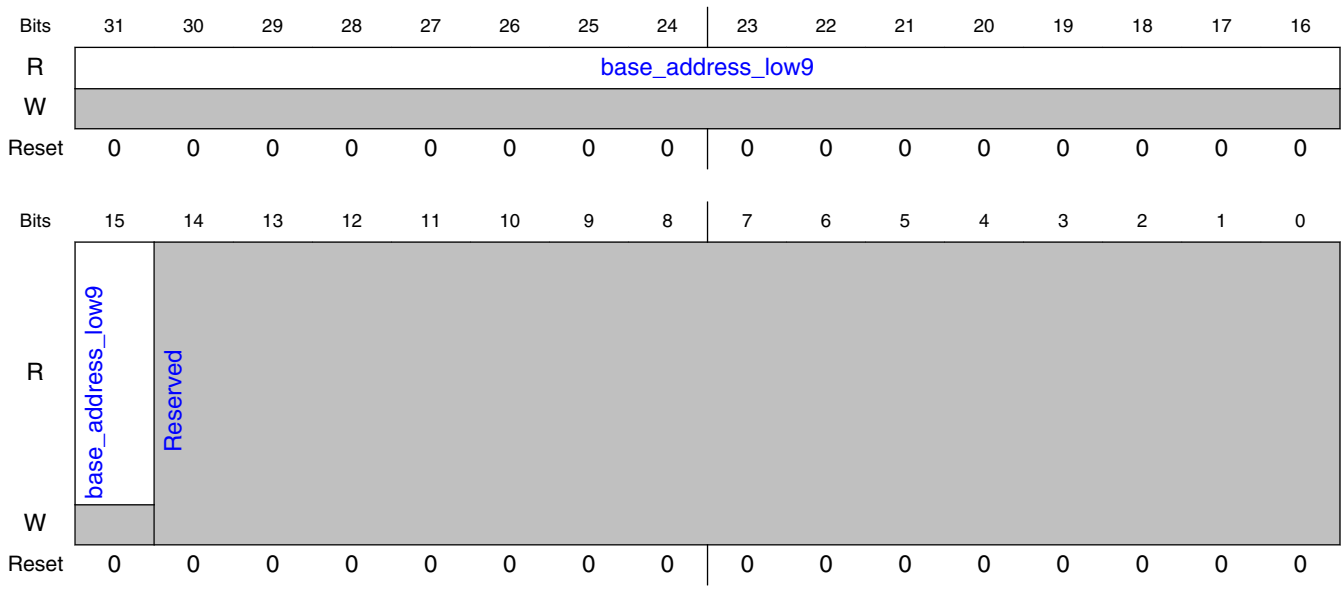
Field	Function
	Permissions setting for region *n*. If an AXI transaction occurs to region *n*, the value in the sp*n* field controls whether the TZASC permits the transaction to proceed.
27-16 —	Reserved, Should be Zero (SBZ).
15-8 subregion_disable8	subregion_disable8 Regions are split into eight equal-sized sub-regions, and each bit enables the corresponding subregion to be disabled: Bit [15] = 1 Subregion 7 is disabled. Bit [14] = 1 Subregion 6 is disabled. Bit [13] = 1 Subregion 5 is disabled. Bit [12] = 1 Subregion 4 is disabled. Bit [11] = 1 Subregion 3 is disabled. Bit [10] = 1 Subregion 2 is disabled. Bit [9] = 1 Subregion 1 is disabled. Bit [8] = 1 Subregion 0 is disabled.
7 —	Reserved, Should be Zero (SBZ).
6-1 size8	size8 Size of region *n*.The AXI address width, that is AXI_ADDRESS_MSB+1, controls the upper limit value of this field. The table below shows how the size *n* field controls the region size and what constraints, if any, the TZASC applies to the base address to ensure that a region starts on the boundary of the region size.
0 en8	en8 Enable for region *n*: 0 = region *n* is disabled 1 = region *n* is enabled.

10.3.41 Region Setup Low 9 Register (region_setup_low_9)

10.3.41.1 Offset

Register	Offset
region_setup_low_9	190h

10.3.41.2 Diagram



10.3.41.3 Fields

Field	Function
31-15 base_address_low9	base_address_low9 Controls the base address [31:15] of region n. For region 0, this field is Read Only (RO). The TZASC sets the base address of region 0 to 0x0. The TZASC only permits a region to start at address 0x0, or at a multiple of its region size. For example, if the size of a region is 512MB, and it is not at address 0x0, the only valid settings for this field are: b00100000000000000 b01000000000000000 b01100000000000000 b10000000000000000 b10100000000000000 b11000000000000000 b11100000000000000 If you attempt to set an inappropriate base address for the size of the region, the TZASC ignores certain bits depending on the region size. See Table on register size for more information.
14-0 —	Reserved, Should be Zero (SBZ).

10.3.42 Region Setup High 9 Register (region_setup_high_9)

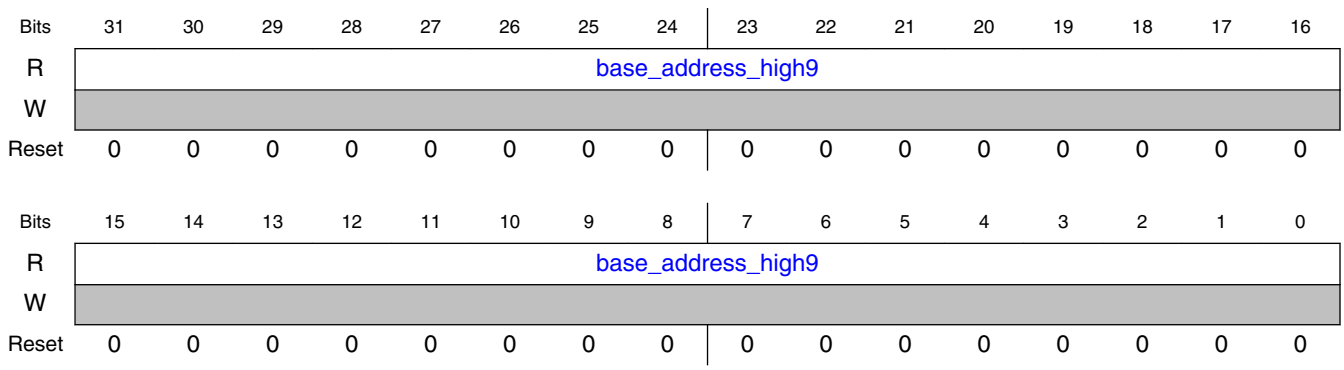
10.3.42.1 Offset

Register	Offset
region_setup_high_9	194h

10.3.42.2 Function

The region_setup_high register is none for region 0, for other regions it controls the base address [63:32] of region n.There are no usage constraints. Available in all configurations of the TZASC.

10.3.42.3 Diagram



10.3.42.4 Fields

Field	Function
31-0 base_address_high9	base_address_high9 Controls the base address [63:32] of region n. For region 0, this field is RO. The TZASC sets the base address of region 0 to 0x0. The TZASC only permits a region to start at address 0x0, or at a multiple of its region size. If you program a region size to be 8GB or more, then the TZASC might ignore certain bits depending on the region size.

10.3.43 Region Attributes 9 Register (region_attributes_9)

10.3.43.1 Offset

Register	Offset
region_attributes_9	198h

10.3.43.2 Function

The region_attributes_n register controls the permissions, region size, subregion disable, and region enable. There are no usage constraints. Available in all configurations of the TZASC.

Table 10-9. Region size

size<n> field	Size of region <n>	Base address [1] constraints
b000000-b001101	Reserved	-
b001110	32KB	-
b001111	64KB	Bit [15] must be zero
b010000	128KB	Bits [16:15] must be zero
b010001	256KB	Bits [17:15] must be zero
b010010	512KB	Bits [18:15] must be zero
b010011	1MB	Bits [19:15] must be zero
b010100	2MB	Bits [20:15] must be zero
b010101	4MB	Bits [21:15] must be zero
b010110	8MB	Bits [22:15] must be zero
b010111	16MB	Bits [23:15] must be zero
b011000	32MB	Bits [24:15] must be zero
b011001	64MB	Bits [25:15] must be zero
b011010	128MB	Bits [26:15] must be zero
b011011	256MB	Bits [27:15] must be zero
b011100	512MB	Bits [28:15] must be zero
b011101	1GB	Bits [29:15] must be zero
b011110	2GB	Bits [30:15] must be zero
b011111	4GB	Bits [31:15] must be zero
b100000	8GB	Bits [32:15] must be zero
b100001	16GB	Bits [33:15] must be zero
b100010	32GB	Bits [34:15] must be zero
b100011	64GB	Bits [35:15] must be zero

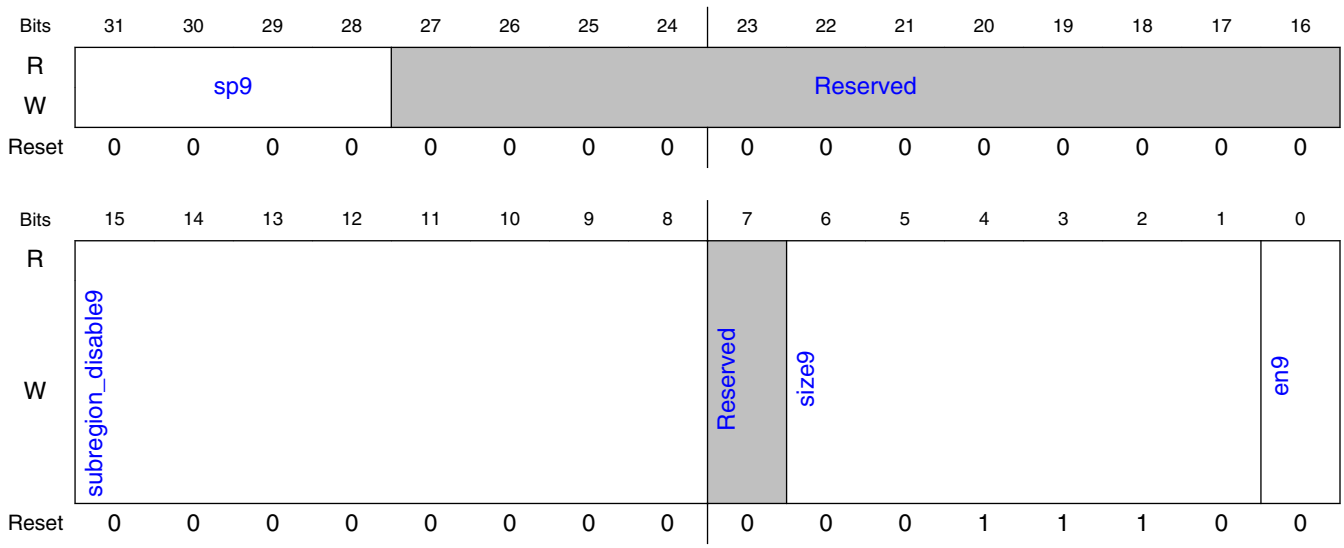
Table continues on the next page...

Table 10-9. Region size (continued)

b100100	128GB	Bits [36:15] must be zero
b100101	256GB	Bits [37:15] must be zero
b100110	512GB	Bits [38:15] must be zero
b100111	1TB	Bits [39:15] must be zero
b101000	2TB	Bits [40:15] must be zero
b101001	4TB	Bits [41:15] must be zero
b101010	8TB	Bits [42:15] must be zero
b101011	16TB	Bits [43:15] must be zero
b101100	32TB	Bits [44:15] must be zero
b101101	64TB	Bits [45:15] must be zero
b101110	128TB	Bits [46:15] must be zero
b101111	256TB	Bits [47:15] must be zero
b110000	512TB	Bits [48:15] must be zero
b110001	1PB	Bits [49:15] must be zero
b110010	2PB	Bits [50:15] must be zero
b110011	4PB	Bits [51:15] must be zero
b110100	8PB	Bits [52:15] must be zero
b110101	16PB	Bits [53:15] must be zero
b110110	32PB	Bits [54:15] must be zero
b110111	64PB	Bits [55:15] must be zero
b111000	128PB	Bits [56:15] must be zero
b111001	256PB	Bits [57:15] must be zero
b111010	512PB	Bits [58:15] must be zero
b111011	1EB	Bits [59:15] must be zero
b111100	2EB	Bits [60:15] must be zero
b111101	4EB	Bits [61:15] must be zero
b111110	8EB	Bits [62:15] must be zero
b111111	16EB	Bits [63:15] must be zero

[1]The region_setup_low_<n> Register and region_setup_high_<n> Register contain the base address. See Table 313 on page 318 and Table 314 on page 319.

10.3.43.3 Diagram



10.3.43.4 Fields

Field	Function
31-28 sp9	sp9 Permissions setting for region *n*. If an AXI transaction occurs to region *n*, the value in the sp*n* field controls whether the TZASC permits the transaction to proceed.
27-16 —	Reserved, Should be Zero (SBZ).
15-8 subregion_disable9	subregion_disable9 Regions are split into eight equal-sized sub-regions, and each bit enables the corresponding subregion to be disabled: Bit [15] = 1 Subregion 7 is disabled. Bit [14] = 1 Subregion 6 is disabled. Bit [13] = 1 Subregion 5 is disabled. Bit [12] = 1 Subregion 4 is disabled. Bit [11] = 1 Subregion 3 is disabled. Bit [10] = 1 Subregion 2 is disabled. Bit [9] = 1 Subregion 1 is disabled. Bit [8] = 1 Subregion 0 is disabled.
7 —	Reserved, Should be Zero (SBZ).
6-1 size9	size9

Table continues on the next page...

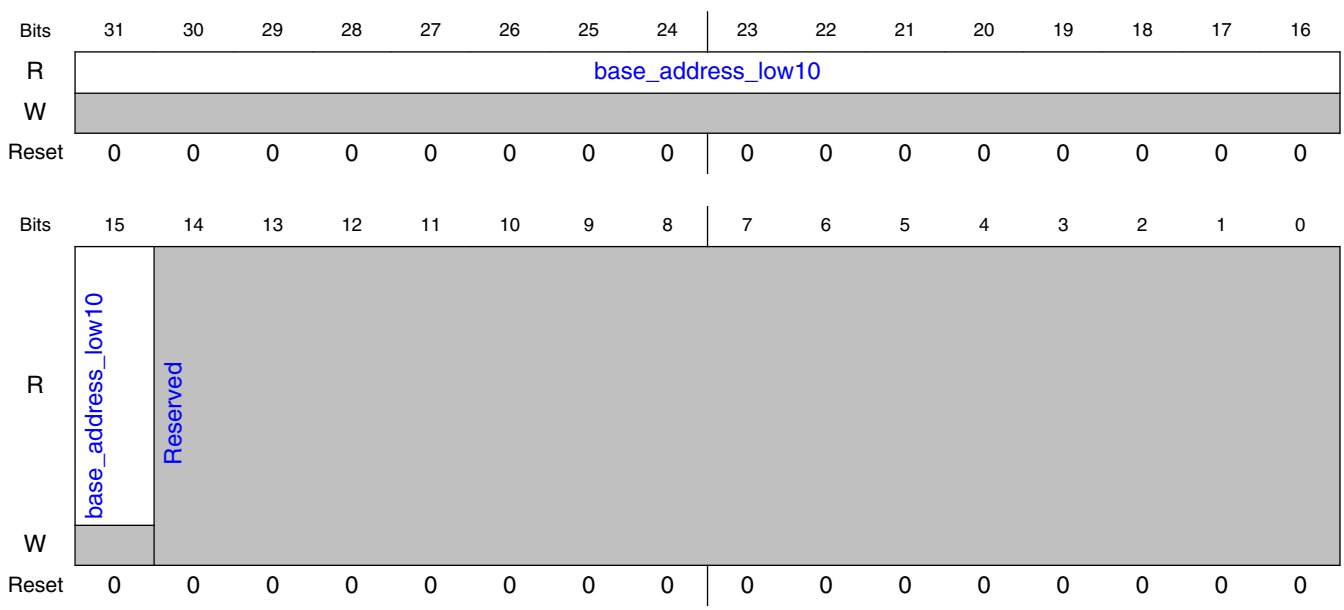
Field	Function
	Size of region *n*.The AXI address width, that is AXI_ADDRESS_MSB+1, controls the upper limit value of this field. The table below shows how the size *n* field controls the region size and what constraints, if any, the TZASC applies to the base address to ensure that a region starts on the boundary of the region size.
0 en9	en9 Enable for region *n*: 0 = region *n* is disabled 1 = region *n* is enabled.

10.3.44 Region Setup Low 10 Register (region_setup_low_10)

10.3.44.1 Offset

Register	Offset
region_setup_low_10	1A0h

10.3.44.2 Diagram



10.3.44.3 Fields

Field	Function
31-15 base_address_low10	base_address_low10 Controls the base address [31:15] of region n. For region 0, this field is Read Only (RO). The TZASC sets the base address of region 0 to 0x0. The TZASC only permits a region to start at address 0x0, or at a multiple of its region size. For example, if the size of a region is 512MB, and it is not at address 0x0, the only valid settings for this field are: b00100000000000000 b01000000000000000 b01100000000000000 b10000000000000000 b10100000000000000 b11000000000000000 b11100000000000000 If you attempt to set an inappropriate base address for the size of the region, the TZASC ignores certain bits depending on the region size. See Table on register size for more information.
14-0 —	Reserved, Should be Zero (SBZ).

10.3.45 Region Setup High 10 Register (region_setup_high_10)

10.3.45.1 Offset

Register	Offset
region_setup_high_10	1A4h

10.3.45.2 Function

The region_setup_high register is none for region 0, for other regions it controls the base address [63:32] of region n.There are no usage constraints. Available in all configurations of the TZASC.

10.3.45.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	base_address_high10															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	base_address_high10															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

10.3.45.4 Fields

Field	Function
31-0 base_address_high10	base_address_high10 Controls the base address [63:32] of region n. For region 0, this field is RO. The TZASC sets the base address of region 0 to 0x0. The TZASC only permits a region to start at address 0x0, or at a multiple of its region size. If you program a region size to be 8GB or more, then the TZASC might ignore certain bits depending on the region size.

10.3.46 Region Attributes 10 Register (region_attributes_10)

10.3.46.1 Offset

Register	Offset
region_attributes_10	1A8h

10.3.46.2 Function

The region_attributes_n register controls the permissions, region size, subregion disable, and region enable. There are no usage constraints. Available in all configurations of the TZASC.

Table 10-10. Region size

size<n> field	Size of region <n>	Base address [1] constraints
b000000-b001101	Reserved	-
b001110	32KB	-
b001111	64KB	Bit [15] must be zero
b010000	128KB	Bits [16:15] must be zero
b010001	256KB	Bits [17:15] must be zero
b010010	512KB	Bits [18:15] must be zero
b010011	1MB	Bits [19:15] must be zero
b010100	2MB	Bits [20:15] must be zero
b010101	4MB	Bits [21:15] must be zero
b010110	8MB	Bits [22:15] must be zero
b010111	16MB	Bits [23:15] must be zero
b011000	32MB	Bits [24:15] must be zero
b011001	64MB	Bits [25:15] must be zero
b011010	128MB	Bits [26:15] must be zero
b011011	256MB	Bits [27:15] must be zero
b011100	512MB	Bits [28:15] must be zero
b011101	1GB	Bits [29:15] must be zero
b011110	2GB	Bits [30:15] must be zero
b011111	4GB	Bits [31:15] must be zero
b100000	8GB	Bits [32:15] must be zero
b100001	16GB	Bits [33:15] must be zero
b100010	32GB	Bits [34:15] must be zero
b100011	64GB	Bits [35:15] must be zero
b100100	128GB	Bits [36:15] must be zero
b100101	256GB	Bits [37:15] must be zero
b100110	512GB	Bits [38:15] must be zero
b100111	1TB	Bits [39:15] must be zero
b101000	2TB	Bits [40:15] must be zero
b101001	4TB	Bits [41:15] must be zero
b101010	8TB	Bits [42:15] must be zero
b101011	16TB	Bits [43:15] must be zero
b101100	32TB	Bits [44:15] must be zero
b101101	64TB	Bits [45:15] must be zero
b101110	128TB	Bits [46:15] must be zero
b101111	256TB	Bits [47:15] must be zero
b110000	512TB	Bits [48:15] must be zero
b110001	1PB	Bits [49:15] must be zero
b110010	2PB	Bits [50:15] must be zero
b110011	4PB	Bits [51:15] must be zero

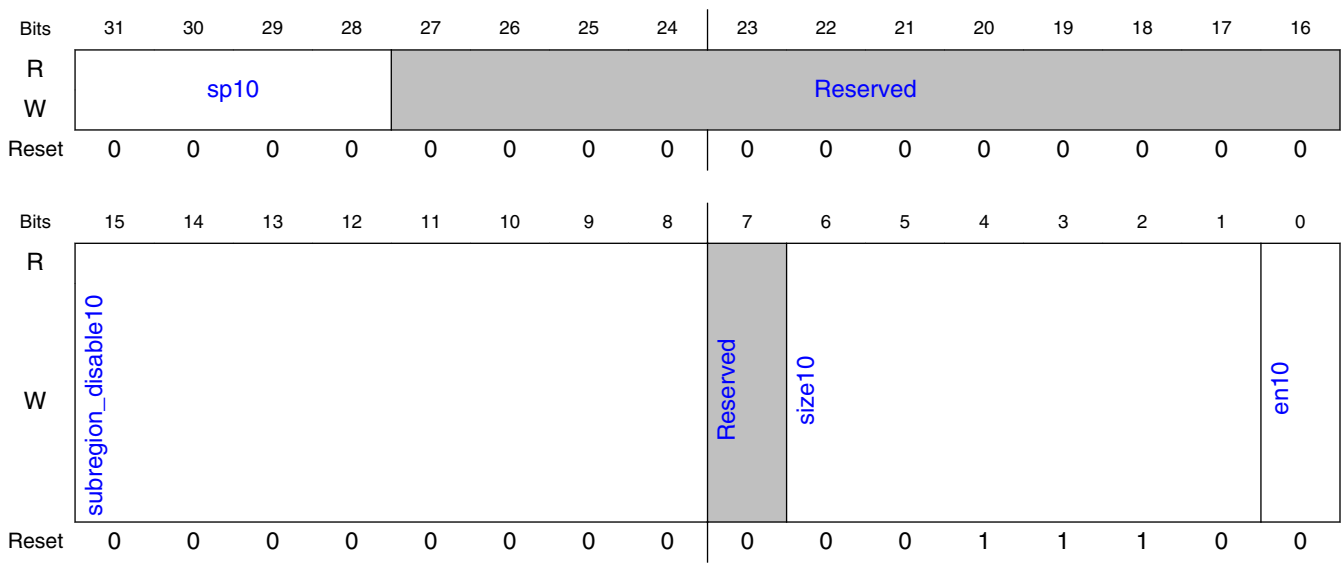
Table continues on the next page...

Table 10-10. Region size (continued)

b110100	8PB	Bits [52:15] must be zero
b110101	16PB	Bits [53:15] must be zero
b110110	32PB	Bits [54:15] must be zero
b110111	64PB	Bits [55:15] must be zero
b111000	128PB	Bits [56:15] must be zero
b111001	256PB	Bits [57:15] must be zero
b111010	512PB	Bits [58:15] must be zero
b111011	1EB	Bits [59:15] must be zero
b111100	2EB	Bits [60:15] must be zero
b111101	4EB	Bits [61:15] must be zero
b111110	8EB	Bits [62:15] must be zero
b111111	16EB	Bits [63:15] must be zero

[1]The region_setup_low_<n> Register and region_setup_high_<n> Register contain the base address. See Table 313 on page 318 and Table 314 on page 319.

10.3.46.3 Diagram



10.3.46.4 Fields

Field	Function
31-28 sp10	sp10

Table continues on the next page...

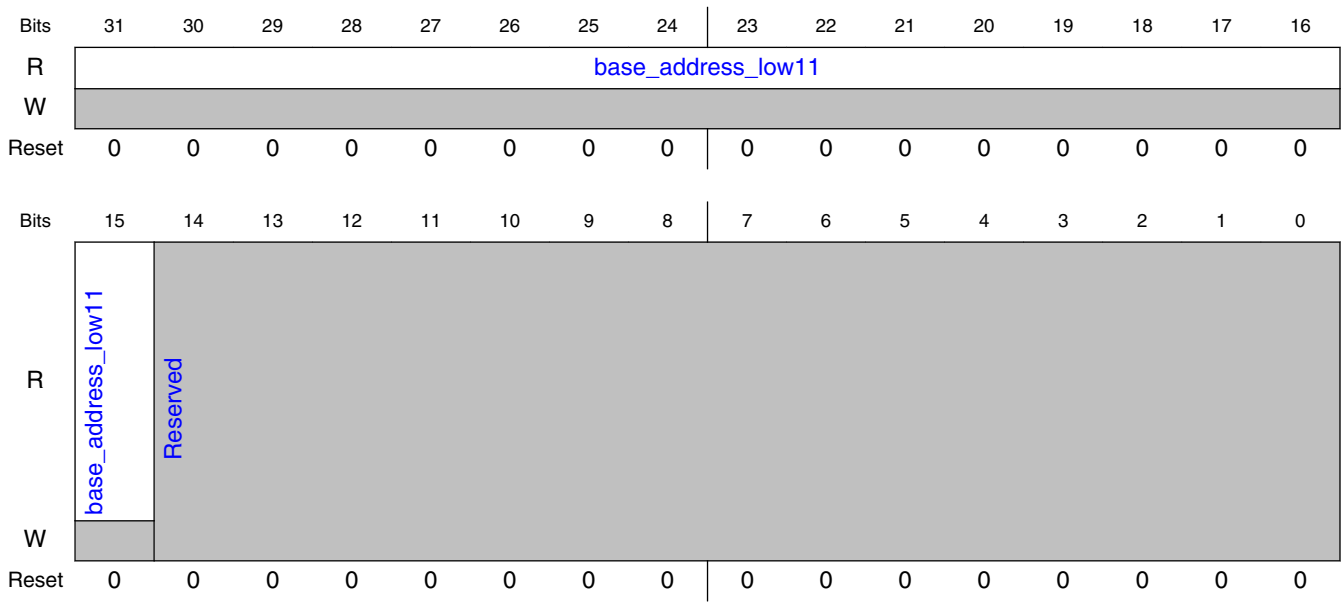
Field	Function
	Permissions setting for region *n*. If an AXI transaction occurs to region *n*, the value in the sp*n* field controls whether the TZASC permits the transaction to proceed.
27-16 —	Reserved, Should be Zero (SBZ).
15-8 subregion_disable10	subregion_disable10 Regions are split into eight equal-sized sub-regions, and each bit enables the corresponding subregion to be disabled: Bit [15] = 1 Subregion 7 is disabled. Bit [14] = 1 Subregion 6 is disabled. Bit [13] = 1 Subregion 5 is disabled. Bit [12] = 1 Subregion 4 is disabled. Bit [11] = 1 Subregion 3 is disabled. Bit [10] = 1 Subregion 2 is disabled. Bit [9] = 1 Subregion 1 is disabled. Bit [8] = 1 Subregion 0 is disabled.
7 —	Reserved, Should be Zero (SBZ).
6-1 size10	size10 Size of region *n*.The AXI address width, that is AXI_ADDRESS_MSB+1, controls the upper limit value of this field. The table below shows how the size *n* field controls the region size and what constraints, if any, the TZASC applies to the base address to ensure that a region starts on the boundary of the region size.
0 en10	en10 Enable for region *n*: 0 = region *n* is disabled 1 = region *n* is enabled.

10.3.47 Region Setup Low 11 Register (region_setup_low_11)

10.3.47.1 Offset

Register	Offset
region_setup_low_11	1B0h

10.3.47.2 Diagram



10.3.47.3 Fields

Field	Function
31-15 base_address_low11	base_address_low11 Controls the base address [31:15] of region n. For region 0, this field is Read Only (RO). The TZASC sets the base address of region 0 to 0x0. The TZASC only permits a region to start at address 0x0, or at a multiple of its region size. For example, if the size of a region is 512MB, and it is not at address 0x0, the only valid settings for this field are: b00100000000000000 b01000000000000000 b01100000000000000 b10000000000000000 b10100000000000000 b11000000000000000 b11100000000000000 If you attempt to set an inappropriate base address for the size of the region, the TZASC ignores certain bits depending on the region size. See Table on register size for more information.
14-0 —	Reserved, Should be Zero (SBZ).

10.3.48 Region Setup High 11 Register (region_setup_high_11)

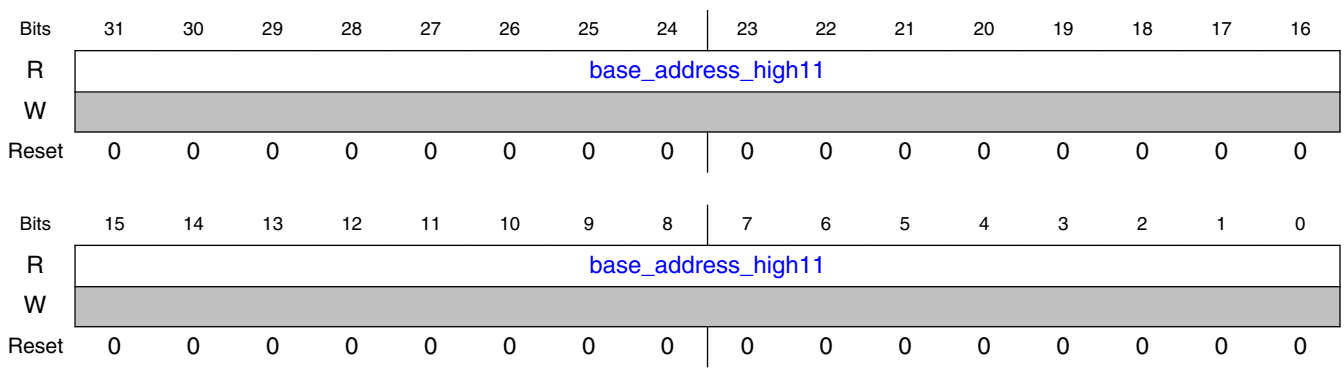
10.3.48.1 Offset

Register	Offset
region_setup_high_11	1B4h

10.3.48.2 Function

The region_setup_high register is none for region 0, for other regions it controls the base address [63:32] of region n.There are no usage constraints. Available in all configurations of the TZASC.

10.3.48.3 Diagram



10.3.48.4 Fields

Field	Function
31-0 base_address_high11	base_address_high11 Controls the base address [63:32] of region n. For region 0, this field is RO. The TZASC sets the base address of region 0 to 0x0. The TZASC only permits a region to start at address 0x0, or at a multiple of its region size. If you program a region size to be 8GB or more, then the TZASC might ignore certain bits depending on the region size.

10.3.49 Region Attributes 11 Register (region_attributes_11)

10.3.49.1 Offset

Register	Offset
region_attributes_11	1B8h

10.3.49.2 Function

The region_attributes_n register controls the permissions, region size, subregion disable, and region enable. There are no usage constraints. Available in all configurations of the TZASC.

Table 10-11. Region size

size<n> field	Size of region <n>	Base address [1] constraints
b000000-b001101	Reserved	-
b001110	32KB	-
b001111	64KB	Bit [15] must be zero
b010000	128KB	Bits [16:15] must be zero
b010001	256KB	Bits [17:15] must be zero
b010010	512KB	Bits [18:15] must be zero
b010011	1MB	Bits [19:15] must be zero
b010100	2MB	Bits [20:15] must be zero
b010101	4MB	Bits [21:15] must be zero
b010110	8MB	Bits [22:15] must be zero
b010111	16MB	Bits [23:15] must be zero
b011000	32MB	Bits [24:15] must be zero
b011001	64MB	Bits [25:15] must be zero
b011010	128MB	Bits [26:15] must be zero
b011011	256MB	Bits [27:15] must be zero
b011100	512MB	Bits [28:15] must be zero
b011101	1GB	Bits [29:15] must be zero
b011110	2GB	Bits [30:15] must be zero
b011111	4GB	Bits [31:15] must be zero
b100000	8GB	Bits [32:15] must be zero
b100001	16GB	Bits [33:15] must be zero
b100010	32GB	Bits [34:15] must be zero
b100011	64GB	Bits [35:15] must be zero

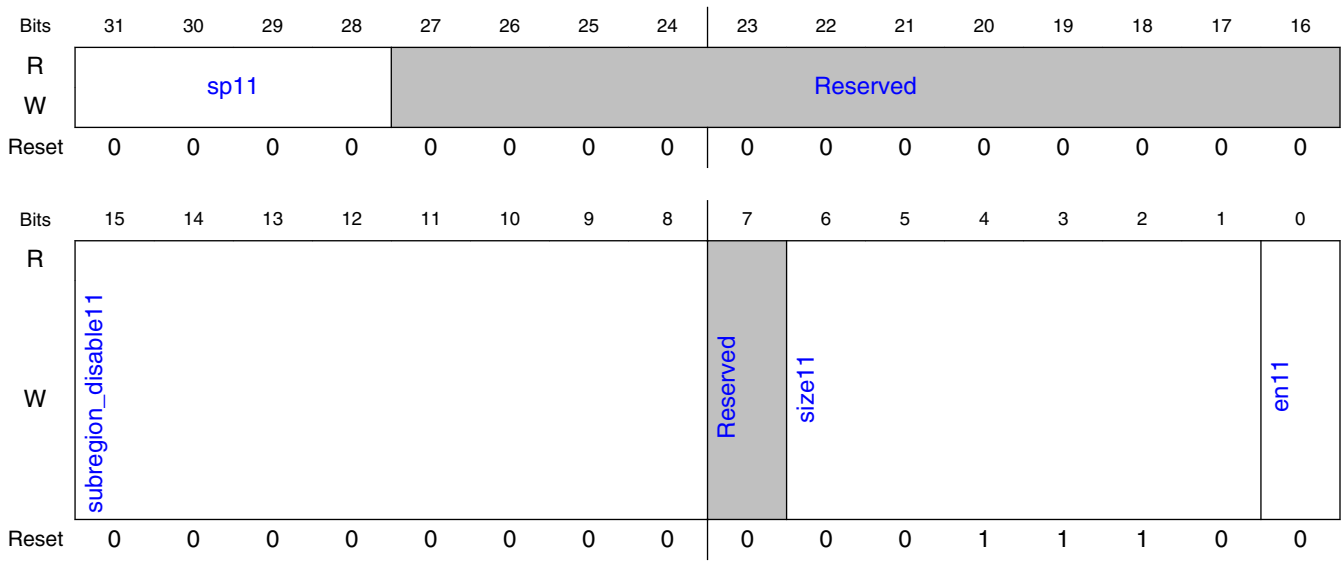
Table continues on the next page...

Table 10-11. Region size (continued)

b100100	128GB	Bits [36:15] must be zero
b100101	256GB	Bits [37:15] must be zero
b100110	512GB	Bits [38:15] must be zero
b100111	1TB	Bits [39:15] must be zero
b101000	2TB	Bits [40:15] must be zero
b101001	4TB	Bits [41:15] must be zero
b101010	8TB	Bits [42:15] must be zero
b101011	16TB	Bits [43:15] must be zero
b101100	32TB	Bits [44:15] must be zero
b101101	64TB	Bits [45:15] must be zero
b101110	128TB	Bits [46:15] must be zero
b101111	256TB	Bits [47:15] must be zero
b110000	512TB	Bits [48:15] must be zero
b110001	1PB	Bits [49:15] must be zero
b110010	2PB	Bits [50:15] must be zero
b110011	4PB	Bits [51:15] must be zero
b110100	8PB	Bits [52:15] must be zero
b110101	16PB	Bits [53:15] must be zero
b110110	32PB	Bits [54:15] must be zero
b110111	64PB	Bits [55:15] must be zero
b111000	128PB	Bits [56:15] must be zero
b111001	256PB	Bits [57:15] must be zero
b111010	512PB	Bits [58:15] must be zero
b111011	1EB	Bits [59:15] must be zero
b111100	2EB	Bits [60:15] must be zero
b111101	4EB	Bits [61:15] must be zero
b111110	8EB	Bits [62:15] must be zero
b111111	16EB	Bits [63:15] must be zero

[1]The region_setup_low_<n> Register and region_setup_high_<n> Register contain the base address. See Table 313 on page 318 and Table 314 on page 319.

10.3.49.3 Diagram



10.3.49.4 Fields

Field	Function
31-28 sp11	sp11 Permissions setting for region *n*. If an AXI transaction occurs to region *n*, the value in the sp*n* field controls whether the TZASC permits the transaction to proceed.
27-16 —	Reserved, Should be Zero (SBZ).
15-8 subregion_disable11	subregion_disable11 Regions are split into eight equal-sized sub-regions, and each bit enables the corresponding subregion to be disabled: Bit [15] = 1 Subregion 7 is disabled. Bit [14] = 1 Subregion 6 is disabled. Bit [13] = 1 Subregion 5 is disabled. Bit [12] = 1 Subregion 4 is disabled. Bit [11] = 1 Subregion 3 is disabled. Bit [10] = 1 Subregion 2 is disabled. Bit [9] = 1 Subregion 1 is disabled. Bit [8] = 1 Subregion 0 is disabled.
7 —	Reserved, Should be Zero (SBZ).
6-1	size11

Table continues on the next page...

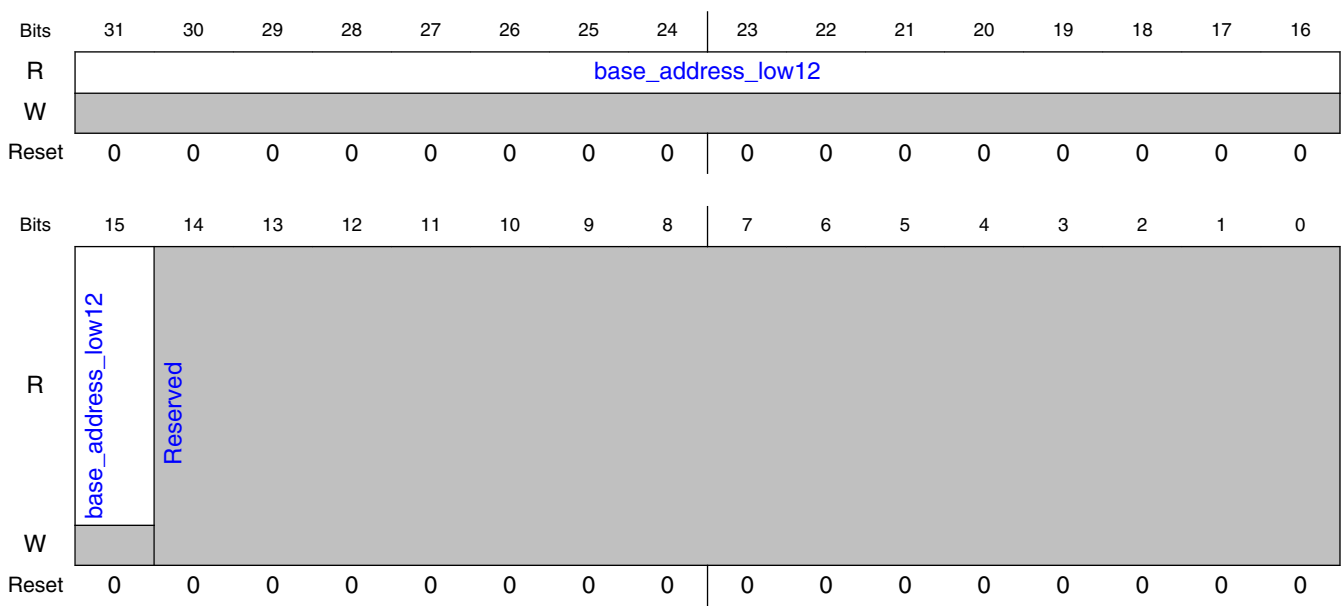
Field	Function
size11	Size of region *n*.The AXI address width, that is AXI_ADDRESS_MSB+1, controls the upper limit value of this field. The table below shows how the size *n* field controls the region size and what constraints, if any, the TZASC applies to the base address to ensure that a region starts on the boundary of the region size.
0 en11	en11 Enable for region *n*: 0 = region *n* is disabled 1 = region *n* is enabled.

10.3.50 Region Setup Low 12 Register (region_setup_low_12)

10.3.50.1 Offset

Register	Offset
region_setup_low_12	1C0h

10.3.50.2 Diagram



10.3.50.3 Fields

Field	Function
31-15 base_address_low12	base_address_low12 Controls the base address [31:15] of region n. For region 0, this field is Read Only (RO). The TZASC sets the base address of region 0 to 0x0. The TZASC only permits a region to start at address 0x0, or at a multiple of its region size. For example, if the size of a region is 512MB, and it is not at address 0x0, the only valid settings for this field are: b00100000000000000 b01000000000000000 b01100000000000000 b10000000000000000 b10100000000000000 b11000000000000000 b11100000000000000 If you attempt to set an inappropriate base address for the size of the region, the TZASC ignores certain bits depending on the region size. See Table on register size for more information.
14-0 —	Reserved, Should be Zero (SBZ).

10.3.51 Region Setup High 12 Register (region_setup_high_12)

10.3.51.1 Offset

Register	Offset
region_setup_high_12	1C4h

10.3.51.2 Function

The region_setup_high register is none for region 0, for other regions it controls the base address [63:32] of region n.There are no usage constraints. Available in all configurations of the TZASC.

10.3.51.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	base_address_high12															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	base_address_high12															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

10.3.51.4 Fields

Field	Function
31-0 base_address_high12	base_address_high12 Controls the base address [63:32] of region n. For region 0, this field is RO. The TZASC sets the base address of region 0 to 0x0. The TZASC only permits a region to start at address 0x0, or at a multiple of its region size. If you program a region size to be 8GB or more, then the TZASC might ignore certain bits depending on the region size.

10.3.52 Region Attributes 12 Register (region_attributes_12)

10.3.52.1 Offset

Register	Offset
region_attributes_12	1C8h

10.3.52.2 Function

The region_attributes_n register controls the permissions, region size, subregion disable, and region enable. There are no usage constraints. Available in all configurations of the TZASC.

Table 10-12. Region size

size<n> field	Size of region <n>	Base address [1] constraints
b000000-b001101	Reserved	-
b001110	32KB	-
b001111	64KB	Bit [15] must be zero
b010000	128KB	Bits [16:15] must be zero
b010001	256KB	Bits [17:15] must be zero
b010010	512KB	Bits [18:15] must be zero
b010011	1MB	Bits [19:15] must be zero
b010100	2MB	Bits [20:15] must be zero
b010101	4MB	Bits [21:15] must be zero
b010110	8MB	Bits [22:15] must be zero
b010111	16MB	Bits [23:15] must be zero
b011000	32MB	Bits [24:15] must be zero
b011001	64MB	Bits [25:15] must be zero
b011010	128MB	Bits [26:15] must be zero
b011011	256MB	Bits [27:15] must be zero
b011100	512MB	Bits [28:15] must be zero
b011101	1GB	Bits [29:15] must be zero
b011110	2GB	Bits [30:15] must be zero
b011111	4GB	Bits [31:15] must be zero
b100000	8GB	Bits [32:15] must be zero
b100001	16GB	Bits [33:15] must be zero
b100010	32GB	Bits [34:15] must be zero
b100011	64GB	Bits [35:15] must be zero
b100100	128GB	Bits [36:15] must be zero
b100101	256GB	Bits [37:15] must be zero
b100110	512GB	Bits [38:15] must be zero
b100111	1TB	Bits [39:15] must be zero
b101000	2TB	Bits [40:15] must be zero
b101001	4TB	Bits [41:15] must be zero
b101010	8TB	Bits [42:15] must be zero
b101011	16TB	Bits [43:15] must be zero
b101100	32TB	Bits [44:15] must be zero
b101101	64TB	Bits [45:15] must be zero
b101110	128TB	Bits [46:15] must be zero
b101111	256TB	Bits [47:15] must be zero
b110000	512TB	Bits [48:15] must be zero
b110001	1PB	Bits [49:15] must be zero
b110010	2PB	Bits [50:15] must be zero
b110011	4PB	Bits [51:15] must be zero

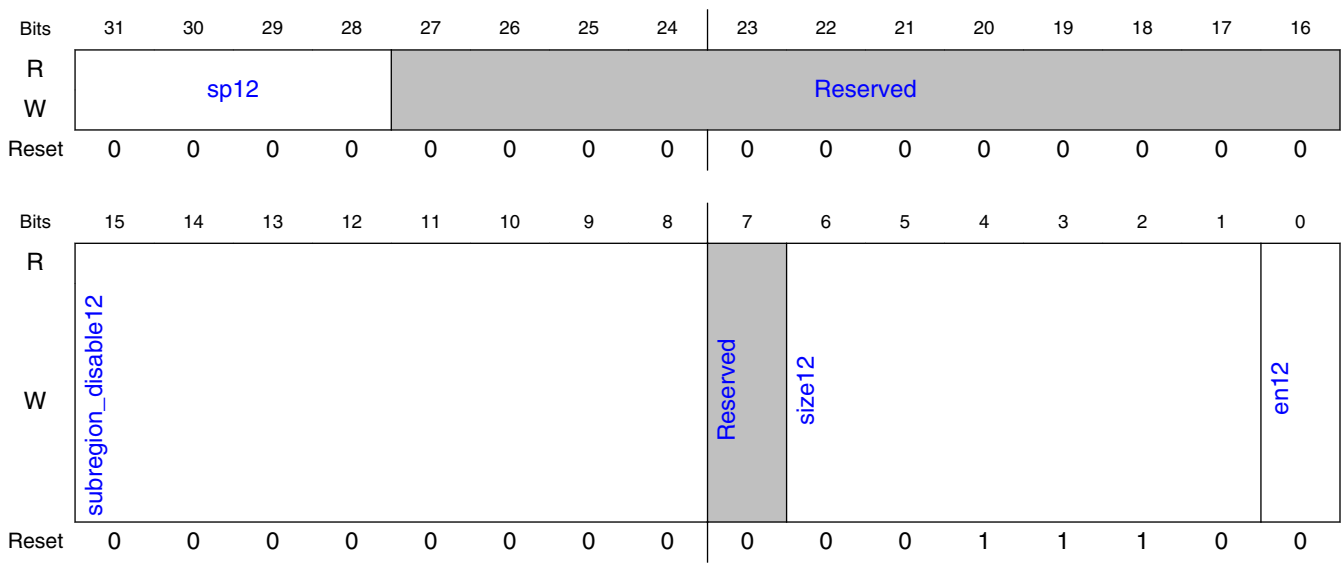
Table continues on the next page...

Table 10-12. Region size (continued)

b110100	8PB	Bits [52:15] must be zero
b110101	16PB	Bits [53:15] must be zero
b110110	32PB	Bits [54:15] must be zero
b110111	64PB	Bits [55:15] must be zero
b111000	128PB	Bits [56:15] must be zero
b111001	256PB	Bits [57:15] must be zero
b111010	512PB	Bits [58:15] must be zero
b111011	1EB	Bits [59:15] must be zero
b111100	2EB	Bits [60:15] must be zero
b111101	4EB	Bits [61:15] must be zero
b111110	8EB	Bits [62:15] must be zero
b111111	16EB	Bits [63:15] must be zero

[1]The region_setup_low_<n> Register and region_setup_high_<n> Register contain the base address. See Table 313 on page 318 and Table 314 on page 319.

10.3.52.3 Diagram



10.3.52.4 Fields

Field	Function
31-28 sp12	sp12

Table continues on the next page...

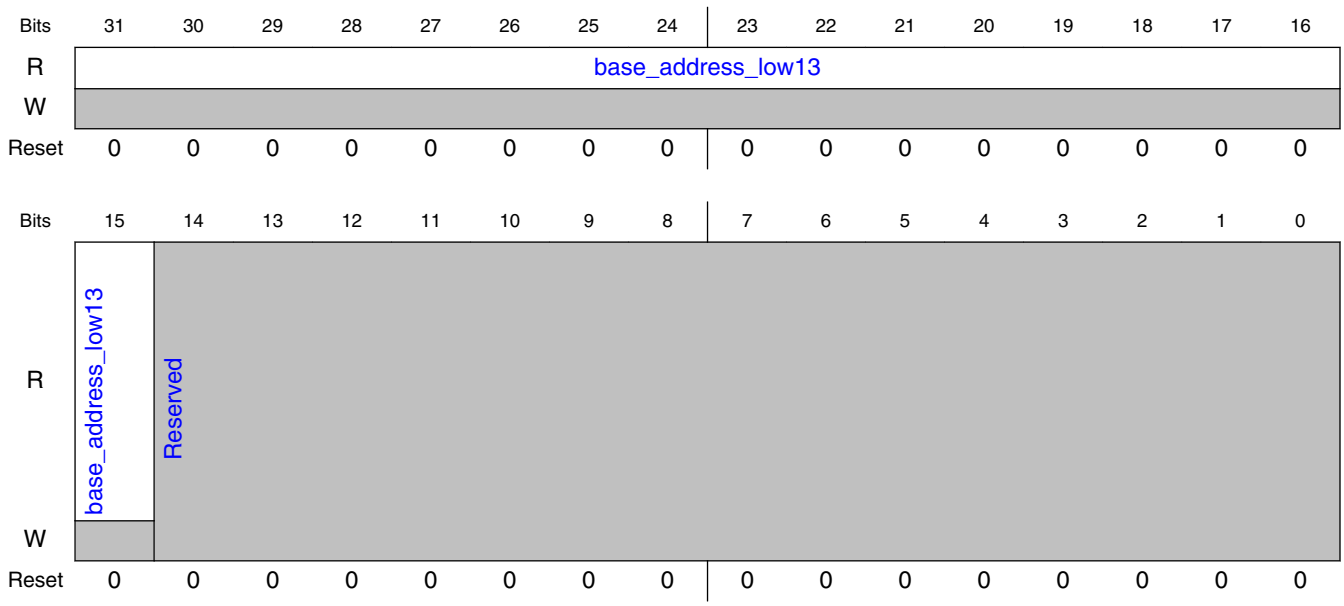
Field	Function
	Permissions setting for region *n*. If an AXI transaction occurs to region *n*, the value in the sp*n* field controls whether the TZASC permits the transaction to proceed.
27-16 —	Reserved, Should be Zero (SBZ).
15-8 subregion_disable12	subregion_disable12 Regions are split into eight equal-sized sub-regions, and each bit enables the corresponding subregion to be disabled: Bit [15] = 1 Subregion 7 is disabled. Bit [14] = 1 Subregion 6 is disabled. Bit [13] = 1 Subregion 5 is disabled. Bit [12] = 1 Subregion 4 is disabled. Bit [11] = 1 Subregion 3 is disabled. Bit [10] = 1 Subregion 2 is disabled. Bit [9] = 1 Subregion 1 is disabled. Bit [8] = 1 Subregion 0 is disabled.
7 —	Reserved, Should be Zero (SBZ).
6-1 size12	size12 Size of region *n*.The AXI address width, that is AXI_ADDRESS_MSB+1, controls the upper limit value of this field. The table below shows how the size *n* field controls the region size and what constraints, if any, the TZASC applies to the base address to ensure that a region starts on the boundary of the region size.
0 en12	en12 Enable for region *n*: 0 = region *n* is disabled 1 = region *n* is enabled.

10.3.53 Region Setup Low 13 Register (region_setup_low_13)

10.3.53.1 Offset

Register	Offset
region_setup_low_13	1D0h

10.3.53.2 Diagram



10.3.53.3 Fields

Field	Function
31-15 base_address_low13	base_address_low13 Controls the base address [31:15] of region n. For region 0, this field is Read Only (RO). The TZASC sets the base address of region 0 to 0x0. The TZASC only permits a region to start at address 0x0, or at a multiple of its region size. For example, if the size of a region is 512MB, and it is not at address 0x0, the only valid settings for this field are: b00100000000000000 b01000000000000000 b01100000000000000 b10000000000000000 b10100000000000000 b11000000000000000 b11100000000000000 If you attempt to set an inappropriate base address for the size of the region, the TZASC ignores certain bits depending on the region size. See Table on register size for more information.
14-0 —	Reserved, Should be Zero (SBZ).

10.3.54 Region Setup High 13 Register (region_setup_high_13)

10.3.54.1 Offset

Register	Offset
region_setup_high_13	1D4h

10.3.54.2 Function

The region_setup_high register is none for region 0, for other regions it controls the base address [63:32] of region n.There are no usage constraints. Available in all configurations of the TZASC.

10.3.54.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	base_address_high13															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	base_address_high13															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

10.3.54.4 Fields

Field	Function
31-0 base_address_high13	base_address_high13 Controls the base address [63:32] of region n. For region 0, this field is RO. The TZASC sets the base address of region 0 to 0x0. The TZASC only permits a region to start at address 0x0, or at a multiple of its region size. If you program a region size to be 8GB or more, then the TZASC might ignore certain bits depending on the region size.

10.3.55 Region Attributes 13 Register (region_attributes_13)

10.3.55.1 Offset

Register	Offset
region_attributes_13	1D8h

10.3.55.2 Function

The region_attributes_n register controls the permissions, region size, subregion disable, and region enable. There are no usage constraints. Available in all configurations of the TZASC.

Table 10-13. Region size

size<n> field	Size of region <n>	Base address [1] constraints
b000000-b001101	Reserved	-
b001110	32KB	-
b001111	64KB	Bit [15] must be zero
b010000	128KB	Bits [16:15] must be zero
b010001	256KB	Bits [17:15] must be zero
b010010	512KB	Bits [18:15] must be zero
b010011	1MB	Bits [19:15] must be zero
b010100	2MB	Bits [20:15] must be zero
b010101	4MB	Bits [21:15] must be zero
b010110	8MB	Bits [22:15] must be zero
b010111	16MB	Bits [23:15] must be zero
b011000	32MB	Bits [24:15] must be zero
b011001	64MB	Bits [25:15] must be zero
b011010	128MB	Bits [26:15] must be zero
b011011	256MB	Bits [27:15] must be zero
b011100	512MB	Bits [28:15] must be zero
b011101	1GB	Bits [29:15] must be zero
b011110	2GB	Bits [30:15] must be zero
b011111	4GB	Bits [31:15] must be zero
b100000	8GB	Bits [32:15] must be zero
b100001	16GB	Bits [33:15] must be zero
b100010	32GB	Bits [34:15] must be zero
b100011	64GB	Bits [35:15] must be zero

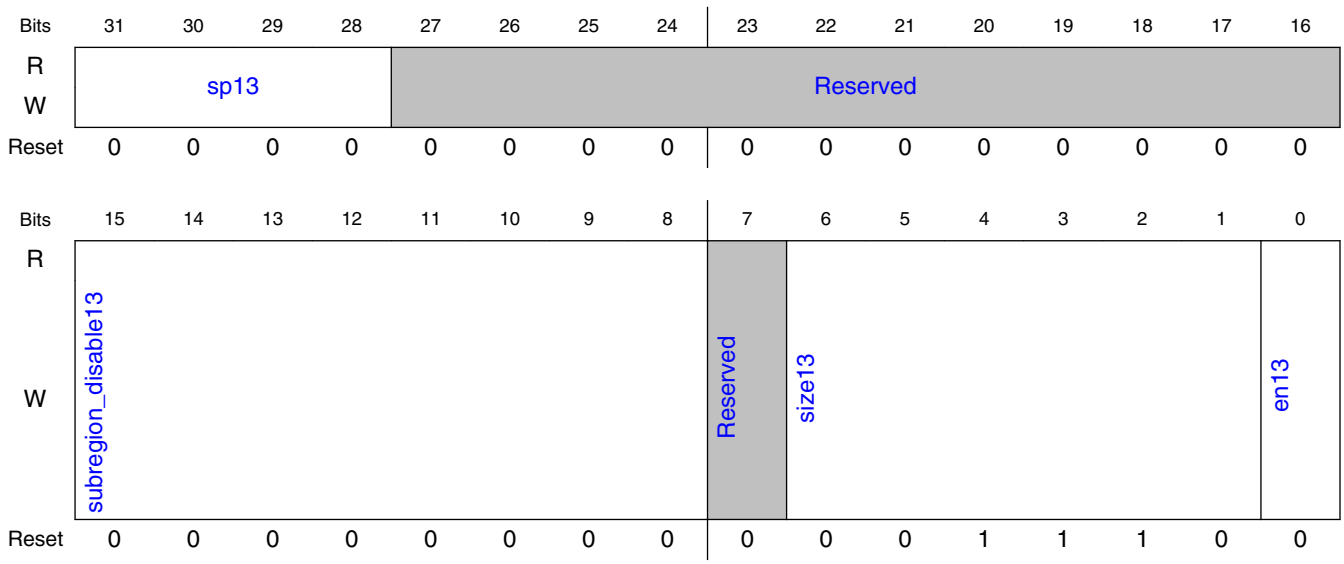
Table continues on the next page...

Table 10-13. Region size (continued)

b100100	128GB	Bits [36:15] must be zero
b100101	256GB	Bits [37:15] must be zero
b100110	512GB	Bits [38:15] must be zero
b100111	1TB	Bits [39:15] must be zero
b101000	2TB	Bits [40:15] must be zero
b101001	4TB	Bits [41:15] must be zero
b101010	8TB	Bits [42:15] must be zero
b101011	16TB	Bits [43:15] must be zero
b101100	32TB	Bits [44:15] must be zero
b101101	64TB	Bits [45:15] must be zero
b101110	128TB	Bits [46:15] must be zero
b101111	256TB	Bits [47:15] must be zero
b110000	512TB	Bits [48:15] must be zero
b110001	1PB	Bits [49:15] must be zero
b110010	2PB	Bits [50:15] must be zero
b110011	4PB	Bits [51:15] must be zero
b110100	8PB	Bits [52:15] must be zero
b110101	16PB	Bits [53:15] must be zero
b110110	32PB	Bits [54:15] must be zero
b110111	64PB	Bits [55:15] must be zero
b111000	128PB	Bits [56:15] must be zero
b111001	256PB	Bits [57:15] must be zero
b111010	512PB	Bits [58:15] must be zero
b111011	1EB	Bits [59:15] must be zero
b111100	2EB	Bits [60:15] must be zero
b111101	4EB	Bits [61:15] must be zero
b111110	8EB	Bits [62:15] must be zero
b111111	16EB	Bits [63:15] must be zero

[1]The region_setup_low_<n> Register and region_setup_high_<n> Register contain the base address. See Table 313 on page 318 and Table 314 on page 319.

10.3.55.3 Diagram



10.3.55.4 Fields

Field	Function
31-28 sp13	sp13 Permissions setting for region *n*. If an AXI transaction occurs to region *n*, the value in the sp*n* field controls whether the TZASC permits the transaction to proceed.
27-16 —	Reserved, Should be Zero (SBZ).
15-8 subregion_disable13	subregion_disable13 Regions are split into eight equal-sized sub-regions, and each bit enables the corresponding subregion to be disabled: Bit [15] = 1 Subregion 7 is disabled. Bit [14] = 1 Subregion 6 is disabled. Bit [13] = 1 Subregion 5 is disabled. Bit [12] = 1 Subregion 4 is disabled. Bit [11] = 1 Subregion 3 is disabled. Bit [10] = 1 Subregion 2 is disabled. Bit [9] = 1 Subregion 1 is disabled. Bit [8] = 1 Subregion 0 is disabled.
7 —	Reserved, Should be Zero (SBZ).
6-1	size13

Table continues on the next page...

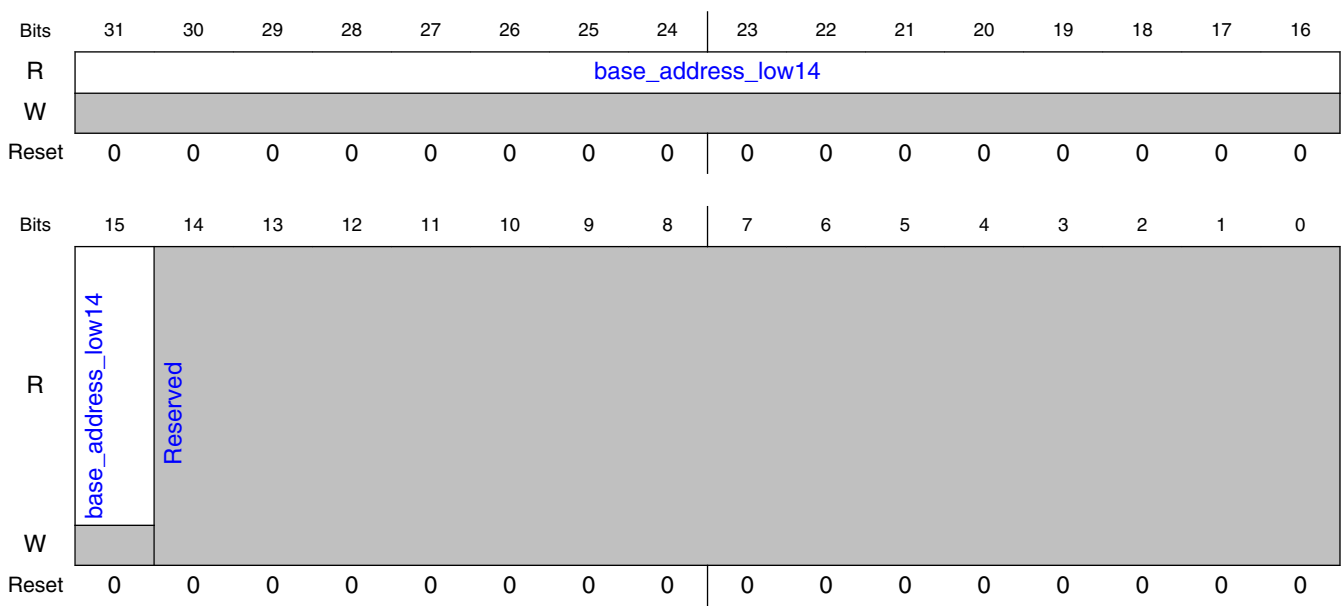
Field	Function
size13	Size of region *n*.The AXI address width, that is AXI_ADDRESS_MSB+1, controls the upper limit value of this field. The table below shows how the size *n* field controls the region size and what constraints, if any, the TZASC applies to the base address to ensure that a region starts on the boundary of the region size.
0 en13	en13 Enable for region *n*: 0 = region *n* is disabled 1 = region *n* is enabled.

10.3.56 Region Setup Low 14 Register (region_setup_low_14)

10.3.56.1 Offset

Register	Offset
region_setup_low_14	1E0h

10.3.56.2 Diagram



10.3.56.3 Fields

Field	Function
31-15 base_address_low14	base_address_low14 Controls the base address [31:15] of region n. For region 0, this field is Read Only (RO). The TZASC sets the base address of region 0 to 0x0. The TZASC only permits a region to start at address 0x0, or at a multiple of its region size. For example, if the size of a region is 512MB, and it is not at address 0x0, the only valid settings for this field are: b00100000000000000 b01000000000000000 b01100000000000000 b10000000000000000 b10100000000000000 b11000000000000000 b11100000000000000 If you attempt to set an inappropriate base address for the size of the region, the TZASC ignores certain bits depending on the region size. See Table on register size for more information.
14-0 —	Reserved, Should be Zero (SBZ).

10.3.57 Region Setup High 14 Register (region_setup_high_14)

10.3.57.1 Offset

Register	Offset
region_setup_high_14	1E4h

10.3.57.2 Function

The region_setup_high register is none for region 0, for other regions it controls the base address [63:32] of region n.There are no usage constraints. Available in all configurations of the TZASC.

10.3.57.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	base_address_high14															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	base_address_high14															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

10.3.57.4 Fields

Field	Function
31-0 base_address_high14	base_address_high14 Controls the base address [63:32] of region n. For region 0, this field is RO. The TZASC sets the base address of region 0 to 0x0. The TZASC only permits a region to start at address 0x0, or at a multiple of its region size. If you program a region size to be 8GB or more, then the TZASC might ignore certain bits depending on the region size.

10.3.58 Region Attributes 14 Register (region_attributes_14)

10.3.58.1 Offset

Register	Offset
region_attributes_14	1E8h

10.3.58.2 Function

The region_attributes_n register controls the permissions, region size, subregion disable, and region enable. There are no usage constraints. Available in all configurations of the TZASC.

Table 10-14. Region size

size<n> field	Size of region <n>	Base address [1] constraints
b000000-b001101	Reserved	-
b001110	32KB	-
b001111	64KB	Bit [15] must be zero
b010000	128KB	Bits [16:15] must be zero
b010001	256KB	Bits [17:15] must be zero
b010010	512KB	Bits [18:15] must be zero
b010011	1MB	Bits [19:15] must be zero
b010100	2MB	Bits [20:15] must be zero
b010101	4MB	Bits [21:15] must be zero
b010110	8MB	Bits [22:15] must be zero
b010111	16MB	Bits [23:15] must be zero
b011000	32MB	Bits [24:15] must be zero
b011001	64MB	Bits [25:15] must be zero
b011010	128MB	Bits [26:15] must be zero
b011011	256MB	Bits [27:15] must be zero
b011100	512MB	Bits [28:15] must be zero
b011101	1GB	Bits [29:15] must be zero
b011110	2GB	Bits [30:15] must be zero
b011111	4GB	Bits [31:15] must be zero
b100000	8GB	Bits [32:15] must be zero
b100001	16GB	Bits [33:15] must be zero
b100010	32GB	Bits [34:15] must be zero
b100011	64GB	Bits [35:15] must be zero
b100100	128GB	Bits [36:15] must be zero
b100101	256GB	Bits [37:15] must be zero
b100110	512GB	Bits [38:15] must be zero
b100111	1TB	Bits [39:15] must be zero
b101000	2TB	Bits [40:15] must be zero
b101001	4TB	Bits [41:15] must be zero
b101010	8TB	Bits [42:15] must be zero
b101011	16TB	Bits [43:15] must be zero
b101100	32TB	Bits [44:15] must be zero
b101101	64TB	Bits [45:15] must be zero
b101110	128TB	Bits [46:15] must be zero
b101111	256TB	Bits [47:15] must be zero
b110000	512TB	Bits [48:15] must be zero
b110001	1PB	Bits [49:15] must be zero
b110010	2PB	Bits [50:15] must be zero
b110011	4PB	Bits [51:15] must be zero

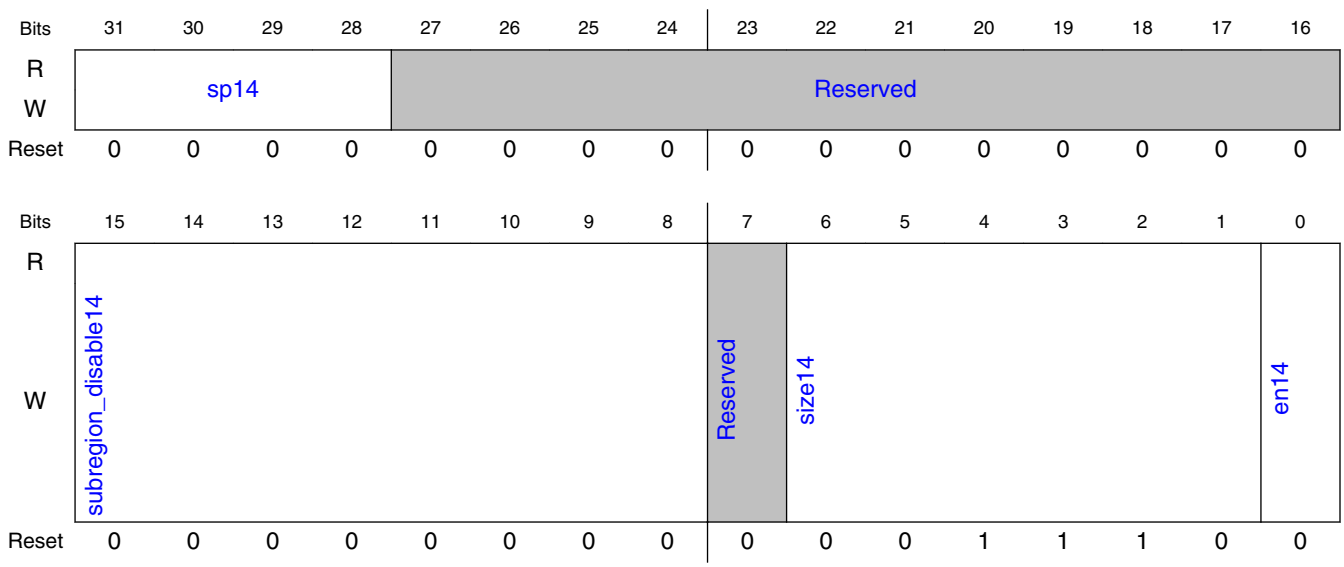
Table continues on the next page...

Table 10-14. Region size (continued)

b110100	8PB	Bits [52:15] must be zero
b110101	16PB	Bits [53:15] must be zero
b110110	32PB	Bits [54:15] must be zero
b110111	64PB	Bits [55:15] must be zero
b111000	128PB	Bits [56:15] must be zero
b111001	256PB	Bits [57:15] must be zero
b111010	512PB	Bits [58:15] must be zero
b111011	1EB	Bits [59:15] must be zero
b111100	2EB	Bits [60:15] must be zero
b111101	4EB	Bits [61:15] must be zero
b111110	8EB	Bits [62:15] must be zero
b111111	16EB	Bits [63:15] must be zero

[1]The region_setup_low_<n> Register and region_setup_high_<n> Register contain the base address. See Table 313 on page 318 and Table 314 on page 319.

10.3.58.3 Diagram



10.3.58.4 Fields

Field	Function
31-28 sp14	sp14

Table continues on the next page...

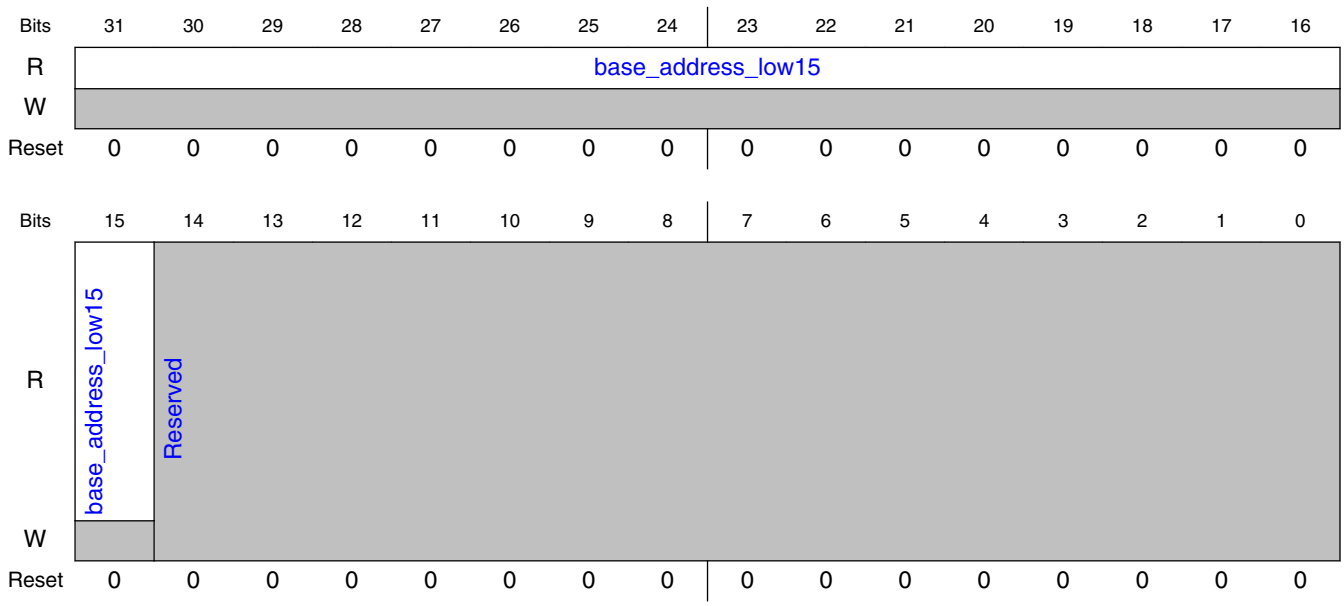
Field	Function
	Permissions setting for region *n*. If an AXI transaction occurs to region *n*, the value in the sp*n* field controls whether the TZASC permits the transaction to proceed.
27-16 —	Reserved, Should be Zero (SBZ).
15-8 subregion_disable14	subregion_disable14 Regions are split into eight equal-sized sub-regions, and each bit enables the corresponding subregion to be disabled: Bit [15] = 1 Subregion 7 is disabled. Bit [14] = 1 Subregion 6 is disabled. Bit [13] = 1 Subregion 5 is disabled. Bit [12] = 1 Subregion 4 is disabled. Bit [11] = 1 Subregion 3 is disabled. Bit [10] = 1 Subregion 2 is disabled. Bit [9] = 1 Subregion 1 is disabled. Bit [8] = 1 Subregion 0 is disabled.
7 —	Reserved, Should be Zero (SBZ).
6-1 size14	size14 Size of region *n*.The AXI address width, that is AXI_ADDRESS_MSB+1, controls the upper limit value of this field. The table below shows how the size *n* field controls the region size and what constraints, if any, the TZASC applies to the base address to ensure that a region starts on the boundary of the region size.
0 en14	en14 Enable for region *n*: 0 = region *n* is disabled 1 = region *n* is enabled.

10.3.59 Region Setup Low 15 Register (region_setup_low_15)

10.3.59.1 Offset

Register	Offset
region_setup_low_15	1F0h

10.3.59.2 Diagram



10.3.59.3 Fields

Field	Function
31-15 base_address_low15	base_address_low15 Controls the base address [31:15] of region n. For region 0, this field is Read Only (RO). The TZASC sets the base address of region 0 to 0x0. The TZASC only permits a region to start at address 0x0, or at a multiple of its region size. For example, if the size of a region is 512MB, and it is not at address 0x0, the only valid settings for this field are: b00100000000000000 b01000000000000000 b01100000000000000 b10000000000000000 b10100000000000000 b11000000000000000 b11100000000000000 If you attempt to set an inappropriate base address for the size of the region, the TZASC ignores certain bits depending on the region size. See Table on register size for more information.
14-0 —	Reserved, Should be Zero (SBZ).

10.3.60 Region Setup High 15 Register (region_setup_high_15)

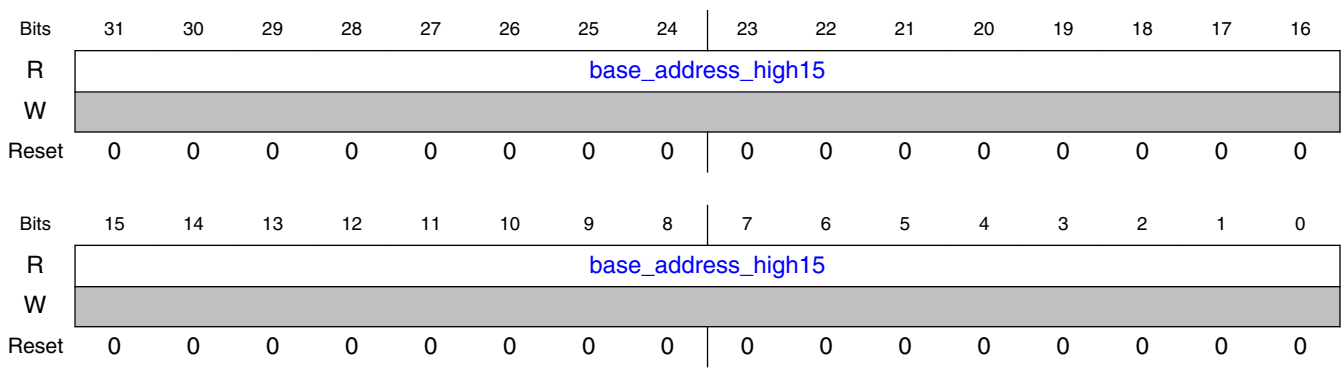
10.3.60.1 Offset

Register	Offset
region_setup_high_15	1F4h

10.3.60.2 Function

The region_setup_high register is none for region 0, for other regions it controls the base address [63:32] of region n.There are no usage constraints. Available in all configurations of the TZASC.

10.3.60.3 Diagram



10.3.60.4 Fields

Field	Function
31-0 base_address_high15	base_address_high15 Controls the base address [63:32] of region n. For region 0, this field is RO. The TZASC sets the base address of region 0 to 0x0. The TZASC only permits a region to start at address 0x0, or at a multiple of its region size. If you program a region size to be 8GB or more, then the TZASC might ignore certain bits depending on the region size.

10.3.61 Region Attributes 15 Register (region_attributes_15)

10.3.61.1 Offset

Register	Offset
region_attributes_15	1F8h

10.3.61.2 Function

The region_attributes_n register controls the permissions, region size, subregion disable, and region enable. There are no usage constraints. Available in all configurations of the TZASC.

Table 10-15. Region size

size<n> field	Size of region <n>	Base address [1] constraints
b000000-b001101	Reserved	-
b001110	32KB	-
b001111	64KB	Bit [15] must be zero
b010000	128KB	Bits [16:15] must be zero
b010001	256KB	Bits [17:15] must be zero
b010010	512KB	Bits [18:15] must be zero
b010011	1MB	Bits [19:15] must be zero
b010100	2MB	Bits [20:15] must be zero
b010101	4MB	Bits [21:15] must be zero
b010110	8MB	Bits [22:15] must be zero
b010111	16MB	Bits [23:15] must be zero
b011000	32MB	Bits [24:15] must be zero
b011001	64MB	Bits [25:15] must be zero
b011010	128MB	Bits [26:15] must be zero
b011011	256MB	Bits [27:15] must be zero
b011100	512MB	Bits [28:15] must be zero
b011101	1GB	Bits [29:15] must be zero
b011110	2GB	Bits [30:15] must be zero
b011111	4GB	Bits [31:15] must be zero
b100000	8GB	Bits [32:15] must be zero
b100001	16GB	Bits [33:15] must be zero
b100010	32GB	Bits [34:15] must be zero
b100011	64GB	Bits [35:15] must be zero

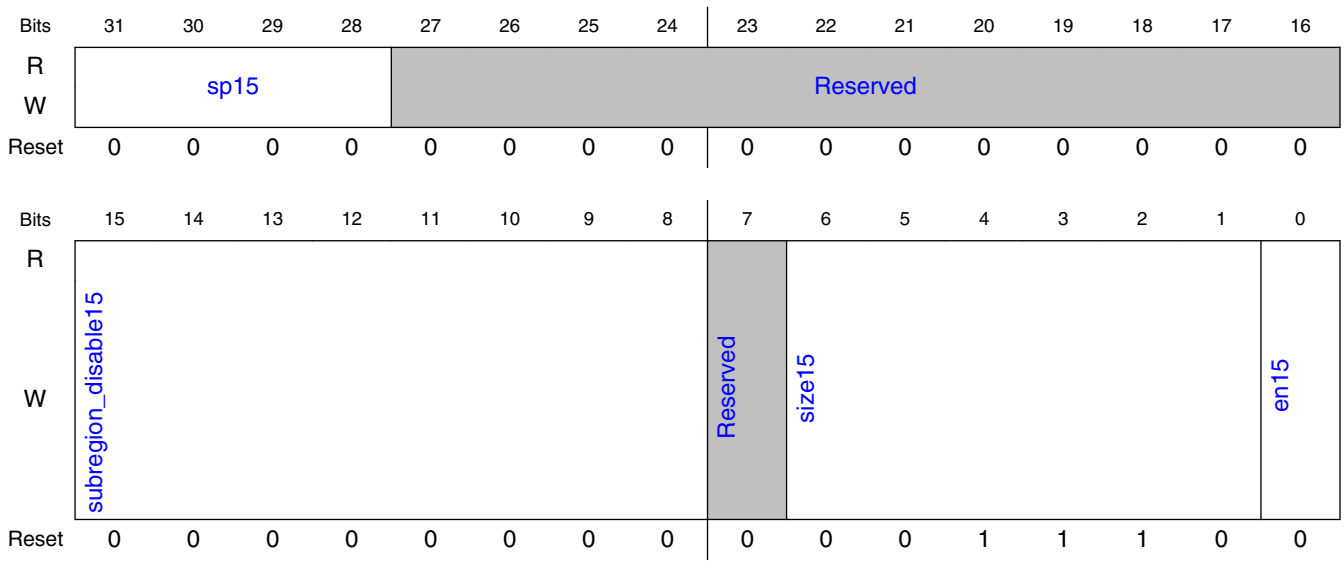
Table continues on the next page...

Table 10-15. Region size (continued)

b100100	128GB	Bits [36:15] must be zero
b100101	256GB	Bits [37:15] must be zero
b100110	512GB	Bits [38:15] must be zero
b100111	1TB	Bits [39:15] must be zero
b101000	2TB	Bits [40:15] must be zero
b101001	4TB	Bits [41:15] must be zero
b101010	8TB	Bits [42:15] must be zero
b101011	16TB	Bits [43:15] must be zero
b101100	32TB	Bits [44:15] must be zero
b101101	64TB	Bits [45:15] must be zero
b101110	128TB	Bits [46:15] must be zero
b101111	256TB	Bits [47:15] must be zero
b110000	512TB	Bits [48:15] must be zero
b110001	1PB	Bits [49:15] must be zero
b110010	2PB	Bits [50:15] must be zero
b110011	4PB	Bits [51:15] must be zero
b110100	8PB	Bits [52:15] must be zero
b110101	16PB	Bits [53:15] must be zero
b110110	32PB	Bits [54:15] must be zero
b110111	64PB	Bits [55:15] must be zero
b111000	128PB	Bits [56:15] must be zero
b111001	256PB	Bits [57:15] must be zero
b111010	512PB	Bits [58:15] must be zero
b111011	1EB	Bits [59:15] must be zero
b111100	2EB	Bits [60:15] must be zero
b111101	4EB	Bits [61:15] must be zero
b111110	8EB	Bits [62:15] must be zero
b111111	16EB	Bits [63:15] must be zero

[1]The region_setup_low_<n> Register and region_setup_high_<n> Register contain the base address. See Table 313 on page 318 and Table 314 on page 319.

10.3.61.3 Diagram



10.3.61.4 Fields

Field	Function
31-28 sp15	sp15 Permissions setting for region *n*. If an AXI transaction occurs to region *n*, the value in the sp*n* field controls whether the TZASC permits the transaction to proceed.
27-16 —	Reserved, Should be Zero (SBZ).
15-8 subregion_disable15	subregion_disable15 Regions are split into eight equal-sized sub-regions, and each bit enables the corresponding subregion to be disabled: Bit [15] = 1 Subregion 7 is disabled. Bit [14] = 1 Subregion 6 is disabled. Bit [13] = 1 Subregion 5 is disabled. Bit [12] = 1 Subregion 4 is disabled. Bit [11] = 1 Subregion 3 is disabled. Bit [10] = 1 Subregion 2 is disabled. Bit [9] = 1 Subregion 1 is disabled. Bit [8] = 1 Subregion 0 is disabled.
7 —	Reserved, Should be Zero (SBZ).
6-1	size15

Table continues on the next page...

Field	Function
size15	Size of region *n*.The AXI address width, that is AXI_ADDRESS_MSB+1, controls the upper limit value of this field. The table below shows how the size *n* field controls the region size and what constraints, if any, the TZASC applies to the base address to ensure that a region starts on the boundary of the region size.
0 en15	en15 Enable for region *n*: 0 = region *n* is disabled 1 = region *n* is enabled.

10.3.62 Integration Test Control Register (itcrg)

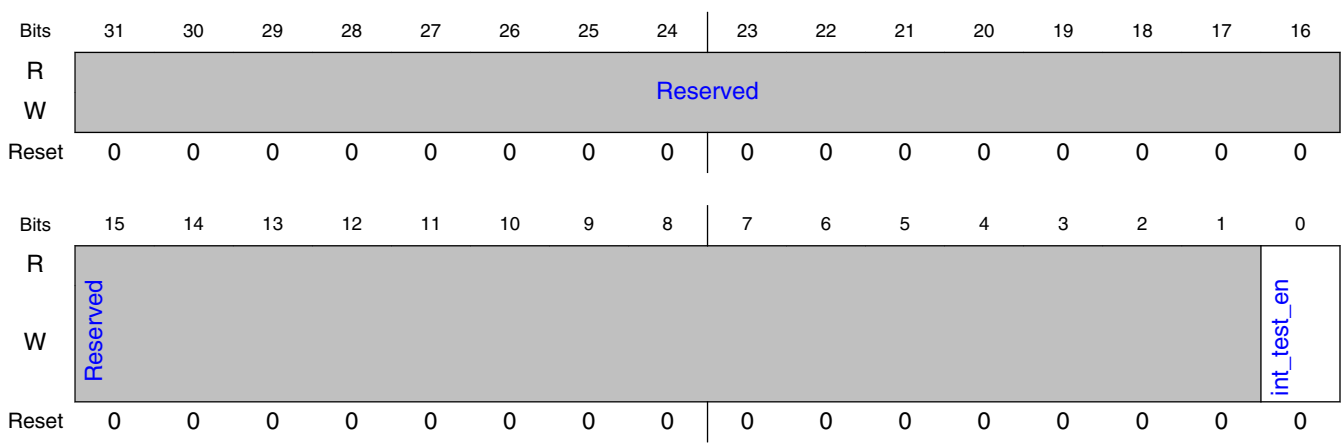
10.3.62.1 Offset

Register	Offset
itcrg	E00h

10.3.62.2 Function

The itcrg register enables the integration test logic. Use this in integration test mode. Available in all configurations of the TZASC.

10.3.62.3 Diagram



10.3.62.4 Fields

Field	Function
31-1 —	Undefined. Write as zero.
0 int_test_en	int_test_en Controls the enabling of, or provides the status of, the integration test logic: 0 = integration test logic is disabled 1 = integration test logic is enabled.

10.3.63 Integration Test Input Register (itip)

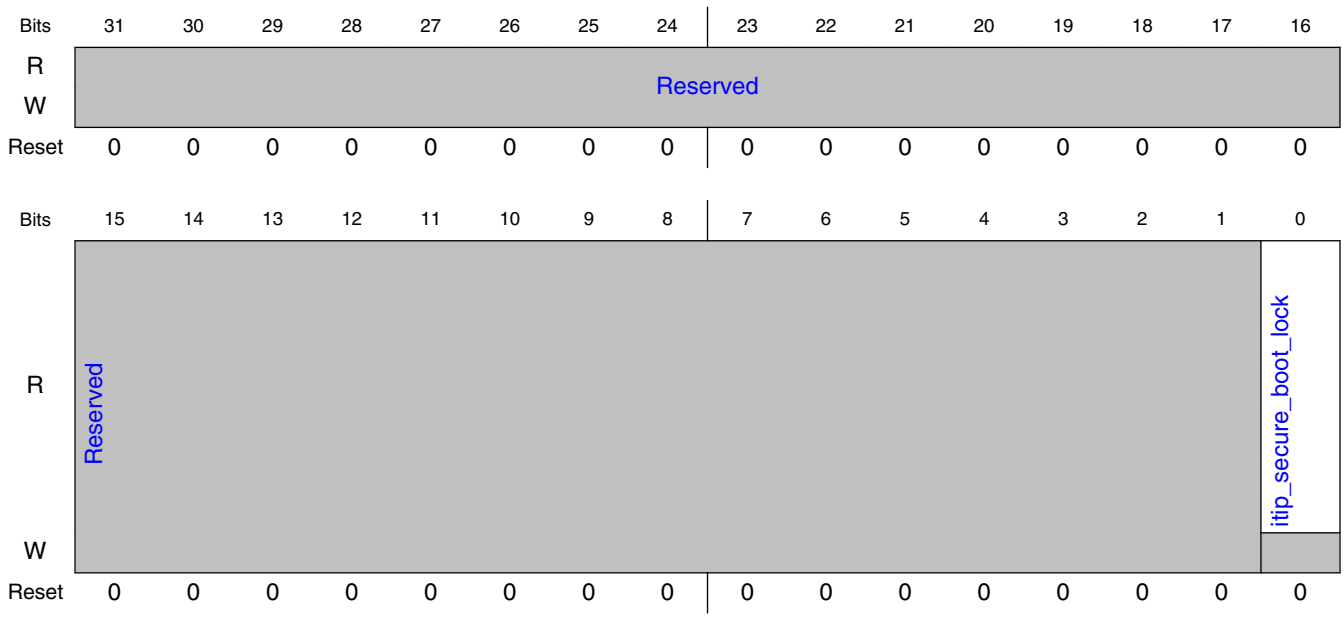
10.3.63.1 Offset

Register	Offset
itip	E04h

10.3.63.2 Function

The itip register enables a processor to read the status of secure_boot_lock. Integration test logic must be enabled otherwise reads return 0x0. See Integration Test Control Register for information about enabling the integration test logic. Available in all configurations of the TZASC.

10.3.63.3 Diagram



10.3.63.4 Fields

Field	Function
31-1 —	Reserved.
0 itip_secure_boot_lock	itip_secure_boot_lock

10.3.64 Integration Test Output Register (itop)

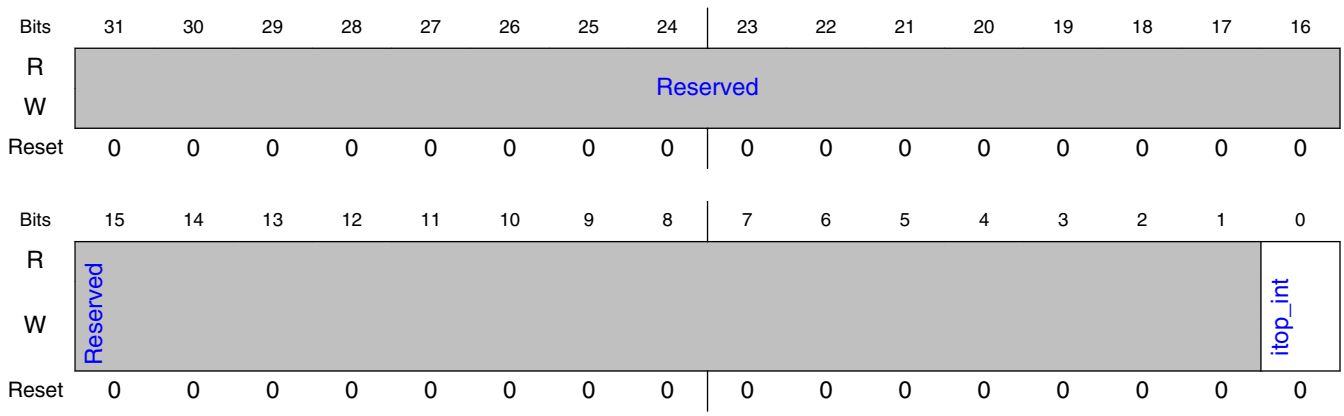
10.3.64.1 Offset

Register	Offset
itop	E08h

10.3.64.2 Function

The itop register enables a processor to set the status of tzasc_int in integration test mode. Usage constraints Integration test logic must be enabled otherwise it ignores writes and reads return 0x0. See Integration Test Control Register for information about enabling the integration test logic. Available in all configurations of the TZASC.

10.3.64.3 Diagram



10.3.64.4 Fields

Field	Function
31-1 —	Undefined. Write as zero.
0 itop_int	itop_int Set or reset the value of tzasc_int port by writing 1 or 0 into itop_int bit. If you read, the written value can be read back. 0 = tzasc_int is LOW 1 = tzasc_int is HIGH.

10.4 Functional description

This chapter describes the TZASC operation. It contains the following sections:

- Functional operation
- Constraints of use

10.4.1 Functional operation

The TZASC performs security checks on AXI accesses to memory and DDR memory space. This supports configurable number of regions. Each region is programmable for size, base address, enable, and security parameters. Using the secure_boot_lock, the programmers view can be locked to prevent erroneous writes. See Preventing writes to registers and using secure_boot_lock. The TZASC provides programmability in reporting faults using AXI response channel and interrupt.

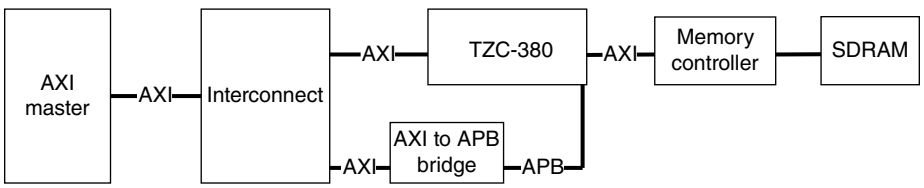


Figure 10-3. Functional operation of TZASC

NOTE

The *CoreLink TrustZone Address Space Controller (TZC-380) Supplement to AMBA Designer (ADR-301) User Guide* provides information about how to configure the controller.

10.4.1.1 Regions

A region is a contiguous area of address space. The TZASC provides each region with a programmable security permissions field. The security permissions value is used to enable the TZASC to either accept or deny a transaction access to that region. The transaction's secure vs non-secure attributes are used to determine the security settings of that transaction.

The TZASC always provides two regions, region 0 and region 1, and you can configure it to provide additional regions. With the exception of region 0, the TZASC enables you to program the following operating parameters for each region:

- Region enable.
- Security permissions.
- Base address.
- Size. The minimum address size of a region is 32KB.
- Subregion disable. See Subregions.

NOTE

Region 0 is known as the background region because it occupies the total memory space. You can program the security permissions of region 0, but the following parameters are fixed:

- **Base address:**0x0
- **Size:**The AXI_ADDRESS_MSB configuration parameter controls the address range of the TZASC, and therefore the region size.
- **Subregion disable:**This feature is not available for region 0.

10.4.1.2 Priority

The priority of a region is fixed and is determined by the region number. Figure 10-4 shows how the priority of a region increases with the region number.

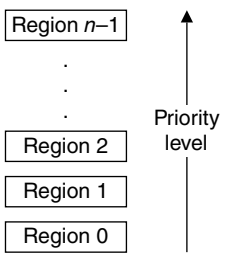


Figure 10-4. Region priority

When a transaction is received, its address is checked for a match with all the configured regions in turn. The order in which the regions are checked is determined by the priority level, the highest priority level is first. The first region that matches the transaction address match is used as the matching region. The matching regions security permission determines whether the transaction is permitted.

10.4.1.3 Subregions

The TZASC divides each region into eight equal-sized, non-overlapping subregions. Figure 10-5 shows the subregions for an example region that is programmed to occupy an address span of 32KB.

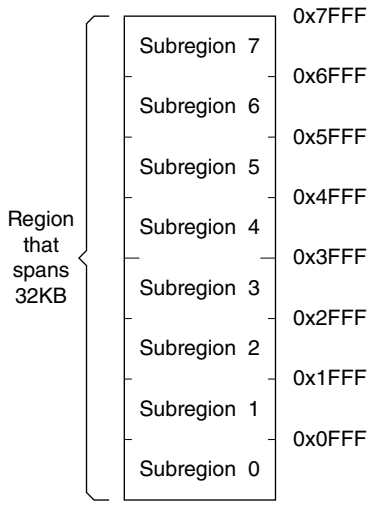


Figure 10-5. Subregion example

10.4.1.4 Subregion disable

With the exception of region 0, you can program the TZASC to disable any or all of the eight subregions that comprise a region. When a subregion is disabled, the security permissions for its address range are provided by the next highest priority region that overlaps the address range.

Example configuration for subregion disable

Figure 10-6 shows an example configuration that supports four regions, where:

- region 2 and region 3 are partially overlapped
- region 1 and region 3 are partially overlapped
- region 0 is overlapped with all regions.

With some subregions of region 1, region 2, and region 3 are disabled, and the resulting region permissions of the entire address space is shown in the Figure 10-6.

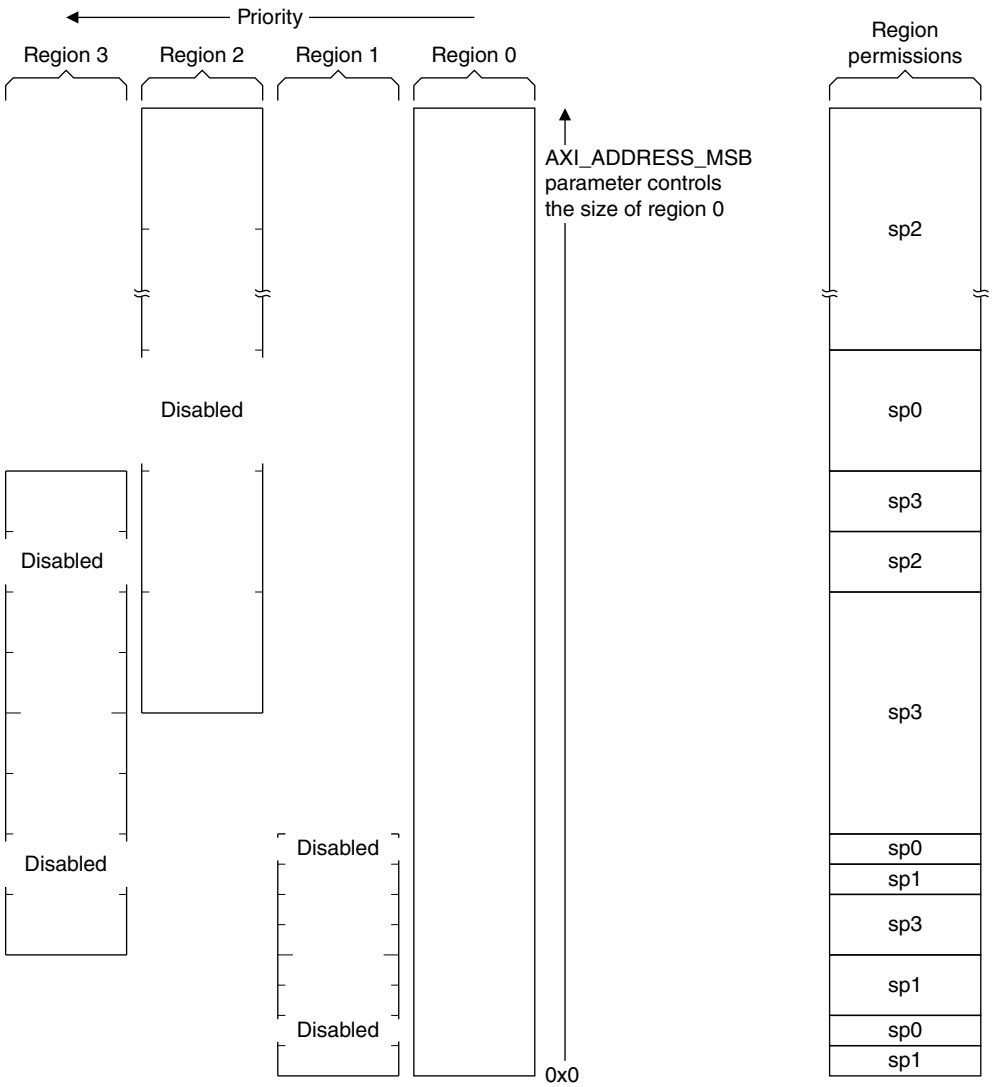


Figure 10-6. Subregion disable example

NOTE

In Figure 10-6

- all subregions are enabled unless otherwise stated
- sp*n* represents the region permissions of region *n* .

10.4.1.5 Region security permissions

The TZASC enables you to program the security access permissions for any region that it is configured. A region is assigned a security permissions field, sp<n>, in its region_attributes_<n> Register that enables you to have complete control of the permissions for that region. See Register Descriptions.

Security inversion

There are two modes of operation for the region security permissions, with or without security inversion.

By default, if you program a region to support non-secure accesses, the TZASC ensures that region must also support secure accesses. For example, if you program the region permissions for region 3 to be non-secure read only, the TZASC permits access to region 3 for secure reads and non-secure reads.

If you require that some regions are not accessible to masters in Secure state, but are accessible in Non-secure state, then you must enable security inversion.

See Region security permissions and Security Inversion Register (security_inversion_en) for more information.

Programming security permissions when security inversion is disabled

By default, security inversion is disabled and therefore the TZASC only permits you to program certain combinations of security permissions. These combinations ensure that a master in Secure state is not denied access to a region that is programmed to only accept non-secure accesses. Table 10-16 shows the possible security permissions when security inversion is disabled.

Table 10-16. Region security permissions when security inversion is disabled

	sp<n> field controls if the TZASC permits access for the following AXI transactions			
sp<n> field[1]	Secure read	Secure write	Non-secure read	Non-secure write
b0000	No	No	No	No
b0100	No	Yes	No	No
b0001, b0101	No	Yes	No	Yes
b1000	Yes	No	No	No
b0010, b1010	Yes	No	Yes	No
b1100	Yes	Yes	No	No
b1001, b1101	Yes	Yes	No	Yes
b0110, b1110	Yes	Yes	Yes	No
b0011, b0111, b1011, b1111	Yes	Yes	Yes	Yes

1. See Region Attributes 0 Register (region_attributes_0) for programming information.

Programming security permissions when security inversion is enabled

If you enable security inversion, the TZASC permits you to program any combination of security permissions as Table 10-17 shows.

Table 10-17. Region security permissions when security inversion is enabled

	sp<n> field controls if the TZASC permits access for the following AXI transactions			
sp<n> field[1]	Secure read	Secure write	Non-secure read	Non-secure write
b0000	No	No	No	No
b0001	No	No	No	Yes
b0010	No	No	Yes	No
b0011	No	No	Yes	Yes
b0100	No	Yes	No	No
b0101	No	Yes	No	Yes
b0110	No	Yes	Yes	No
b0111	No	Yes	Yes	Yes
b1000	Yes	No	No	No
b1001	Yes	No	No	Yes
b1010	Yes	No	Yes	No
b1011	Yes	No	Yes	Yes
b1100	Yes	Yes	No	No
b1101	Yes	Yes	No	Yes
b1110	Yes	Yes	Yes	No
b1111	Yes	Yes	Yes	Yes

1. See Region Attributes 0 Register (region_attributes_0) for programming information.

Table 10-18 shows a typical example of memory map along with the register programming. The TZASC is configured to have 16 regions.

Table 10-18. Typical example of memory map along with the register programming

Region	Region[1]	Lock	Starting address	Region size	Region size	Sp[2]	Description
Region_0 (Default)	Enable	No	0x0	max	-	1100	Secure *Read Write* access (RW).
Region_1	Enable	No	0x0	64MB	b011001	1111	Non-secure *Read or Write access* (R/W), Secure R/W.

Table continues on the next page...

Table 10-18. Typical example of memory map along with the register programming (continued)

Region_2	Enable	No	0x0	16MB	b010111	1110	Non-secure *Read Only access* (RO), Secure RW for the normal world OS kernel.
Region_3	Enable	No	0x3D00000	512KB	b010010	1111	Regularly switched Non-secure, or Secure RW for a more complex shared memory buffers.
Region_4	Enable	No	0x3D80000	512KB	b010010	1100	Non-secure *No Access* (NA), Secure RW, a dedicated area for secure LCD Controller frame buffer.
Region_5	Enable	No	0x80000000	32KB	b001110	1111	Non-secure RW, Secure RW for address range of general peripherals such as screen control, and keyboard hardware.
Region_6	Enable	Yes	0x3C00000	512KB	b010010	1011	Non-secure RW, Secure RO for streaming from the normal world to the secure world.
Region_7	Enable	Yes	0x3C80000	512KB	b010010	1110	Non-secure RO, Secure RW for streaming from the

Table continues on the next page...

Table 10-18. Typical example of memory map along with the register programming (continued)

							secure world to the normal world.
Region_8	Enable	Yes	0x3E00000	512KB	b010010	1000	Non-secure NA, Secure RO for the secure world OS kernel.
Region_9	Enable	Yes	0x3E80000	512KB	b010010	1100	Non-secure NA, Secure RW for the secure world OS applications, heap, and stacks.
Region_10	Enable	Yes	0x3F00000	1MB	b010011	1100	Non-secure NA, Secure RW for Secure world OS applications, heap, and stacks.
Region_11 [3]	Enable	Yes	0x80008000	32KB	b001110	1100	Non-secure NA, Secure RW for address range of secure peripherals such as *Random Number Generator* (RNG), and cryptography support hardware.
Region_12	Enable	Yes	0xF0000000	256MB	b011011	0011	Non-secure RW, Secure NA for FLASH holding normal world OS plus disk.
Region_13	Enable	Yes	0xF0000000	1MB	b010011	1100	Non-secure NA, Secure RW for FLASH for secure boot, secure world

Table continues on the next page...

Table 10-18. Typical example of memory map along with the register programming (continued)

							OS, secure configuration details.
Region_14	Disable	-	-	-	-	-	-
Region_15	Disable	-	-	-	-	-	-

1. Region can be either Enable or Disable.
2. Security Permission (sp).
3. In a more typical system, these devices would be protected by a TrustZone Protection Controller (BP147), and associated TrustZone aware AXI to APB Bridges (BP135).

NOTE

The implementers system design, and security requirements are taken into account for this example. And any actual software programming must depend on the system where TZASC is plugged.

10.4.1.6 Denied AXI transactions

If an AXI transaction has insufficient security privileges then for:

- **Reads:**The TZASC responds to the master by setting all bits of the read data bus to zero.

NOTE

If the TZASC is programmed to perform speculative accesses, it discards the data that it receives on read data.

- **Writes:**The TZASC prevents the transfer of data from the master to the slave by discarding the data of write data bus. If you program the TZASC to perform speculative accesses, it modifies the transfer to the slave by setting all bits of the:
  - write data bus to zero
  - write data strobe to zero.

NOTE

The action Register controls whether the TZASC signals to the master when a region permission failure occurs, and if so, the type of response it provides. See Action register (action).

10.4.1.7 Speculative accesses

By default, the TZASC performs read or write speculative accesses that means it forwards an AXI transaction address to a slave, before it verifies that the AXI transaction is permitted to read address or write address respectively.

The TZASC only permits the transfer of data between its AXI bus interfaces, after verifying the access that the read or write access is permitted respectively. If the verification fails, then it prevents the transfer of data between the master and slave as Denied AXI transactions describes.

You can disable speculative accesses by programming the speculation_control Register. See Speculation Control Register (speculation_control). When speculative accesses are disabled, the TZASC verifies the permissions of the access before it forwards the access to the slave. If the TZASC:

- Permits the access, it commences an AXI transaction to the slave, and it adds one clock latency.
- Denies the access, it prevents the transfer of data between the master and slave as Denied AXI transactions describes. In this situation, the slave is unaware when the TZASC prevents the master from accessing the slave.

NOTE

Enabling speculative access is a potential security risk, if the device that is being protected reacts to this transaction. Most devices do not have to react to this level of access, and speculative access is much faster than validating the address before issuing the transaction.

10.4.1.8 Preventing writes to registers and using secure_boot_lock

By suitably programming lockdown Register, see Lockdown Select Register (lockdown_select), and asserting secure_boot_lock signal makes the following registers read only:

- speculation_control Register. See Speculation Control Register (speculation_control).
- security_inversion_en Register. See Security Inversion Register (security_inversion_en).
- lockdown_range Register. See Lockdown Range Register (lockdown_range).

Locking down the region using lockdown_range and lockdown_select registers

By programming the lockdown_select, and lockdown_range registers, and asserting the secure_boot_lock signal, you can lockdown the behavior of the TZASC so that it prevents unintentional or erroneous write to the regions specified in the lockdown_range Register. However, read access to those regions is permitted:

- region_setup_low_<n> Register. See Region Setup Low 0 Register (region_setup_low_0)
- region_setup_high_<n> Register. See Region Setup High 0 Register (region_setup_high_0)
- region_attributes_<n> Register. See Region Attributes 0 Register (region_attributes_0).

The TZASC expects the secure_boot_lock signal to be asserted for at least one clock cycle. One clock after the secure_boot_lock is sampled HIGH by TZASC, then the registers mentioned in *Locking down the region using lockdown_range and lockdown_select registers* cannot be written, unless the TZASC is reset by asserting aresetn.

10.4.1.9 Using locked transaction sequences

If a master performs locked transaction sequences, a transaction might stall, or an AXI protocol violation might occur when:

Transaction sequence crosses a 4 KB boundary

If a locked transaction sequence crosses a 4 KB boundary and the regions have different region permissions, the TZASC might prevent access to the second region and therefore the slave would not receive the latter part of the locked transaction sequence.

NOTE

The AXI protocol recommends that locked transaction sequences do not cross a 4 KB address boundary.

Secure state change

During a locked transaction sequence, if a master changes the state of secure and non-secure attributes and the region has different region permissions for Secure state and Non-secure state, the TZASC might deny a transaction and therefore the slave would not receive the latter part of the locked transaction sequence.

Reads and writes

During a locked transaction sequence, if a master performs reads and writes to a region, depending on the region permissions, the TZASC might deny a transaction and therefore the slave would not receive the latter part of the locked transaction sequence.

10.4.1.10 Using exclusive accesses

If a master performs exclusive accesses to an address region, you must program the TZASC to permit read and write accesses to that address region, for the expected settings of secure and non-secure attributes, otherwise the read or write transaction might fail.

10.4.2 Constraints of use

The TZASC has the following considerations relating to change in programmers view on an active system:

- When changing the setting of a TZASC region,
  - The current accepted AXI transaction, if it falls into that region, would act according to the previous settings for that region.
  - Any other outstanding AXI transactions, that falls into that region, would effect by the new settings for that region.
- Given little ability to predict that the mentioned AXI transactions would effect, it is obviously desirable that there are no outstanding AXI transactions when a regions setting are changed.
  - In simple systems this can potentially be achieved by the core not accessing the given region during the period of the cores transition between security states. Even in these cases, the status of cached data and instructions needs to be considered.
  - In more complicated systems the code that changes the TZASC region settings must have to inform other AXI bus masters to desist or complete acting on that region before performing the region setting changes. After having such an action acknowledged the code must also have to instigate a suitable delay before then acting.

An example of this can be an LCD controller dealing with a frame buffer that is switching between a Normal world and Secure world use.

NOTE

There is no direct mechanism to ascertain if there are any outstanding AXI transactions, and so the designer must use their system knowledge to apply reasonable mechanisms.

It is recommended that any DECERR, or TZASC interrupt handler is designed to expect, and potentially ignore events generated under these circumstances.

Chapter 11
Supplement Configuration Unit

11.1 Introduction

The supplement configuration unit provides chip-specific configuration and status registers for the device. It is the chip defined module for extending the device configuration unit (DCFG) module. It provides a set of CCSR registers in addition to those available in the device configuration unit. There are no source and target IDs associated with this unit.

SCFG contains the following registers:

- Chip configuration registers
- Pinmux control registers
- Spare registers

11.2 SCFG register descriptions

11.2.1 SCFG Memory map

SCFG base address: 157_0000h

Offset	Register	Width (In bits)	Access	Reset value
70h	USB3 parameter 1 control register (USB3PRM1CR)	32	RW	2767_2B2Ah
74h	USB3 parameter 2 control register (USB3PRM2CR)	32	RW	17C1_FF48h
78h	USB3 parameter 3 control register (USB3PRM3CR)	32	RW	0019_0000h
130h	Core0 soft reset register (CORE0_SFT_RST)	32	RW	0000_0000h
154h	FTM chain configuration register (FTM_CHAIN_CONFIG)	32	RW	0000_0000h

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
158h	Alternate CBAR register (ALTCBAR)	32	RW	0000_0000h
15Ch	QSPI configuration register (QSPI_CFG)	32	RW	1000_0000h
164h	WR_QoS1 register (WR_QoS1)	32	RW	0000_0000h
168h	WR QoS2 register (WR_QoS2)	32	RW	0000_0000h
16Ch	RD QoS1 register (RD_QoS1)	32	RW	0000_0000h
170h	RD QoS2 register (RD_QoS2)	32	RW	0000_0000h
1A4h	Snoop configuration control register (SNPCNFGCR)	32	RW	0000_0000h
204h	Core soft reset enable control register (CORESRENCR)	32	RW	0000_0000h
220h	Core reset vector base address register 0 (RVBAR0_0)	32	RW	0000_0000h
224h	Core reset vector base address register 0 (RVBAR0_1)	32	RW	0000_0000h
240h	Core low power mode control status register (LPMCSR)	32	RW	0000_0000h
408h	SDHC IO VSEL control register (SDHCIOVSELCR)	32	RW	0000_0000h
414h	USB powerfault select register (USB_PWRFAULT_SELCR)	32	RW	0000_0000h
418h	USBPHY control register (USB_PHY_CTRL)	32	RW	8000_009Ch
42Ch	Cluster PM control register (CLUSTERPMCR)	32	RW	0000_0000h
430h	Pinmux control register (PMUXCR0)	32	RW	0000_0000h
434h	RGMII port control register (RGMIIPCR)	32	RW	FFFE_0000h
43Ch	RGMII port status register (RGMIIPSR)	32	RO	0000_0000h
440h	PFE PCS status register 1 (PFEPCSSR1)	32	RO	0000_0000h
444h	PFE interrupt enable control register (PFEINTENCR1)	32	RW	0000_0000h
448h	PFE PCS status register 2 (PFEPCSSR2)	32	RO	0000_0000h
44Ch	PFE interrupt enable control register 2 (PFEINTENCR2)	32	RW	0000_0000h
450h	PFE error control register (PFEERRCR)	32	RW	0000_0000h
454h	PFE error interrupt enable control register (PFEERRINTENCR)	32	W1C	0000_0000h
458h	PFEA sideband control register (PFEASBCR)	32	RW	3C00_0000h
45Ch	PFEB sideband control register (PFEBSBCR)	32	RW	3C00_0000h
484h	MDIO select control register (MDIOSELCR)	32	RW	0000_0000h
500h - 51Ch	Spare control register (SPARECR1 - SPARECR8)	32	RW	0000_0000h
520h	I2C debug mode control register (I2CDBGCR)	32	RW	0000_0000h
600h - 60Ch	Scratch read write register (SCRATCHRW1 - SCRATCHRW4)	32	RW	0000_0000h
680h	Core boot control register (COREBCR)	32	RW	FFFF_FFFFh
2000h	Shared message signaled interrupt index register (PEX1MSIIR)	32	RW	0000_0000h
2004h	Shared message signaled interrupt register (PEX1MSIR)	32	RO	0000_0000h

11.2.2 USB3 parameter 1 control register (USB3PRM1CR)

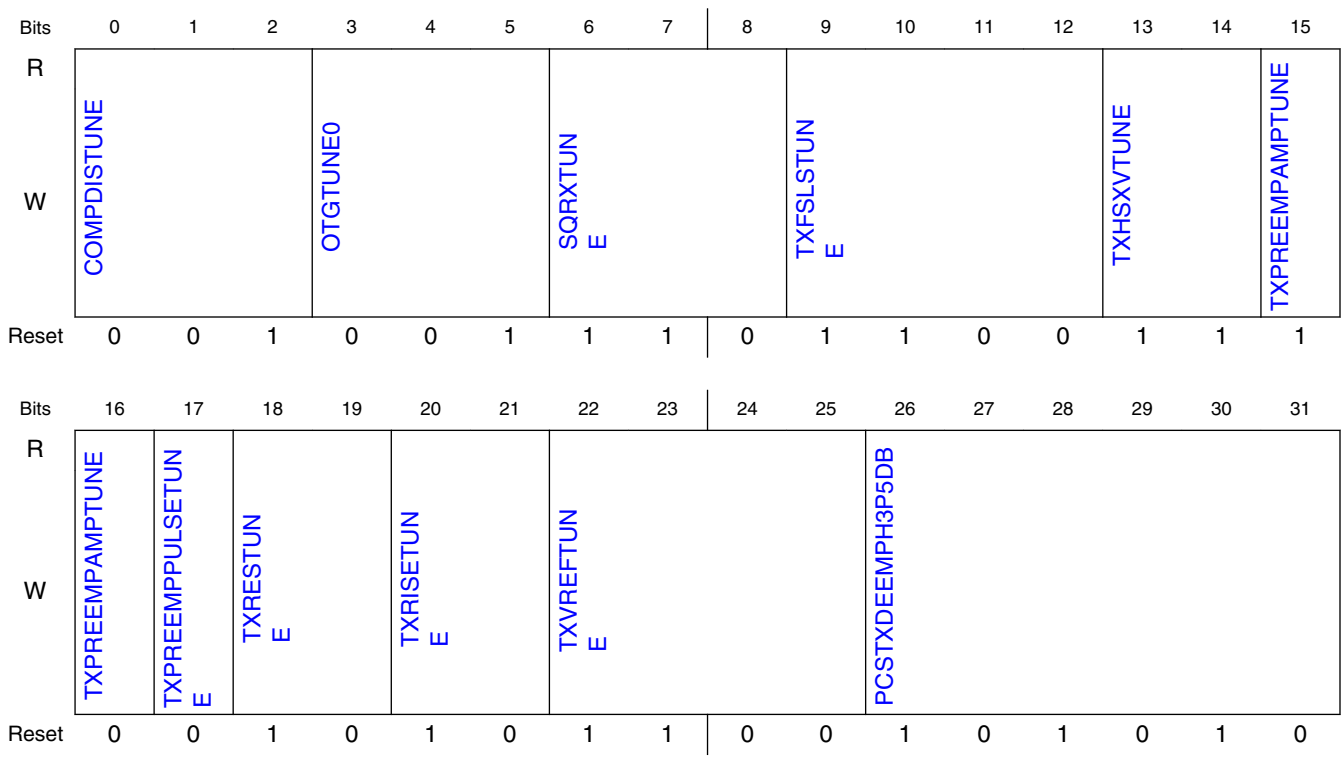
11.2.2.1 Offset

Register	Offset
USB3PRM1CR	70h

11.2.2.2 Function

USB3PRM1CR contains the USB3 parameters signals.

11.2.2.3 Diagram



11.2.2.4 Fields

Field	Function
0-2 COMPDISTUNE	It drives the value of USB3 COMPDISTUNE signal. Disconnect threshold adjustment: It adjusts the voltage level for the threshold used to detect a disconnect event at the host.

Table continues on the next page...

Field	Function
3-5 OTGTUNE0	It drives the value of USB3 OTGTUNE0 signal. V_{BUS} valid threshold adjustment: This bus adjusts the voltage level for the V_{BUS} valid threshold.
6-8 SQRXTUNE	It drives the value of USB3 SQRXTUNE signal. Squelch threshold adjustment: It adjusts the voltage level for the threshold used to detect valid high speed data.
9-12 TXFSLSTUNE	It drives the value of USB3 TXFSLSTUNE signal. FS/LS source impedance adjustment: It adjusts the low and full speed single-ended source impedance while driving high.
13-14 TXHSXVTUNE	It drives the value of USB3 TXHSXVTUNE signal. Transmitter high speed crossover adjustment: This bus adjusts the voltage at which the D+ and D- signals cross while transmitting in HS mode. 00b - Reserved 01b - – 15 mV 10b - + 15 mV 11b - Default
15-16 TXPREEMPAMPTUNE	It drives the value of USB3 TXPREEMPAMPTUNE signal. HS transmitter pre-emphasis current control: This signal controls the amount of current sourced to D+ and D- after a J-to-K or K-to-J transition. The HS transmitter pre-emphasis current is defined in terms of unit amounts. One unit amount is approximately 600 µA and is defined as 1X pre-emphasis current. 00b - HS transmitter pre-emphasis disabled (default) 01b - HS transmitter pre-emphasis circuit sources 1X preemphasis current 10b - HS transmitter pre-emphasis circuit sources 2X pre-emphasis current 11b - HS transmitter pre-emphasis circuit sources 3X pre-emphasis current
17 TXPREEMPPULSETUNE	It drives the value of USB3 TXPREEMPPULSETUNE signal. HS transmitter pre-emphasis duration control: This signal controls the duration for which the HS pre-emphasis current is sourced onto D+ or D-. The HS transmitter pre-emphasis duration is defined in terms of unit amounts. One unit of pre-emphasis duration is approximately 580 ps and is defined as 1X pre-emphasis duration. This signal is valid only if either TXPREEMPAMPTUNE[1] or TXPREEMPAMPTUNE[0] is set to 1'b1. 0b - 2X, long pre-emphasis current duration (default) 1b - 1X, short pre-emphasis current duration
18-19 TXRESTUNE	It drives the value of USB3 TXRESTUNE signal. USB source impedance adjustment: This bus adjusts the driver source impedance to compensate for added series resistance on the USB. 00b - Source impedance is increased by approximately 1.5 Ω 01b - Default 10b - Source impedance is decreased by approximately 2 Ω 11b - Source impedance is decreased by approximately 4 Ω
20-21 TXRISETUNE	It drives the value of USB3 TXRISETUNE signal. HS transmitter rise/fall time adjustment: It adjusts the rise/fall times of the high-speed waveform.
22-25 TXVREFTUNE	It drives the value of USB3 TXVREFTUNE signal. HS DC voltage level adjustment: It adjusts the high speed DC level voltage.
26-31 PCSTXDEEMPH3P5DB	It drives the value of USB3 pcs_tx_deemph_3p5db (Tx de-emphasis at 3.5 dB) signal.

11.2.3 USB3 parameter 2 control register (USB3PRM2CR)

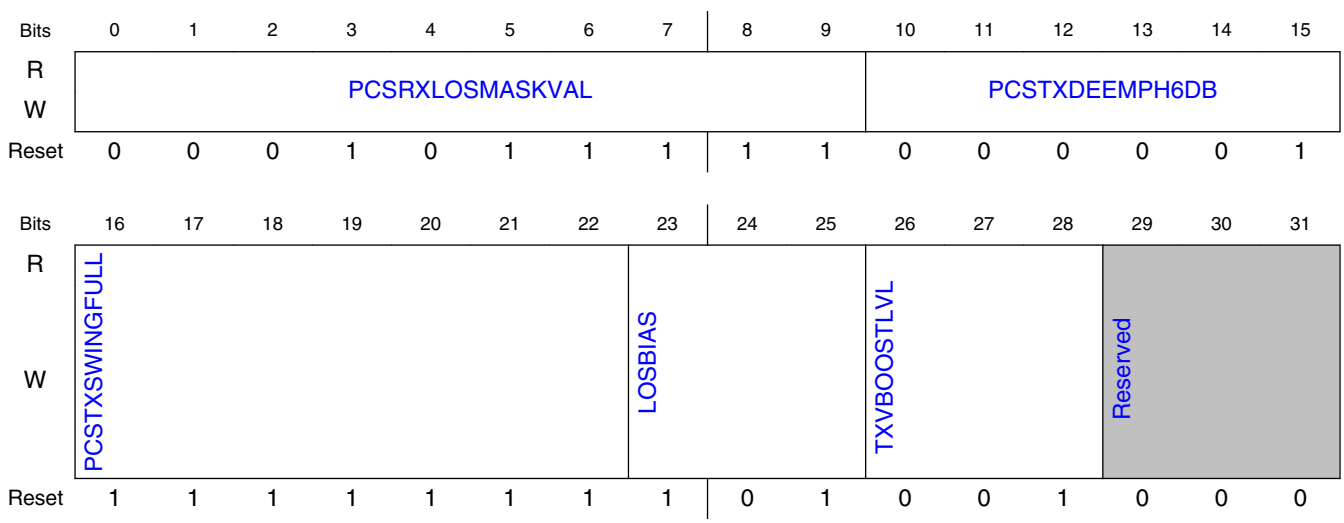
11.2.3.1 Offset

Register	Offset
USB3PRM2CR	74h

11.2.3.2 Function

USB3PRM2CR contains the USB3 parameters signals.

11.2.3.3 Diagram



11.2.3.4 Fields

Field	Function
0-9 PCSRXLOSMASKVAL	It drives the value of USB3 pcs_rx_los_mask_val signal. Configurable loss-of-signal mask width: It sets the number of reference clock cycles to mask the incoming LFPS in U3 and U2 states. It masks the incoming LFPS for the number of reference clock cycles equal to the value of pcs_rx_los_mask_val[9:0]. This control filters out short, non-compliant LFPS glitches sent by a non-compliant host.

Table continues on the next page...

Field	Function
	If this bus is set to 10'b0, it disables masking. This bus should be accessible to general configuration registers for system testing and debug. The value should be defined only in reset. Changing this value during operation might disrupt normal operation of the link.
10-15 PCSTXDEEMPH6DB	It drives the value of USB3 pcs_tx_deemph_6db signal. Tx de-emphasis at 6 dB: This bus is provided for completeness and as a second potential launch amplitude.
16-22 PCSTXSWINGFULL	It drives the value of USB3 pcs_tx_swing_full signal. Tx amplitude (full swing mode): This static value sets the launch amplitude of the transmitter.
23-25 LOSBIAS	It drives the value of USB3 los_bias signal. Loss-of-signal detector threshold level control: It sets the LOS detection threshold level. A positive binary bit setting change results in a +15 mVp incremental change in the LOS threshold. A negative binary bit setting change results in a –15 mVp incremental change in the LOS threshold. The 3b'000 setting is reserved and must not be used.
26-28 TXVBOOSTLVL	It drives the value of USB3 tx_vboost_lvl signal. Tx voltage boost level: It sets the boosted transmit launch amplitude (mVppd).
29-31 —	Reserved

11.2.4 USB3 parameter 3 control register (USB3PRM3CR)

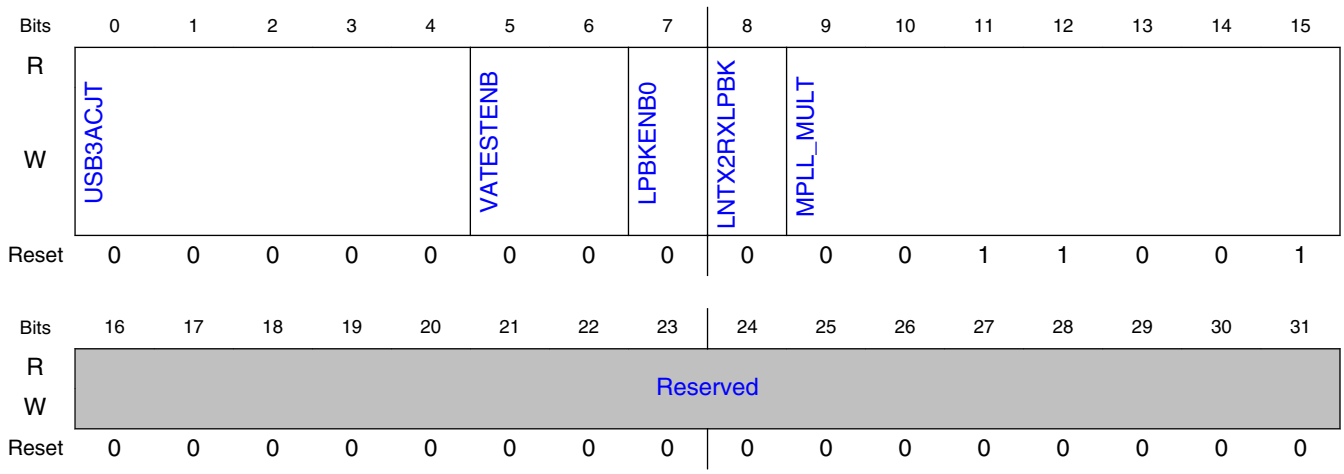
11.2.4.1 Offset

Register	Offset
USB3PRM3CR	78h

11.2.4.2 Function

USB3PRM3CR contains the USB3 parameters signals.

11.2.4.3 Diagram



11.2.4.4 Fields

Field	Function
0-4 USB3ACJT	It drives the value of USB3 acjt_level signal. Receiver sensitivity level control: It selects the sensitivity level for the 1149.6 receiver for AC JTAG. The level is set based on the amplitude of the incoming step during AC JTAG boundary scan testing.
5-6 VATESTENB	It drives the value of USB3 PHY VATESTENB signal. Analog test pin select: It enables analog test voltages to be placed on the ID pin. 00b - Analog test voltages cannot be viewed or applied on ID. 01b - Analog test voltages can be viewed or applied on ID. 10-11b - Reserved
7 LPBKENB0	It drives the value of USB3 PHY LOOPBACKENB0 signal. Loopback test enable: This controller signal places the USB3 PHY in HS loopback mode, which concurrently enables the HS receive and transmit logic. 0b - During HS data transmission, the HS receive logic is disabled. 1b - During HS data transmission, the HS receive logic is enabled.
8 LNTX2RXLPBK	It drives the value of USB3 lane0_tx2rx_loopbk signal. Loopback: When this signal is asserted, data from the transmit predriver is looped back to the receive slicers. LOS is bypassed and based on the tx0_en input so that rx0_los = !tx0_data_en.
9-15 MPLL_MULT	It drives the value of USB3 mpll_multiplier signal. MPLL frequency multiplier control: It multiplies the reference clock to a frequency suitable for intended operating speed.
16-31 —	Reserved

11.2.5 Core0 soft reset register (CORE0_SFT_RST)

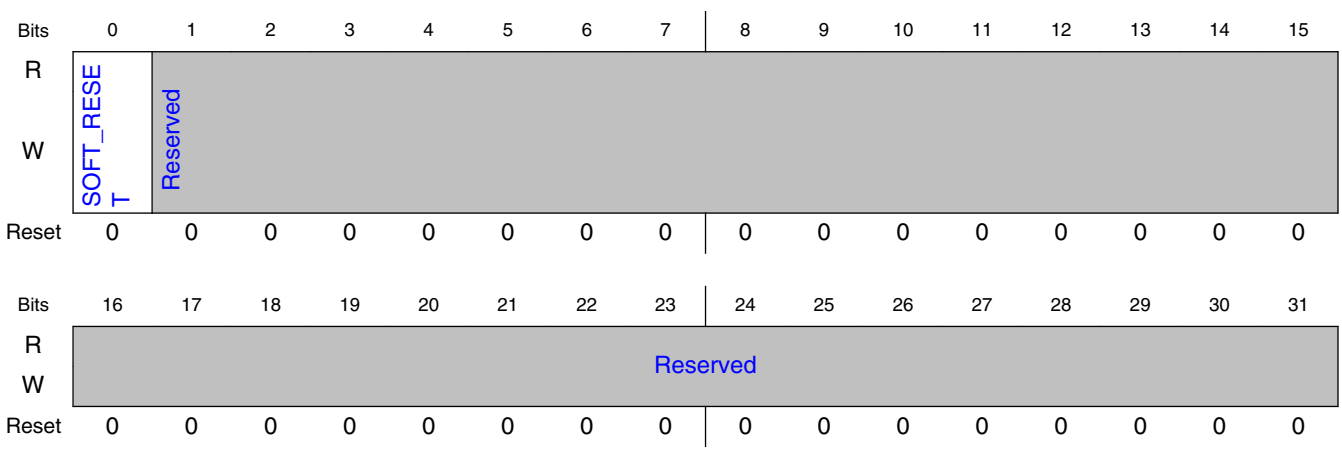
11.2.5.1 Offset

Register	Offset
CORE0_SFT_RST	130h

11.2.5.2 Function

This register contains bits to provide the soft reset to core 0 of A53 complex.

11.2.5.3 Diagram



11.2.5.4 Fields

Field	Function
0 SOFT_RESET	Core 0 soft reset request. This bit is self clearing. 0b - default 1b - Soft reset core 0
1-31 —	Reserved Reserved

11.2.6 FTM chain configuration register (FTM_CHAIN_CONFIG)

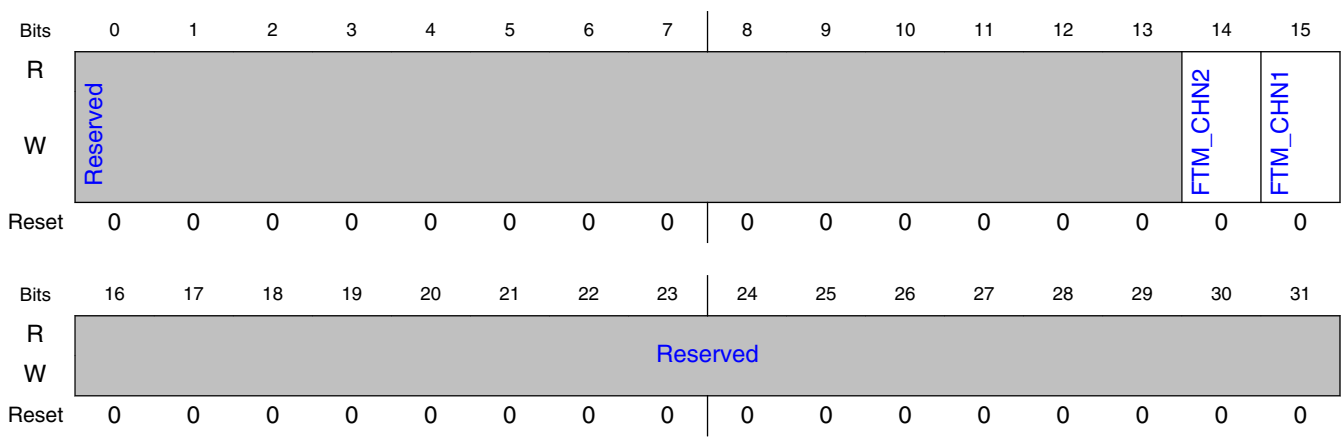
11.2.6.1 Offset

Register	Offset
FTM_CHAIN_CONFIG	154h

11.2.6.2 Function

This register contains bits to enable chaining of FlexTimers.

11.2.6.3 Diagram



11.2.6.4 Fields

Field	Function
0-13 —	Reserved Reserved
14 FTM_CHN2	FTM2 chaining 0b - FTM2 is not chained 1b - FTM2 is chained
15	FTM1 chaining

Table continues on the next page...

Field	Function
FTM_CHN1	0b - FTM1 is not chained 1b - FTM1 is chained
16-31 —	Reserved Reserved

11.2.7 Alternate CBAR register (ALTCBAR)

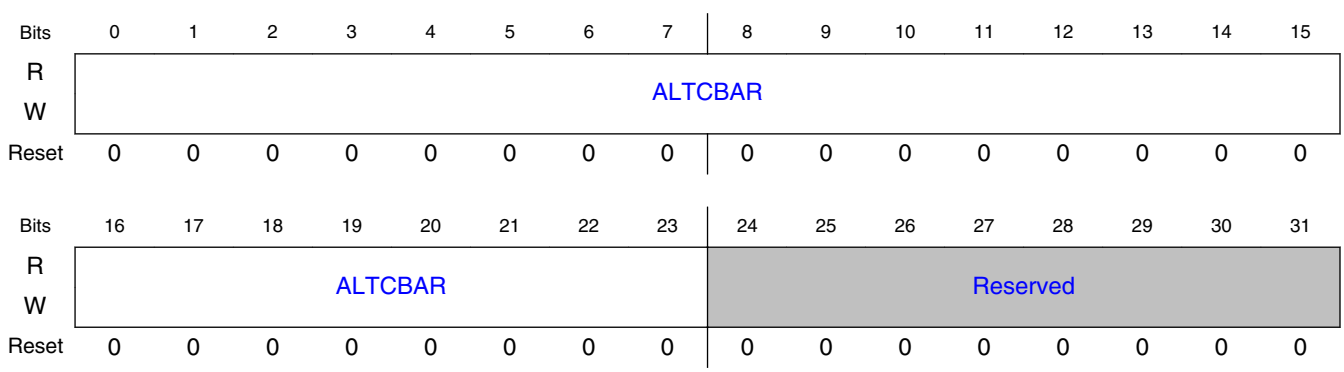
11.2.7.1 Offset

Register	Offset
ALTCBAR	158h

11.2.7.2 Function

This register contains bits for configuration of alternate base address register for PBL to access the address space of more than 32 MB.

11.2.7.3 Diagram



11.2.7.4 Fields

Field	Function
0-23	Alternate configuration base address register for PBL

Table continues on the next page...

Field	Function
ALTCBAR	
24-31 —	Reserved

11.2.8 QSPI configuration register (QSPI_CFG)

11.2.8.1 Offset

Register	Offset
QSPI_CFG	15Ch

11.2.8.2 Function

This register contains bits for selecting the clock source for QuadSPI and enabling or disabling the QuadSPI clock. It also provides bits for using different ratios of core PLL to drive the QuadSPI clock.

11.2.8.3 Diagram

Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CLK_SEL				Reserved				Reserved							
Reset	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

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

11.2.8.4 Fields

Field	Function
0-3 CLK_SEL	Clock select This bit provides the settings for QuadSPI clock division. 0000b - Cluster PLL/256 0001b - Cluster PLL/64 (default) 0010b - Cluster PLL/32 0011b - Cluster PLL/24 0100b - Cluster PLL/20 0101b - Cluster PLL/16 0110b - Cluster PLL/12 0111b - Cluster PLL/8
4-7 —	Reserved
8-31 —	Reserved

11.2.9 WR_QoS1 register (WR_QoS1)

11.2.9.1 Offset

Register	Offset
WR_QoS1	164h

11.2.9.2 Function

This register contains bits for write port QoS inputs to CCI-400. Note that the bits are described as [x:x+3]; but these are connected to QoS ports as [3:0]. So, there is a bit reversal involved.

11.2.9.3 Diagram

Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Reserved				PFE1_QOS				PFE2_QOS				Reserved			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Reserved				PEX1_QOS				SEC_QOS				Reserved			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

11.2.9.4 Fields

Field	Function
0-3 —	Reserved
4-7 PFE1_QOS	QoS[3:0] for PFE DDR AXI master interface
8-11 PFE2_QOS	QoS[3:0] for PFE HDBUD master interface
12-15 —	Reserved
16-19 —	Reserved
20-23 PEX1_QOS	QoS[3:0] for PEX1
24-27 SEC_QOS	QoS[3:0] for SEC
28-31 —	Reserved

11.2.10 WR QoS2 register (WR_QoS2)

11.2.10.1 Offset

Register	Offset
WR_QoS2	168h

11.2.10.2 Function

This register contains bits for write port QoS inputs to CCI-400. Note that the bits are described as [x:x+3]; but these are connected to QoS ports as [3:0]. So, there is a bit reversal involved.

11.2.10.3 Diagram

Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	USB3_QOS				Reserved								A53_QOS			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ESDHC2_QOS				ESDHC1_QOS				USB2_QOS				SATA_QOS			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

11.2.10.4 Fields

Field	Function
0-3 USB3_QOS	QoS[3:0] for USB3
4-11 —	Reserved
12-15 A53_QOS	QoS[3:0] for A53
16-19 ESDHC2_QOS	QoS[3:0] for eSDHC2
20-23 ESDHC1_QOS	QoS[3:0] for eSDHC1

Table continues on the next page...

Field	Function
24-27 USB2_QOS	QoS[3:0] for USB2
28-31 SATA_QOS	QoS[3:0] for SATA

11.2.11 RD QoS1 register (RD_QoS1)

11.2.11.1 Offset

Register	Offset
RD_QoS1	16Ch

11.2.11.2 Function

This register contains bits for read port QoS inputs to CCI-400. Note that the bits are described as [x:x+3]; but these are connected to QoS ports as [3:0]. So, there is a bit reversal involved.

11.2.11.3 Diagram

Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Reserved				PFE1_QOS				PFE2_QOS				Reserved			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Reserved				PEX1_QOS				SEC_QOS				Reserved			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

11.2.11.4 Fields

Field	Function
0-3 —	Reserved
4-7 PFE1_QOS	QoS[3:0] for PFE DDR AXI master interface
8-11 PFE2_QOS	QoS[3:0] for PFE HDBUS master interface
12-19 —	Reserved
20-23 PEX1_QOS	QoS[3:0] for PEX1
24-27 SEC_QOS	QoS[3:0] for SEC
28-31 —	Reserved

11.2.12 RD QoS2 register (RD_QoS2)

11.2.12.1 Offset

Register	Offset
RD_QoS2	170h

11.2.12.2 Function

This register contains bits for read port QoS inputs to CCI-400. Note that the bits are described as [x:x+3]; but these are connected to QoS ports as [3:0]. So, there is a bit reversal involved.

11.2.12.3 Diagram

Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	USB3_QOS				Reserved								A53_QOS			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ESDHC2_QOS				ESDHC1_QOS				USB2_QOS				SATA_QOS			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

11.2.12.4 Fields

Field	Function
0-3 USB3_QOS	QoS[3:0] for USB3
4-11 —	Reserved
12-15 A53_QOS	QoS[3:0] for A53
16-19 ESDHC2_QOS	QoS[3:0] for eSDHC2
20-23 ESDHC1_QOS	QoS[3:0] for eSDHC1
24-27 USB2_QOS	QoS[3:0] for USB2
28-31 SATA_QOS	QoS[3:0] for SATA

11.2.13 Snoop configuration control register (SNPCNFGCR)

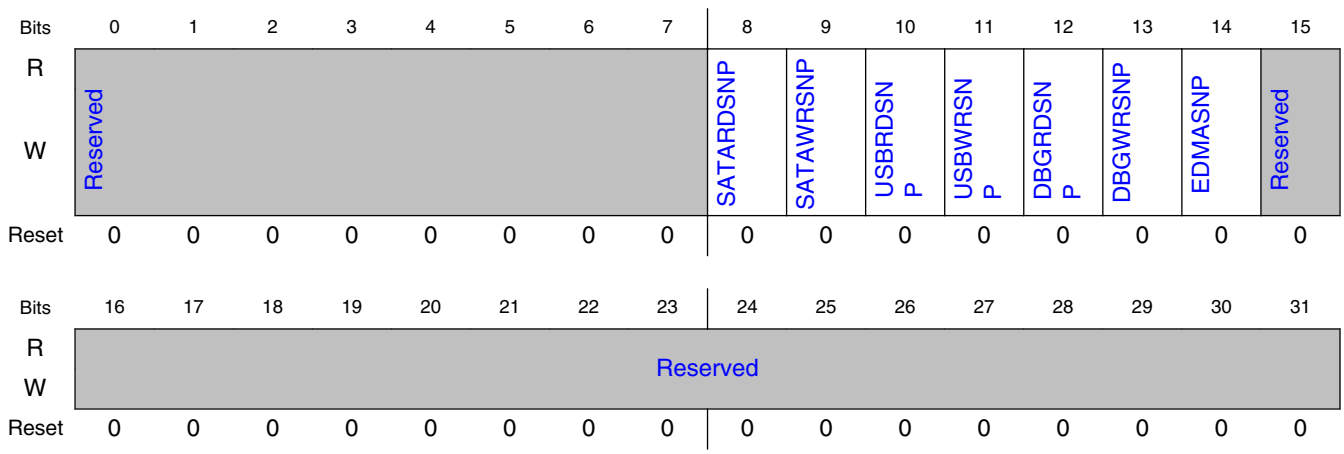
11.2.13.1 Offset

Register	Offset
SNPCNFGCR	1A4h

11.2.13.2 Function

This register contains bits to drive snoop signal for various masters.

11.2.13.3 Diagram



11.2.13.4 Fields

Field	Function
0-7 —	Reserved
8 SATARDSNP	Drives SATA read snoop signal
9 SATAWRSNP	Drives SATA write snoop signal
10 USBRDSNP	Drives USB3 read snoop signal
11 USBWRSNP	Drives USB3 write snoop signal
12 DBGRDSNP	Drives Debug read snoop signal
13 DBGWRSNP	Drives Debug write snoop signal
14	Drives eDMA snoop signal

Table continues on the next page...

Field	Function
EDMASNP	
15-31 —	Reserved

11.2.14 Core soft reset enable control register (CORESRENCR)

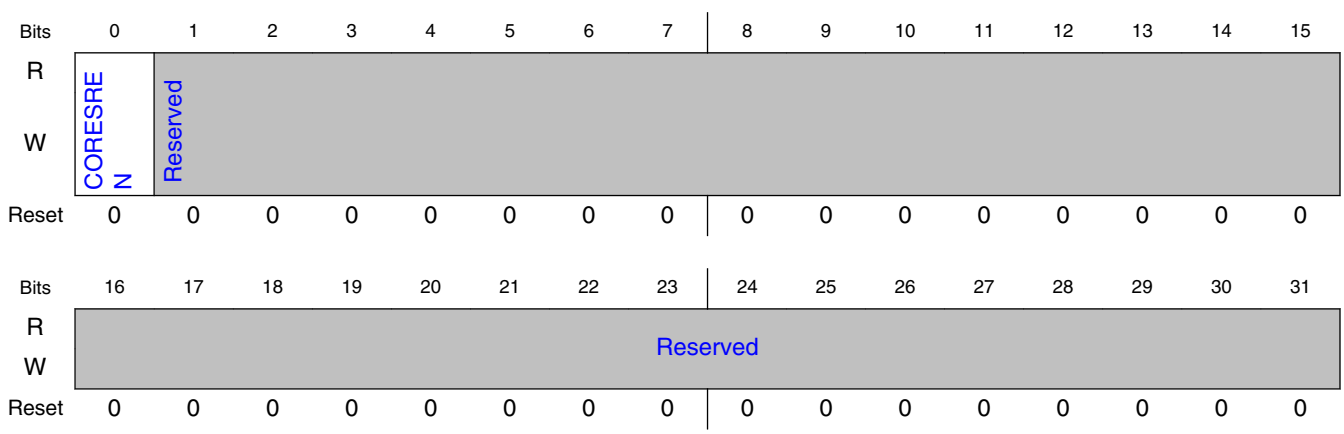
11.2.14.1 Offset

Register	Offset
CORESRENCR	204h

11.2.14.2 Function

This register contains the bit to enable the soft reset functionality.

11.2.14.3 Diagram



11.2.14.4 Fields

Field	Function
0	Enable/disable the core soft reset functionality

Table continues on the next page...

Field	Function
CORESREN	0b - Core soft-reset is disabled (default) 1b - Core soft-reset is enabled
1-31 —	Reserved

11.2.15 Core reset vector base address register 0 (RVBAR0_0)

11.2.15.1 Offset

Register	Offset
RVBAR0_0	220h

11.2.15.2 Function

This register controls the reset vector base address for core for bits [33:2]. This register should be programmed in the PBI phase.

NOTE

After core warm reset, the core can boot from a reset vector configured in SCFG_RVBAR0_0 (Core reset vector base address 0) and SCFG_RVBAR0_1 (Core reset vector base address 1).

The 40-b address should be programmed in RVBAR0_0/ RVBAR0_1 register for the entry point of the vector from where the core executes the first instruction after coming out from the warm-reset. If not configured, its reset value will be 0x0 and the first instruction will be executed from the internal boot ROM.

11.2.15.3 Diagram

Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	RVBAR0_0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RVBAR0_0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

11.2.15.4 Fields

Field	Function
0-31 RVBAR0_0	Core reset vector base address for bits [33:2].

11.2.16 Core reset vector base address register 0 (RVBAR0_1)

11.2.16.1 Offset

Register	Offset
RVBAR0_1	224h

11.2.16.2 Function

This register controls the reset vector base address for core for bits [33:2]. This register should be programmed in the PBI phase.

NOTE

After core warm reset, the core can boot from a reset vector configured in SCFG_RVBAR0_0 (Core reset vector base address 0) and SCFG_RVBAR0_1 (Core reset vector base address 1).

The 40-b address should be programmed in RVBAR0_0/ RVBAR0_1 register for the entry point of the vector from where the core executes the first instruction after coming out from the warm-reset. If not configured, its reset value will be 0x0 and the first instruction will be executed from the internal boot ROM.

11.2.16.3 Diagram

Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	RVBAR0_1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	RVBAR0_1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

11.2.16.4 Fields

Field	Function
0-31 RVBAR0_1	Core reset vector base address for bits [39:34].

11.2.17 Core low power mode control status register (LPMCSR)

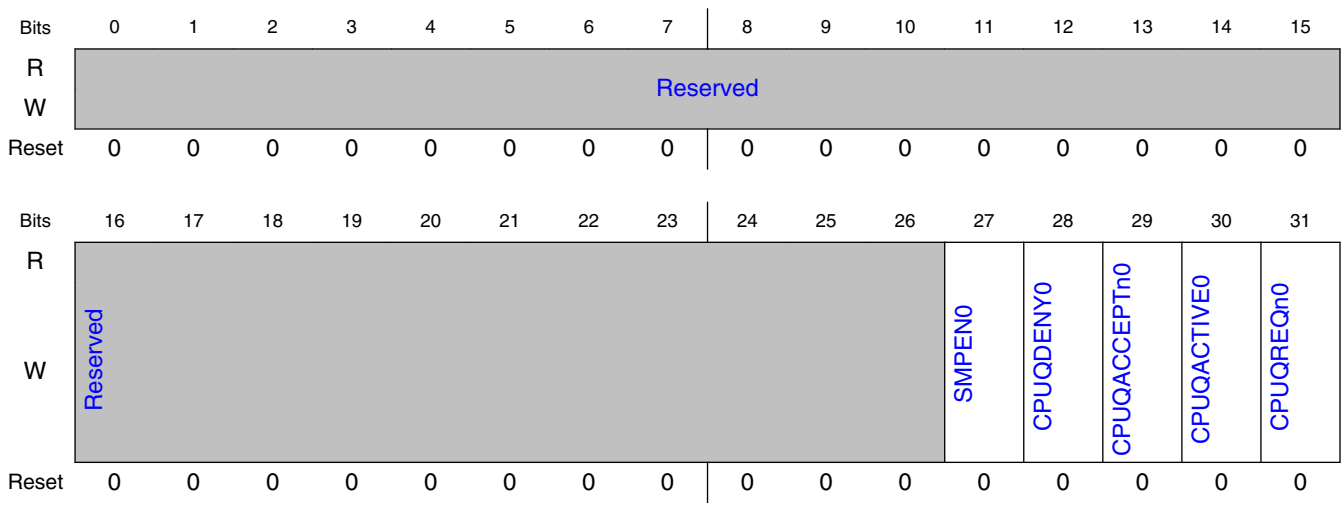
11.2.17.1 Offset

Register	Offset
LPMCSR	240h

11.2.17.2 Function

This register provides status and control bits for various signals on A53.

11.2.17.3 Diagram



11.2.17.4 Fields

Field	Function
0-26 —	Reserved
27 SMPEN0	Status bit for SMPEN signal of core 0.
28 CPUQDENY0	Status bit for CPUQDENY signal for core 0.
29 CPUQACCEPTn0	Status bit for CPUQACCEPTn signal for core 0.

Table continues on the next page...

Field	Function
30 CPUQACTIVE0	Status bit for CPUQACTIVE signal for core 0.
31 CPUQREQn0	Control bit for CPUQREQn for core 0.

11.2.18 SDHC IO VSEL control register (SDHCIOVSELCR)

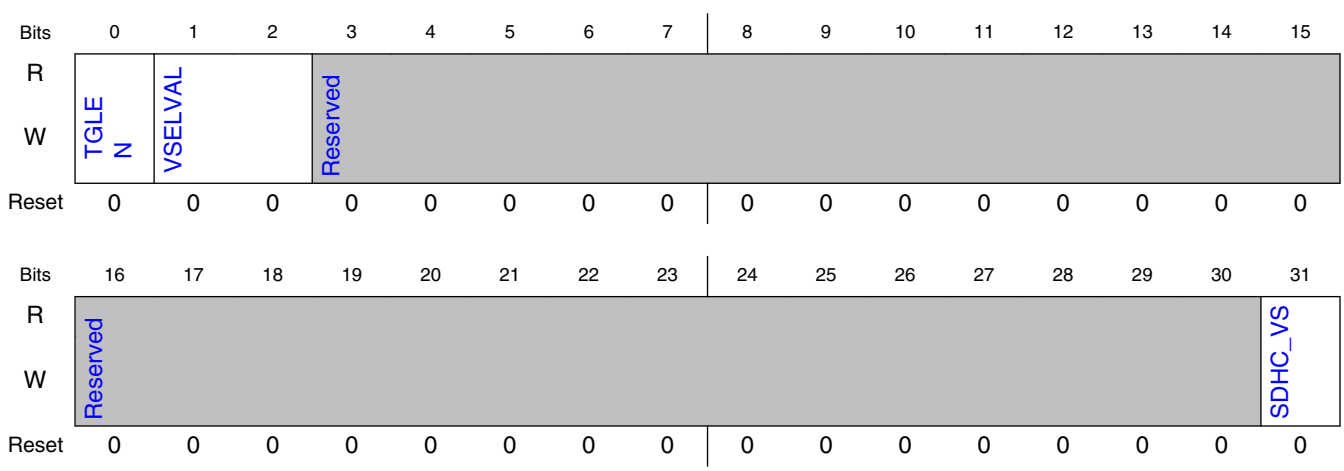
11.2.18.1 Offset

Register	Offset
SDHCIOVSELCR	408h

11.2.18.2 Function

This register contains bits to support SDHC IO voltage switching.

11.2.18.3 Diagram



11.2.18.4 Fields

Field	Function
0 TGLEN	SDHC IO voltage switching enable 0b - Voltage switching not enabled (default) 1b - Voltage switching enabled
1-2 VSELVAL	Configures voltage for SDHC (if enabled by TGLEN) 00b - 1.8V 01b - Reserved 10b - 3.3V 11b - Auto-voltage-selection enabled
3-30 —	Reserved
31 SDHC_VS	SCFG bit reflecting shadow bit controlled by software for eSDHC_PROCTL[VOLT_SEL] 0b - Change the SD bus supply voltage to high voltage range (3.3V) 1b - Change the SD bus supply voltage to low voltage range (1.8V)

11.2.19 USB powerfault select register (USB_PWRFAULT_SEL CR)

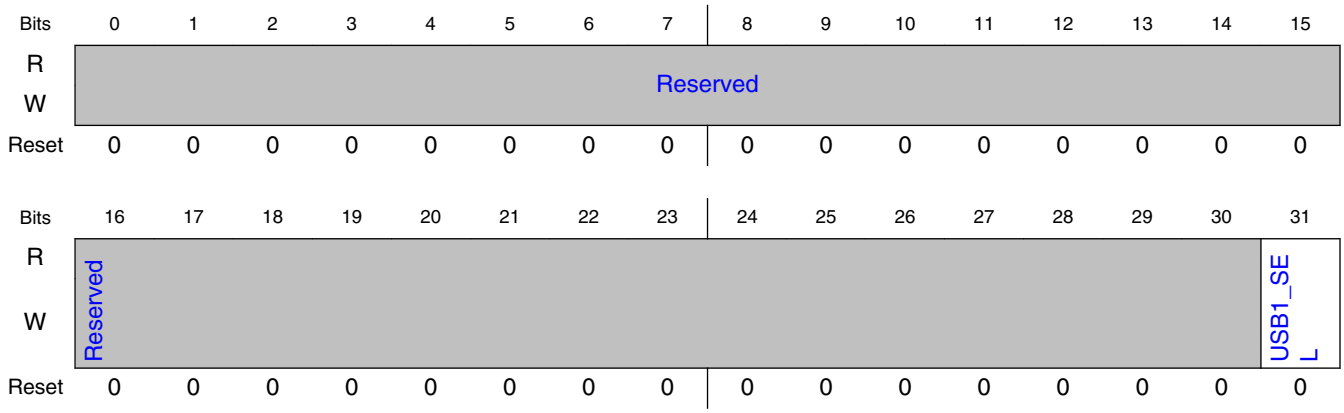
11.2.19.1 Offset

Register	Offset
USB_PWRFAULT_SEL CR	414h

11.2.19.2 Function

This register defines the sampling of USB1_PWRFAULT by the USB1 controller.

11.2.19.3 Diagram



11.2.19.4 Fields

Field	Function
0-30 —	Reserved
31 USB1_SEL	USB1 controller connectivity 0b - PWRFAULT pin is available only for USB 3.0 and not for the USB 2.0 interface (default) 1b - PWRFAULT pin is shared between the USB 3.0 and USB 2.0 interfaces.

11.2.20 USBPHY control register (USB_PHY_CTRL)

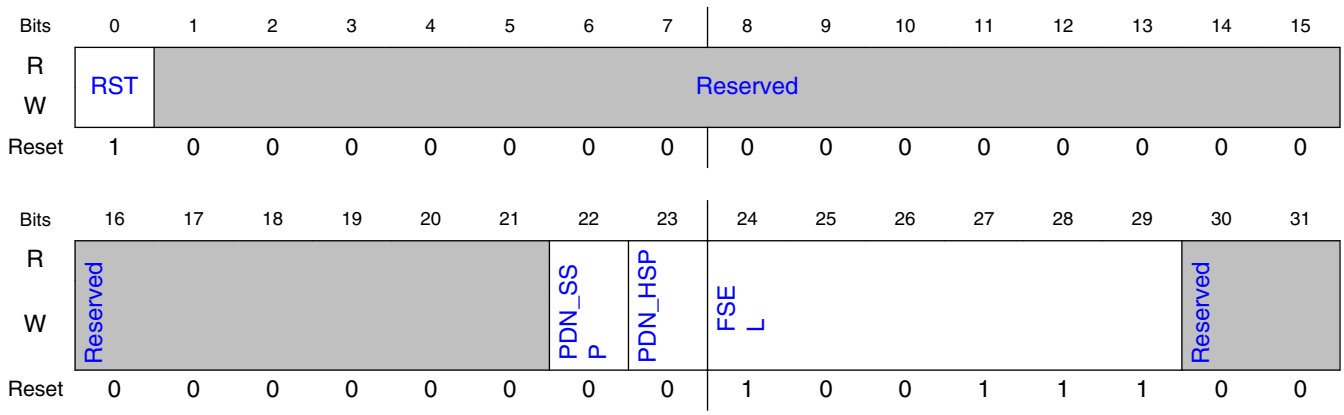
11.2.20.1 Offset

Register	Offset
USB_PHY_CTRL	418h

11.2.20.2 Function

This register contains control bits for USBPHY.

11.2.20.3 Diagram



11.2.20.4 Fields

Field	Function
0 RST	Active high reset for USBPHY. Write 0 to bring the USBPHY out of reset. 0b - Reset is de-asserted 1b - Reset is asserted (default)
1-21 —	Reserved
22 PDN_SSP	This bit forces the super speed function to lowest power state. Default value 0.
23 PDN_HSP	This bit forces the high speed function to lowest power state. Default value 0.
24-29 FSEL	USB PHY 1 fsel[5:0] is connected to this field bit 24 -- fsel bit 5 bit 25 -- fsel bit 4 bit 26 -- fsel bit 3 bit 27 -- fsel bit 2 bit 28 -- fsel bit 1 bit 29 -- fsel bit 0
30-31 —	Reserved

11.2.21 Cluster PM control register (CLUSTERPMCR)

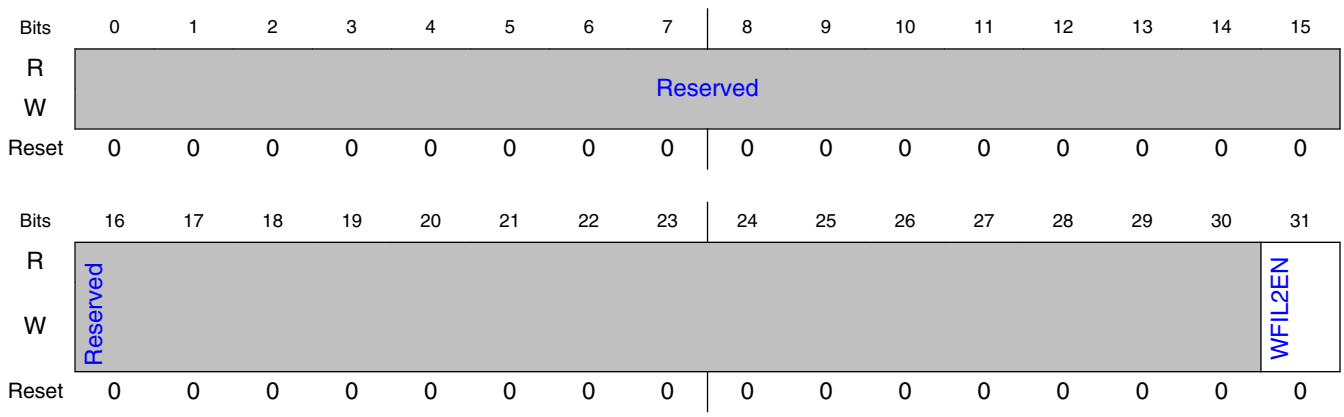
11.2.21.1 Offset

Register	Offset
CLUSTERPMCR	42Ch

11.2.21.2 Function

This register contains the control bit to enable WFIL2.

11.2.21.3 Diagram



11.2.21.4 Fields

Field	Function
0-30 —	Reserved
31 WFIL2EN	WFIL2 enable 0b - No action (default) 1b - STANDBYWFIL2 gets asserted when the core executes STANBYWFI

11.2.22 Pinmux control register (PMUXCR0)

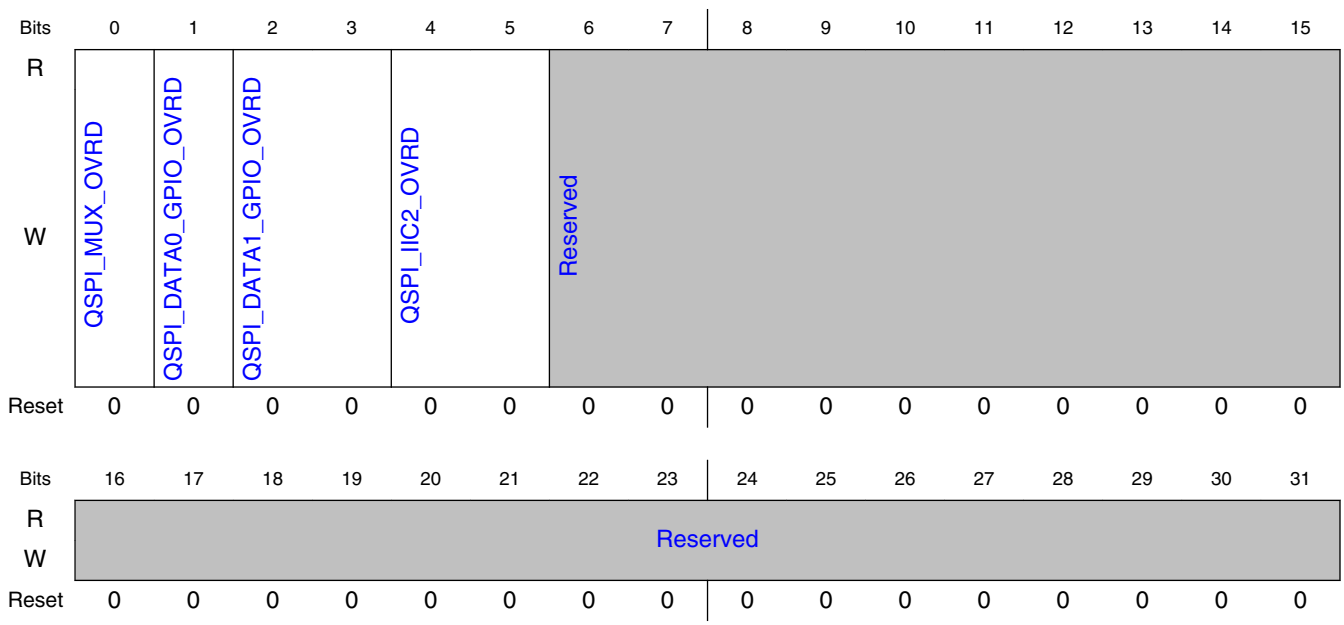
11.2.22.1 Offset

Register	Offset
PMUXCR0	430h

11.2.22.2 Function

This register contains bits to change the QuadSPI pin multiplexing by software.The chip can use QuadSPI for boot and then use these IOs for GPIO.

11.2.22.3 Diagram



11.2.22.4 Fields

Field	Function
0 QSPI_MUX_OVRD	Software override for QuadSPI multiplexing. 0b - Override disabled (default) 1b - Override enabled
1	Software configures alternate functionality of the QuadSPI pins for 1-bit interface. 0b - QSPI_A_DAT0/QSPI_A_SCK/QSPI_A_CS0

Table continues on the next page...

Field	Function
QSPI_DATA0_GPIO_OVRD	1b - GPIO1[4]/GPIO1[11]/GPIO1[5]
2-3 QSPI_DATA1_GPIO_OVRD	Software configures alternate functionality of the QuadSPI pins for 2-bit interface. 00b - QSPI_A_DAT1 01b - GPIO1[12] 10b - Reserved 11b - Reserved
4-5 QSPI_IIC2_OVRD	Software configures alternate functionality of the QuadSPI pins for additional 2-bit interface. **NOTE:** The QuadSPI multiplexing can be changed by software through SCFG registers. 00b - GPIO1[13], GPIO1[14] (default) 01b - Reserved 10b - QSPI_A_DATA2, QSPI_A_DATA3 11b - Reserved
6-31 —	Reserved

11.2.23 RGMII port control register (RGMIIPCR)

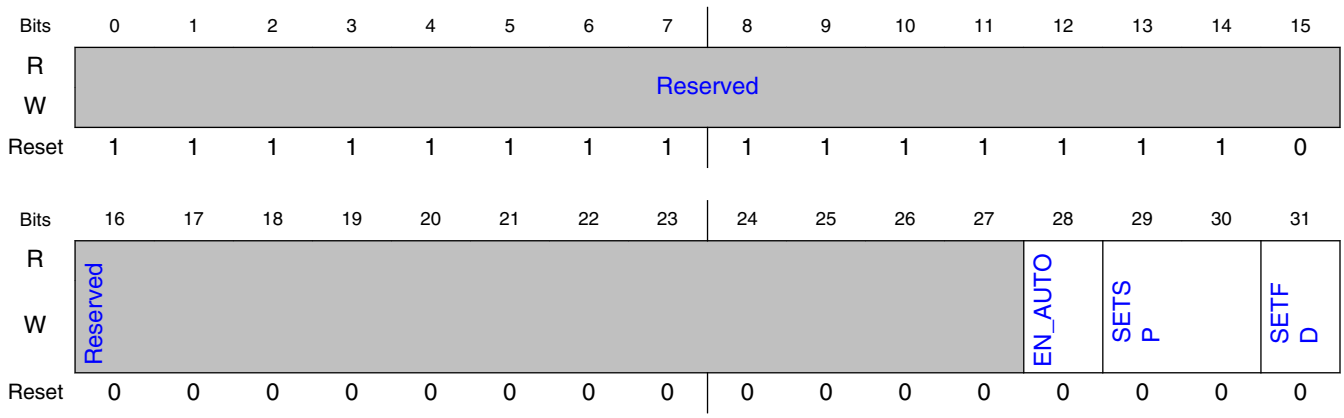
11.2.23.1 Offset

Register	Offset
RGMIIPCR	434h

11.2.23.2 Function

This register contains control bits of the RGMII ports.

11.2.23.3 Diagram



11.2.23.4 Fields

Field	Function
0-27 —	Reserved
28 EN_AUTO	Enable automatic speed selection 0b - SETSP and SETFD bits determine the RGMII link speed and duplex mode 1b - Enable automatic speed selection - RGMII PHY in-band status information is used to select the speed of operation
29-30 SETSP	Set RGMII link speed 00b - 100 Mbps RGMII (valid only if EN_AUTO =0) 01b - 10 Mbps RGMII (valid only if EN_AUTO =0) 10b - 1 Gbps RGMII (valid only if EN_AUTO =0) 11b - Reserved
31 SETFD	Set full duplex RGMII mode 0b - Reserved 1b - Forces full duplex RGMII mode (This bit is valid only if EN_AUTO =0)

11.2.24 RGMII port status register (RGMIIPSR)

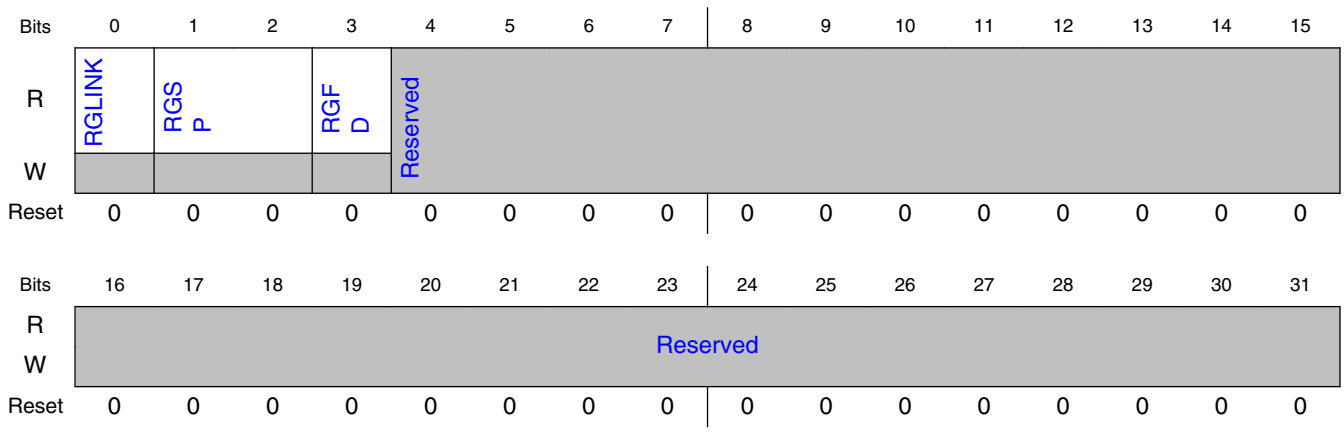
11.2.24.1 Offset

Register	Offset
RGMIIPSR	43Ch

11.2.24.2 Function

This register contains bits to provide status of the RGMII ports.

11.2.24.3 Diagram



11.2.24.4 Fields

Field	Function
0 RGLINK	RGMII link status 0b - No link is established or RGMII PHY does not support the optional in-band signaling 1b - A valid link is established by the RGMII PHY (This bit is valid if PHY supports the optional in-band signaling)
1-2 RGSP	RGMII link speed status 00b - 10 Mbps (either controlled by in-band RGMII PHY status or by forced speed settings) 01b - 100 Mbps (either controlled by in-band RGMII PHY status or by forced speed settings) 10b - 1 Gbps (either controlled by in-band RGMII PHY status or by forced speed settings) 11b - Reserved
3 RGFD	RGMII full duplex link mode status 0b - No link is established or RGMII PHY does not support the optional in-band signaling 1b - RGMII full duplex link is established. This bit is valid if PHY supports the optional in-band signaling.
4-31 —	Reserved

11.2.25 PFE PCS status register 1 (PFEPCSSR1)

11.2.25.1 Offset

Register	Offset
PFEPCSSR1	440h

11.2.25.2 Function

This register contains bits to log the PCS layer events for MAC1 of PFE in the SGMII mode.

11.2.25.3 Diagram

Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PCS	AN	LT	Reserved												
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

11.2.25.4 Fields

Field	Function
0 PCS	Link synchronization event
1 AN	Auto negotiation status
2 LT	New page received by auto negotiation function
3-31 —	Reserved

11.2.26 PFE interrupt enable control register (PFEINTENCR1)

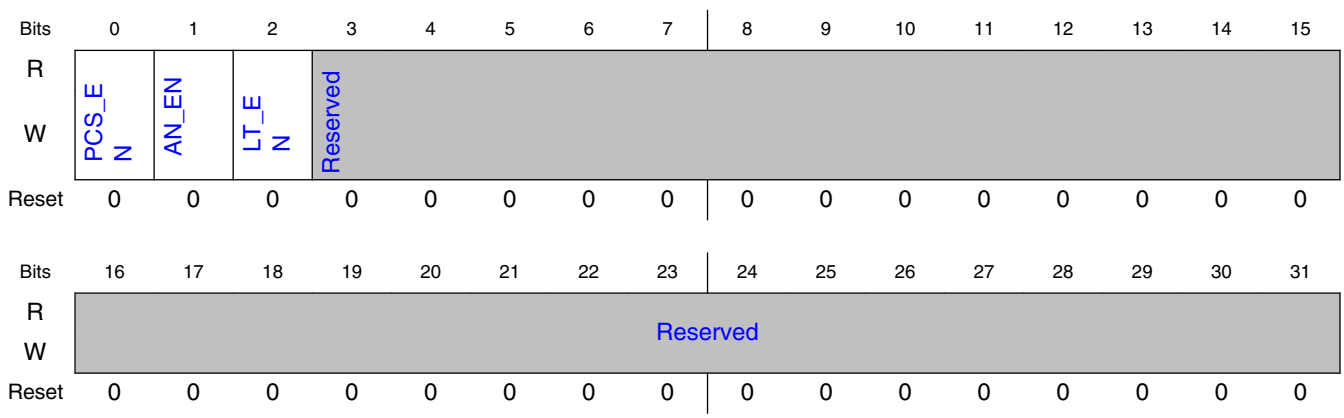
11.2.26.1 Offset

Register	Offset
PFEINTENCR1	444h

11.2.26.2 Function

This register contains control bits to generate an interrupt (IRQ[154]) from the PCS layer events of PFE MAC1.

11.2.26.3 Diagram



11.2.26.4 Fields

Field	Function
0 PCS_EN	Interrupt enable bit for link synchronization event on PFE MAC1.
1 AN_EN	Interrupt enable bit for auto negotiation on PFE MAC1.
2	Interrupt enable bit for new page received on auto negotiation on PFE MAC1.

Table continues on the next page...

Field	Function
LT_EN	
3-31 —	Reserved

11.2.27 PFE PCS status register 2 (PFEPCSSR2)

11.2.27.1 Offset

Register	Offset
PFEPCSSR2	448h

11.2.27.2 Function

This register contains bits to log the PCS layer events for MAC2 of PFE in the SGMII mode.

11.2.27.3 Diagram

Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	PCS	AN	LT	Reserved												
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

11.2.27.4 Fields

Field	Function
0	Link synchronization event

Table continues on the next page...

Field	Function
PCS	
1 AN	Auto negotiation status
2 LT	New page received by auto negotiation function
3-31 —	Reserved

11.2.28 PFE interrupt enable control register 2 (PFEINTENCR2)

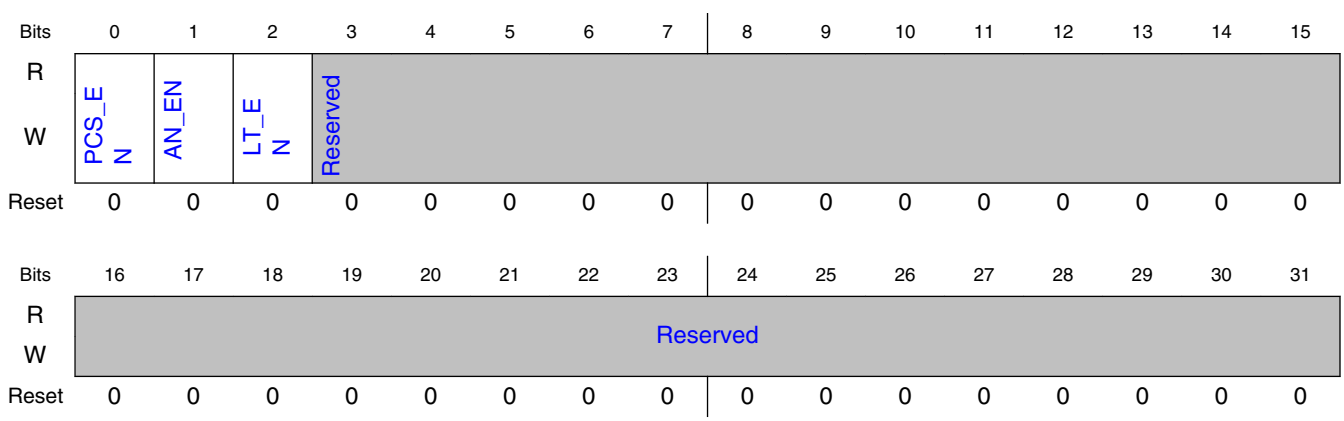
11.2.28.1 Offset

Register	Offset
PFEINTENCR2	44Ch

11.2.28.2 Function

This register contains control bits to generate an interrupt (IRQ[154]) from PCS layer events of PFE MAC2.

11.2.28.3 Diagram



11.2.28.4 Fields

Field	Function
0 PCS_EN	Interrupt enable bit for link synchronization event on PFE MAC2.
1 AN_EN	Interrupt enable bit for auto negotiation on PFE MAC2.
2 LT_EN	Interrupt enable bit for new page received on auto negotiation on PFE MAC2.
3-31 —	Reserved

11.2.29 PFE error control register (PFEERRCR)

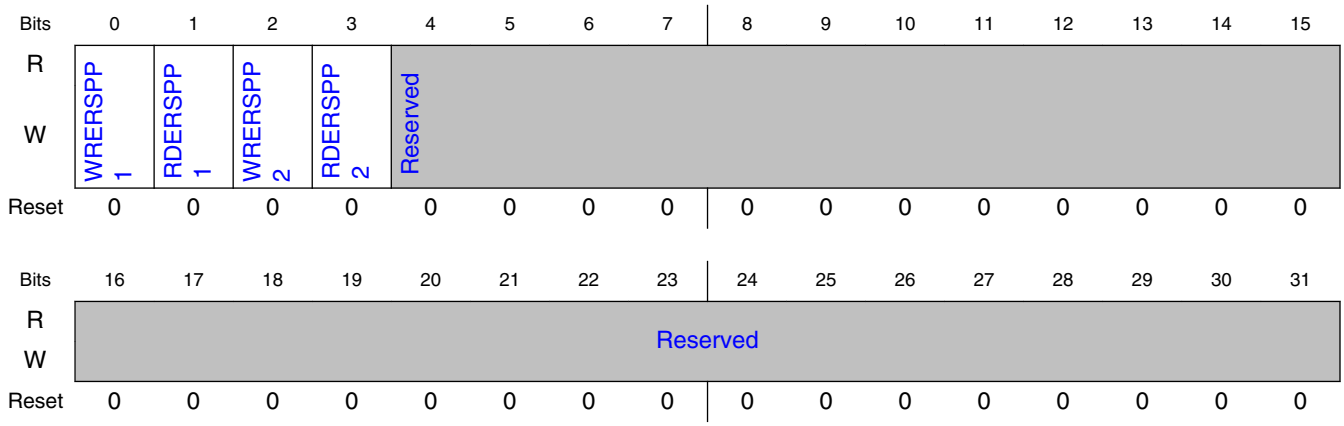
11.2.29.1 Offset

Register	Offset
PFEERRCR	450h

11.2.29.2 Function

This register contains control bits for the error response received by PFE for read and write requests if enabled by PFEINTENCR.

11.2.29.3 Diagram



11.2.29.4 Fields

Field	Function
0 WRERSPP1	Write error response is captured for PFE DDR master interface. Write 1 to clear.
1 RDERSPP1	Read error response is captured for PFE DDR master interface. Write 1 to clear.
2 WRERSPP2	Write error response is captured for PFE HDBUS msster interface. Write 1 to clear.
3 RDERSPP2	Read error response is captured for PFE HDBUS msster Interface. Write 1 to clear.
4-31 —	Reserved

11.2.30 PFE error interrupt enable control register (PFEERRINTENCR)

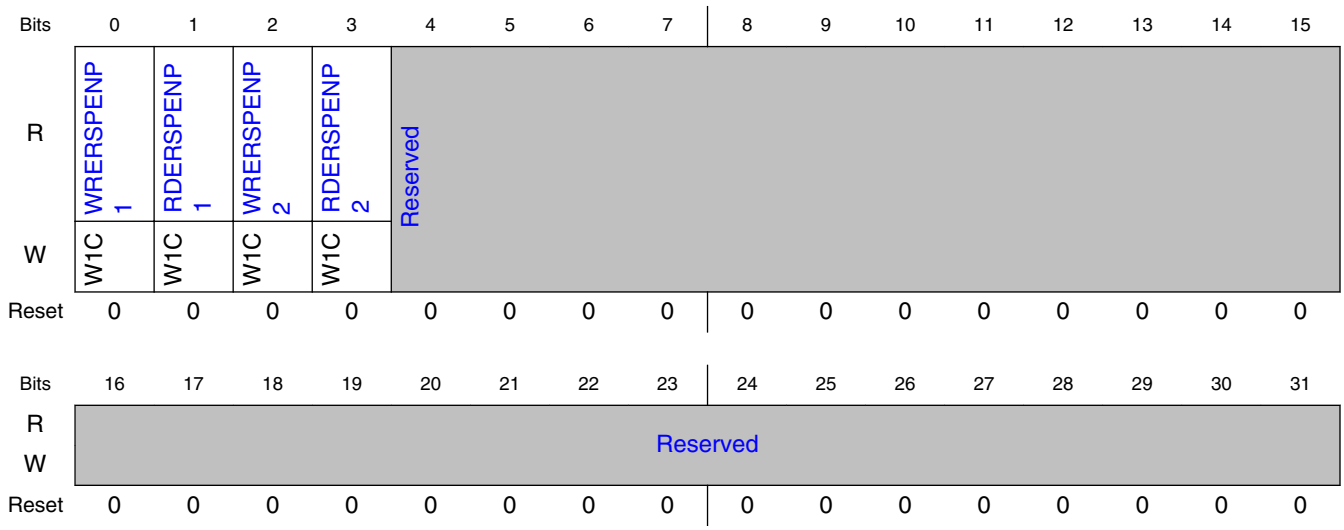
11.2.30.1 Offset

Register	Offset
PFEERRINTENCR	454h

11.2.30.2 Function

This register contains control bits to generate an interrupt for the error response received by PFE for read and write requests.

11.2.30.3 Diagram



11.2.30.4 Fields

Field	Function
0 WRERSPENP1	Interrupt enable bit for write error response captured for PFE DDR master interface.
1 RDERSPENP1	Interrupt enable bit for read error response captured for PFE DDR master interface.
2 WRERSPENP2	Interrupt enable bit for write error response captured for PFE HDBUS master interface.
3 RDERSPENP2	Interrupt enable bit for read error response captured for PFE HDBUS master Interface.
4-31 —	Reserved

11.2.31 PFEA sideband control register (PFEASBCR)

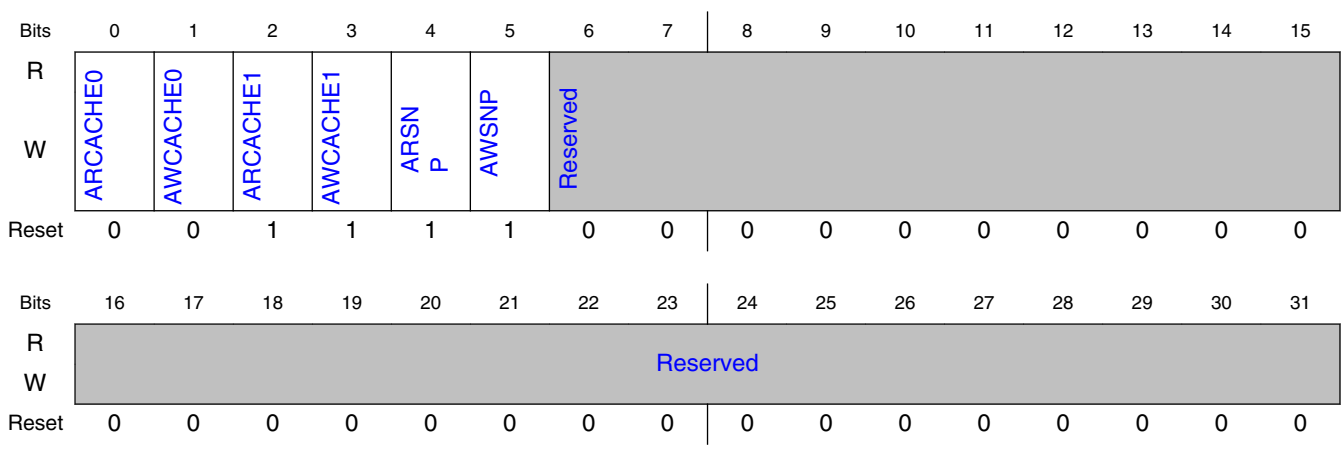
11.2.31.1 Offset

Register	Offset
PFEASBCR	458h

11.2.31.2 Function

This register contains control bits to provide programmability for AXI attributes of PFE DDR master interface.

11.2.31.3 Diagram



11.2.31.4 Fields

Field	Function
0 ARCACHE0	Control bit for ARCACHE[0] attribute. 0b - Non bufferable (default) 1b - Bufferable
1 AWCACHE0	Control bit for AWCACHE[0] attribute. 0b - Non bufferable (default) 1b - Bufferable
2	Control bit for ARCACHE[1] attribute.

Table continues on the next page...

Field	Function
ARCACHE1	0b - Non modifiable (default) 1b - Modifiable
3 AWCACHE1	Control bit for AWCACHE[1] attribute. 0b - Non modifiable (default) 1b - Modifiable
4 ARSNP	Control bit for snoopable attribute of read channel 0b - Non snoopable 1b - Snoopable (default)
5 AWSNP	Control bit for snoopable attribute of write channel 0b - Non snoopable 1b - Snoopable (default)
6-31 —	Reserved

11.2.32 PFEB sideband control register (PFEBSBCR)

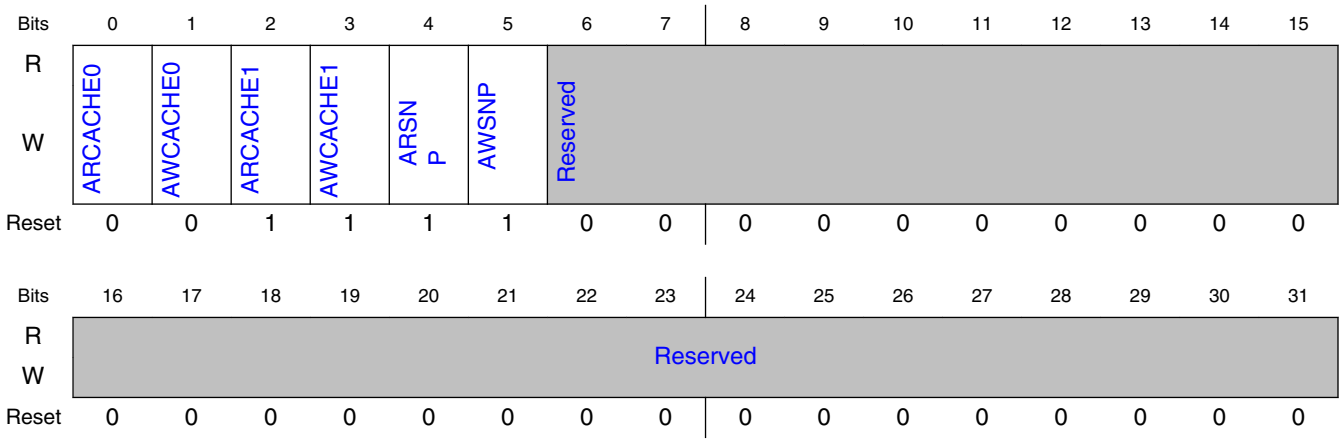
11.2.32.1 Offset

Register	Offset
PFEBSBCR	45Ch

11.2.32.2 Function

This register contains control bits to provide programmability for AXI attributes of PFE HDBUS master interface.

11.2.32.3 Diagram



11.2.32.4 Fields

Field	Function
0 ARCACHE0	Control bit for ARCACHE[0] attribute. 0b - Non bufferable (default) 1b - Bufferable
1 AWCACHE0	Control bit for AWCACHE[0] attribute. 0b - Non bufferable (default) 1b - Bufferable
2 ARCACHE1	Control bit for ARCACHE[1] attribute. 0b - Non modifiable (default) 1b - Modifiable
3 AWCACHE1	Control bit for AWCACHE[1] attribute. 0b - Non modifiable (default) 1b - Modifiable
4 ARSNP	Control bit for snoopable attribute of read channel 0b - Non snoopable 1b - Snoopable (default)
5 AWSNP	Control bit for snoopable attribute of write channel 0b - Non snoopable 1b - Snoopable (default)
6-31 —	Reserved

11.2.33 MDIO select control register (MDIOSELCR)

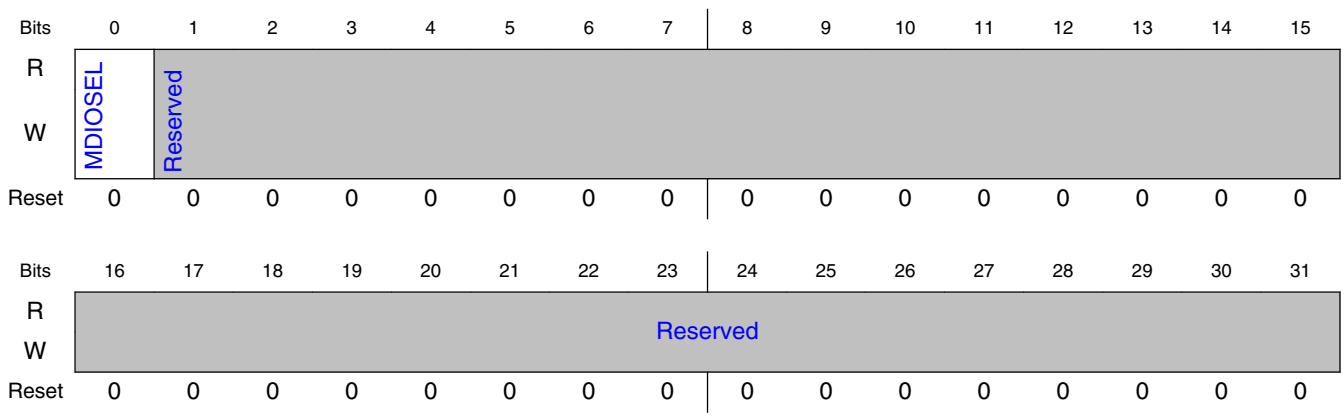
11.2.33.1 Offset

Register	Offset
MDIOSELCR	484h

11.2.33.2 Function

This register contains a control bit to select the source of EMI1_MDIO directed to PFE.

11.2.33.3 Diagram



11.2.33.4 Fields

Field	Function
0 MDIOSEL	EMI1_MDIO source select 0b - Internally generated MDIO from SerDes directed to PFE (default) 1b - MDIO from external Ethernet PHY (EMI1_MDIO) directed to PFE
1-31 —	Reserved

11.2.34 Spare control register (SPARECR1 - SPARECR8)

11.2.34.1 Offset

For a = 1 to 8:

Register	Offset
SPARECRa	4FCh + (a × 4h)

11.2.34.2 Function

This register provides expansion bits for device control.

11.2.34.3 Diagram

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

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

11.2.34.4 Fields

Field	Function
0-31 VAL	32-bit spare contents

11.2.35 I2C debug mode control register (I2CDBGCR)

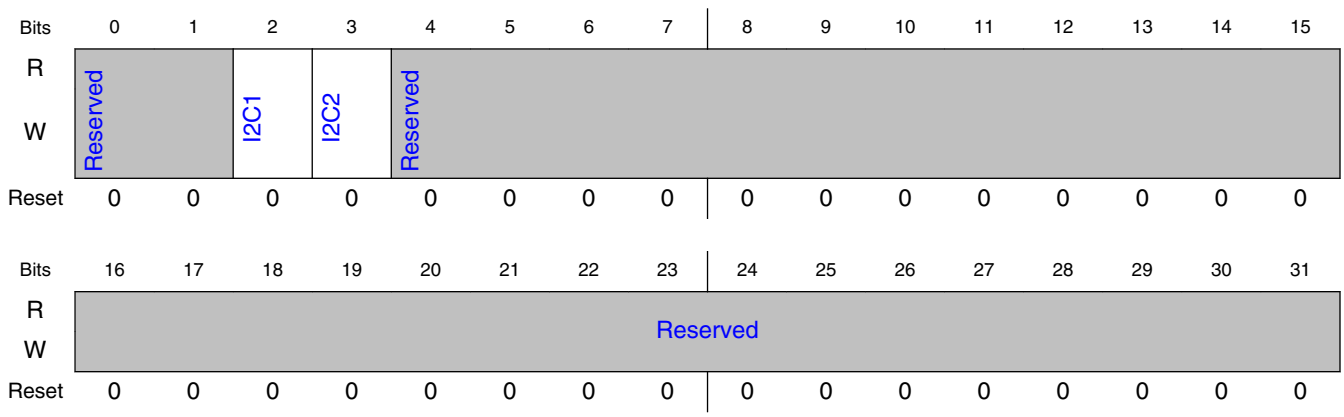
11.2.35.1 Offset

Register	Offset
I2CDBGCR	520h

11.2.35.2 Function

This register provides glitch filter control for I2C1 and I2C2.

11.2.35.3 Diagram



11.2.35.4 Fields

Field	Function
0-1 —	Reserved
2 I2C1	I2C1 glitch filter enable 0b - I2C1 glitch filter disabled 1b - I2C1 glitch filter enabled
3 I2C2	I2C2 glitch filter enable 0b - I2C2 glitch filter disabled 1b - I2C2 glitch filter enabled
4-31 —	Reserved

11.2.36 Scratch read write register (SCRATCHRW1 - SCRATCHRW4)

11.2.36.1 Offset

For a = 1 to 4:

Register	Offset
SCRATCHRWa	5FCh + (a × 4h)

11.2.36.2 Function

The SCRATCHn read write register n (n=1 to 4), provides read / write scratch register locations available to the user. It provides expansion bits for device control.

Note that the definition of SCRATCHRW1 and SCRATCHRW2 is applicable while performing non-secure boot whereas SCRATCHRW3 and SCRACTHRW4 are defined for secure-boot.

While performing secure boot/non-secure boot, these registers are defined as follows:

SCRATCHRW1 - Reserved for future use in case boot location pointer is 64 bit. The value must be 0.

SCRATCHRW2 - 32-bit boot location pointer (BOOTLOCPTR)

SCRATCHRW3 - Register that indicates failures in SEC self tests (if any) conducted as part of secure boot flow

SCRATCHRW4- Reserved

11.2.36.3 Diagram

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

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

11.2.36.4 Fields

Field	Function
0-31 VAL	32-bit scratch contents.

11.2.37 Core boot control register (COREBCR)

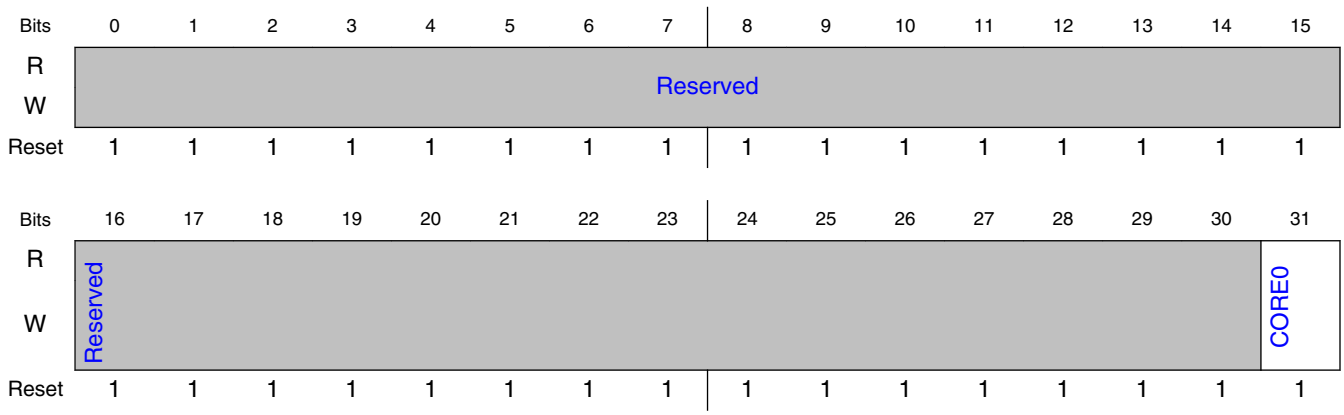
11.2.37.1 Offset

Register	Offset
COREBCR	680h

11.2.37.2 Function

This register provides expansion bits for device control. These bits are set on assertion of core reset. This register is blocked in PBI.

11.2.37.3 Diagram



11.2.37.4 Fields

Field	Function
0-30 —	Reserved
31 CORE0	Write 1 to clear bit for core 0 . Also set with core 0 reset.

11.2.38 Shared message signaled interrupt index register (PEX1 MSIIR)

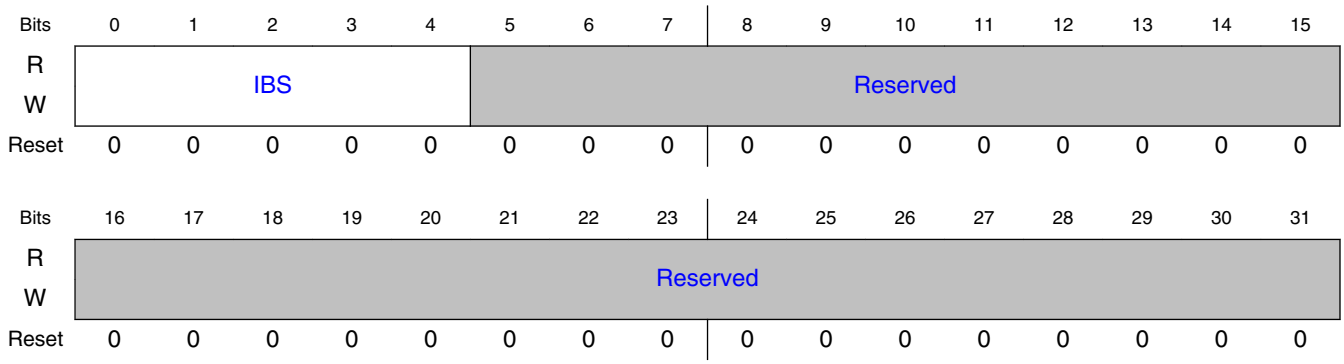
11.2.38.1 Offset

Register	Offset
PEX1MSIIR	2000h

11.2.38.2 Function

This register provides the mechanism for setting an interrupt in the MSIR. When MSIIR is written, MSIIR[IBS] selects the shared interrupt bit in the selected MSIR register. MSIIR is primarily intended to support PCI Express MSIs.

11.2.38.3 Diagram



11.2.38.4 Fields

Field	Function
0-4 IBS	Interrupt bit select. This bit selects the bit to set in MSIR. 00000b - Set field SH0 (bit 31) 00001b - Set field SH1 (bit 30) 00010b - Set field SH2 (bit 29) 11111b - Set field SH31 (bit 0)
5-31 —	Reserved

11.2.39 Shared message signaled interrupt register (PEX1MSIR)

11.2.39.1 Offset

Register	Offset
PEX1MSIR	2004h

11.2.39.2 Function

This register indicates which of the up to 32 interrupt sources sharing a message register have pending interrupts. This register is cleared when read and write to this register has no effect.

11.2.39.3 Diagram

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

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

11.2.39.4 Fields

Field	Function
0-31 SHn	Message sharer n has a pending interrupt. **NOTE:** Bit 0 corresponds to SH0, bit 1 corresponds to SH1, and so on.

Chapter 12
Device Configuration and Pin Control

12.1 Device Configuration and Pin Control Introduction

This chapter describes the device configuration and pin control facilities of the chip.

The device configuration unit provides general purpose configuration and status for the device and the pin control block implements general purpose configuration and status registers for the device, and the logic to configure the I/O pads to operate according to the requirements of the functional components. It also controls the data path between the SoC components and the I/O pads. The pin control block consists of the pins control module, the functional I/O multiplexing, and the JTAG boundary scan logic.

12.2 Features

The device configuration unit features the following:

- Pin sampling of device configuration pins at power-on reset and a corresponding POR status register for capturing the values of these configuration pins
- Reset Configuration Word (RCW) support via a set of RCW status registers written by the Preboot Loader (PBL) during power-on reset in the PBL's RCW stage
- Boot release registers(s) used for releasing cores for booting
- Register file for the Reset module including:
  - Register for initiating a device RESET_REQ_B through software
  - Set of registers for control and status of sources on the device which can drive the device's RESET_REQ_B pin
- Core and device disable registers used for gating off clocks for any IP blocks or cores which are not used at all by an application
- Two small sets of scratch registers:
  - One set of read / write scratch registers
  - One set of write-once / read scratch registers

12.3 DCFG_CCSR register descriptions

The table below shows the memory-mapped CCSR registers of the device configuration module and lists the offset, name, and a cross-reference to the complete description of each register. These registers only support 32-bit accesses.

12.3.1 DCFG_CCSR Memory map

DCFG_CCSR base address: 1EE_0000h

Offset	Register	Width (In bits)	Access	Reset value
0h	POR status register 1 (PORSR1)	32	RO	See description.
4h	POR Status Register 2 (PORSR2)	32	RO	See description.
28h	Fuse Status Register (FUSESR)	32	RO	0000_0000h
70h	Device Disable Register 1 (DEVDISR1)	32	RW	0000_0000h
74h	Device Disable Register 2 (DEVDISR2)	32	RU	0000_0000h
78h	Device Disable Register 3 (DEVDISR3)	32	RU	0000_0000h
7Ch	Device Disable Register 4 (DEVDISR4)	32	RW	0000_0000h
80h	Device Disable Register 5 (DEVDISR5)	32	RW	0000_0000h
A4h	System Version Register (SVR)	32	RO	8704_0120h
B0h	Reset Control Register (RSTCR)	32	RW	0000_0000h
B4h	Reset Request Preboot Loader Status Register (RSTRQPBLSR)	32	W1C	0000_0000h
C0h	Reset Request Mask Register (RSTRQMR1)	32	RW	0000_4000h
C8h	Reset Request Status Register (RSTRQSR1)	32	RO	0000_0000h
E4h	Boot Release Register (BRR)	32	RW	See description.
100h - 13Ch	Reset Control Word Status Register n (RCWSR0 - RCWSR15)	32	RO	See description.
200h - 20Ch	Scratch Read / Write Register n (SCRATCHRW1 - SCRATCHRW4)	32	RW	0000_0000h
300h - 30Ch	Scratch Read Register n (SCRATCHW1R1 - SCRATCHW1R4)	32	RW	0000_0000h
400h	Core Reset Status Register n (CRSTSR0)	32	W1C	0000_0000h
608h	DMA Control Register (DMACR1)	32	RW	0000_0000h
740h - 83Ch	Topology Initiator Type n Register (TP_ITYP0 - TP_ITYP63)	32	RO	See description.
844h	Core Cluster n Topology Register (TP_CLUSTER1)	32	RO	See description.

12.3.2 POR status register 1 (PORSR1)

12.3.2.1 Offset

Register	Offset
PORSR1	0h

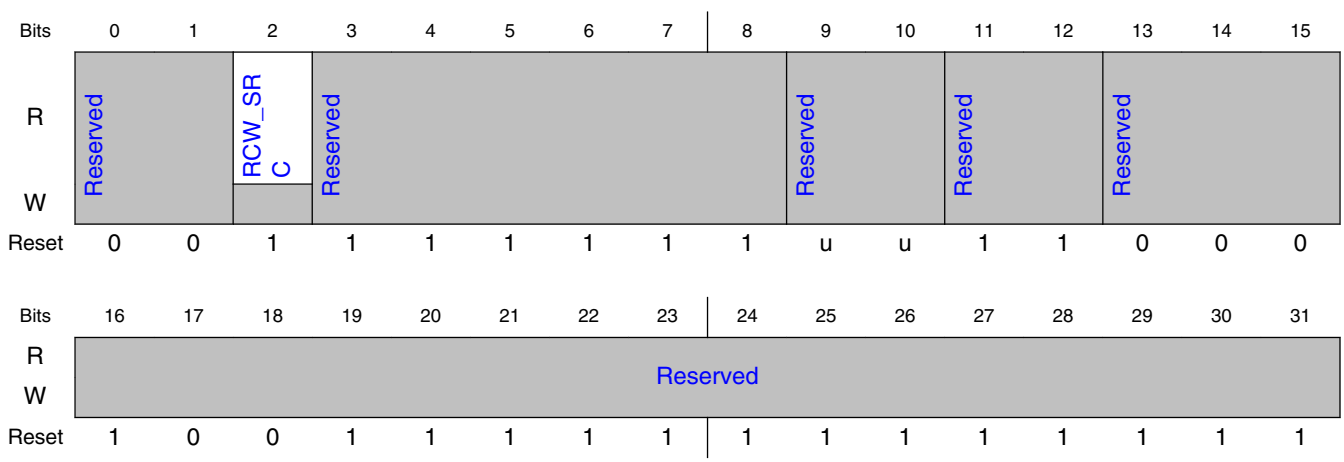
12.3.2.2 Function

PORSR1 captures the values of the device's POR configuration pins.

NOTE

The status of these bits depends on the sampling value driven on the pad.

12.3.2.3 Diagram



12.3.2.4 Fields

Field	Function
0-1 —	Reserved

Table continues on the next page...

Field	Function
2 RCW_SRC	Reset Configuration Word Source. Indicates which QSPI memory contains the RCW information. Loaded with the value of cfg_rcw_src. 0b - Hard-coded RCW source 1b - QSPI is the RCW source
3-8 —	Reserved
9-10 —	Reserved
11-12 —	Reserved
13-31 —	Reserved

12.3.3 POR Status Register 2 (PORSR2)

12.3.3.1 Offset

Register	Offset
PORSR2	4h

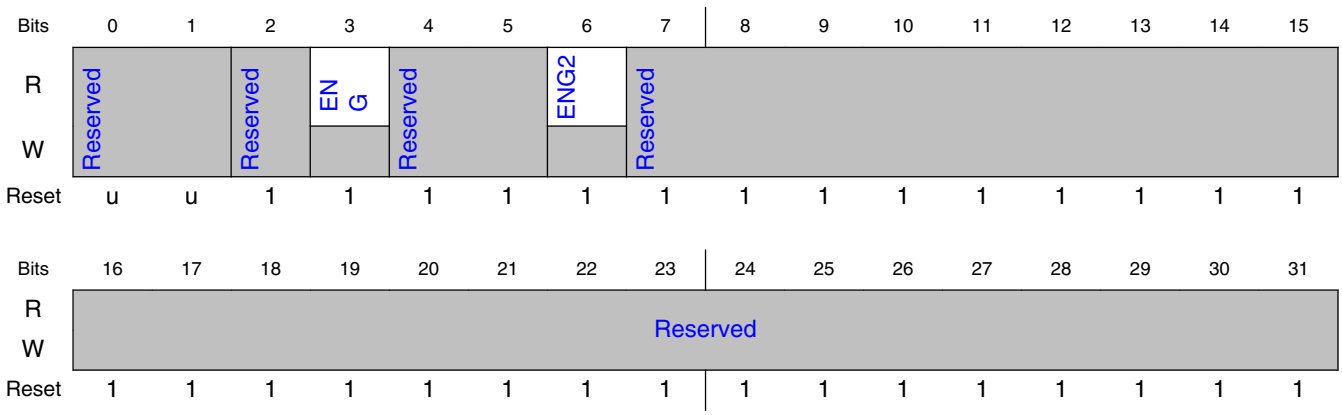
12.3.3.2 Function

PORSR2 captures the values of the device's POR configuration pins.

NOTE

The status of these bits depends on the sampling value driven on the pad.

12.3.3.3 Diagram



12.3.3.4 Fields

Field	Function
0-1 —	Reserved
2 —	Reserved
3 ENG	Provides transconductance value for the 25 MHz crystal and is applied based on the type of crystal used. Loaded with the value of the cfg_eng_use pin at power-on reset. Refer Transconductance selection control for details. 0b - Config pin was not set 1b - Config pin was set
4-5 —	Reserved Reserved
6 ENG2	Together with ENG, this bit provides the 2-bit transconductance value for the 25 MHz crystal and is applied based on the type of crystal used. Loaded with the value of the cfg_eng_use[2] pin at power-on reset. Refer Transconductance selection control for details. 0b - Config pin was not set 1b - Config pin was set
7-31 —	Reserved Reserved

12.3.4 Fuse Status Register (FUSESR)

12.3.4.1 Offset

Register	Offset
FUSESR	28h

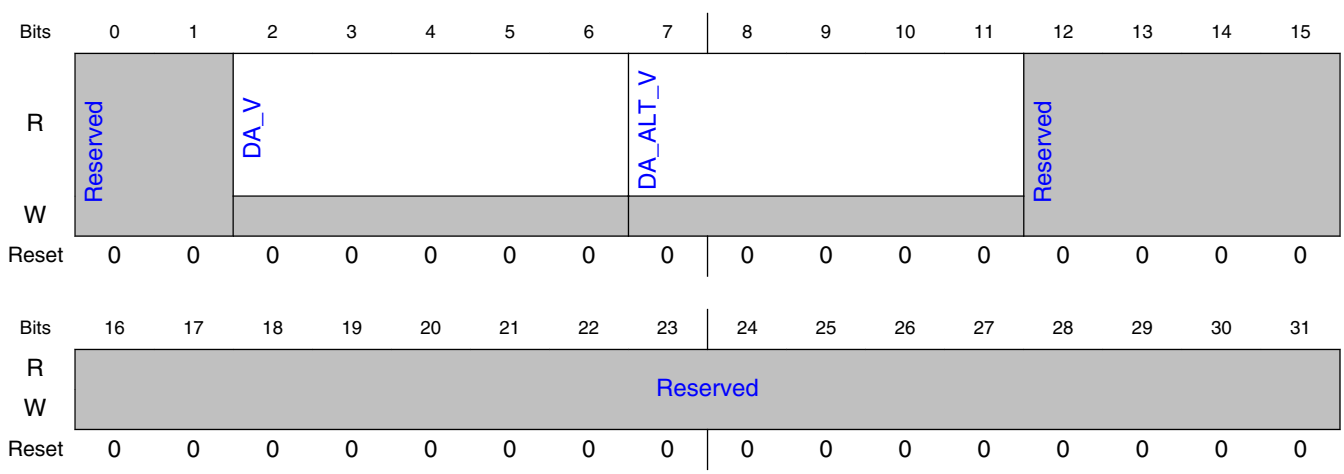
12.3.4.2 Function

The fuse status register provides the values from on-chip voltage ID fuses programmed at the factory. These values define the voltage requirements for the chip. Boot software must read FUSESR and translate the values into the appropriate commands to set the voltage output value of an external voltage regulator.

NOTE

VID is not applicable for this chip, which means the boot software is not required to read this register and adjust voltage.

12.3.4.3 Diagram



12.3.4.4 Fields

Field	Function
0-1 —	Reserved

Table continues on the next page...

Field	Function
2-6 DA_V	VDD voltage. This is the minimum voltage required to reach the frequency specified. Note that most of the possible voltages levels will never be specified by the factory. 00000b - unused (default) 00001b - 0.9875 V 00010b - 0.9750 V 00011b - 0.9625 V 00100b - 0.9500 V 00101b - 0.9375 V 00110b - 0.9250 V 00111b - 0.9125 V 01000b - 0.9000 V 01001b - 0.8875 V 01010b - 0.8750 V 01011b - 0.8625 V 01100b - 0.8500 V 01101b - 0.8375 V 01110b - 0.8250 V 01111b - 0.8125 V 10000b - 1.0000 V 10001b - 1.0125 V 10010b - 1.0250 V 10011b - 1.0375 V 10100b - 1.0500 V 10101b - 1.0625 V 10110b - 1.0750 V 10111b - 1.0875 V 11000b - 1.1000 V 11001b - Reserved 11010b - Reserved 11011b - Reserved 11100b - Reserved 11101b - Reserved 11110b - Reserved 11111b - Use value in DA_ALT_V field.
7-11 DA_ALT_V	VDD alternate voltage. This is a secondary voltage field for VDD. The field encodings are the same as DA_V , except 11111 is reserved..
12-31 —	Reserved

12.3.5 Device Disable Register 1 (DEVDISR1)

12.3.5.1 Offset

Register	Offset
DEVDISR1	70h

12.3.5.2 Function

A given application may not use all the peripherals on the device. In this case, it may be desirable to disable unused peripherals. DEVDISR1 provides a mechanism for gating clocks to IP blocks that are not used when running an application.

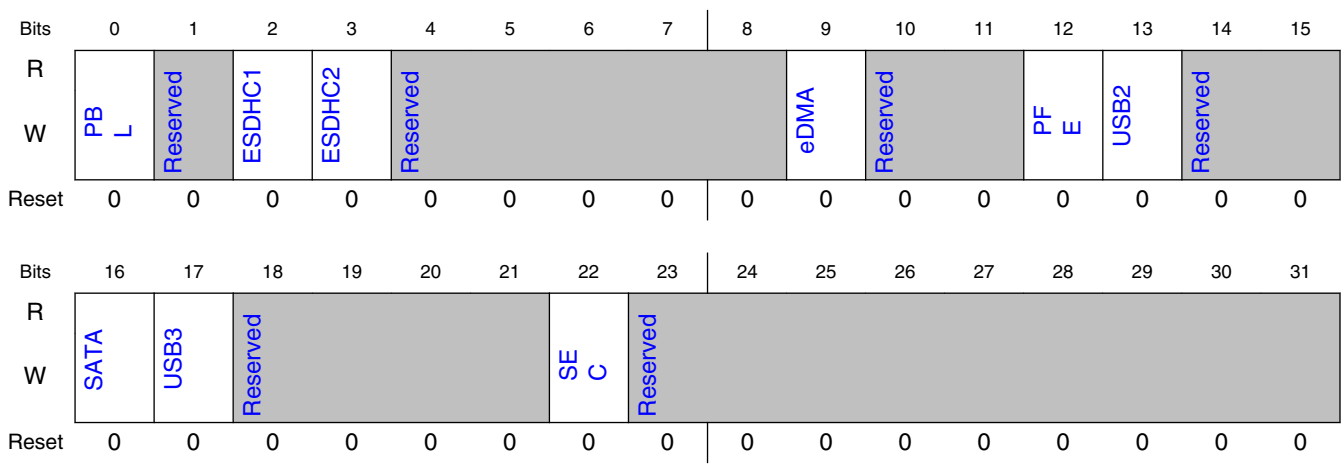
NOTE

IP blocks disabled by setting the corresponding bit in the DEVDISR1 register must not be re-enabled.

NOTE

Power down for any module can be configured by programming the DCFG_DEVDISR register. If any module is powered down, then its data access as well as register access are not allowed.

12.3.5.3 Diagram



12.3.5.4 Fields

Field	Function
0 PBL	Pre-boot loader disable. 0b - Module is enabled 1b - Module is disabled
1 —	Reserved

Table continues on the next page...

Field	Function
2 ESDHC1	eSDHC controller 1 disable. 0b - Module is enabled 1b - Module is disabled
3 ESDHC2	eSDHC controller 2 disable. 0b - Module is enabled 1b - Module is disabled
4-8 —	Reserved
9 eDMA	eDMA controller disable. 0b - Module is enabled 1b - Module is disabled
10-11 —	Reserved
12 PFE	PFE module disable. 0b - Module is enabled 1b - Module is disabled
13 USB2	USB (2.0) controller 2 disable. 0b - Module is enabled 1b - Module is disabled
14-15 —	Reserved
16 SATA	SATA disable. 0b - Module is enabled 1b - Module is disabled
17 USB3	USB (3.0) controller 1 disable. 0b - Module is enabled 1b - Module is disabled
18-21 —	Reserved
22 SEC	SEC module disable. 0b - Module is enabled 1b - Module is disabled
23-31 —	Reserved

12.3.6 Device Disable Register 2 (DEVDISR2)

12.3.6.1 Offset

Register	Offset
DEVDISR2	74h

12.3.6.2 Function

A given application may not use all the peripherals on the device. In this case, it may be desirable to disable unused peripherals. DEVDISR2 provides a mechanism for gating clocks to IP blocks that are not used when running an application.

12.3.6.3 Diagram

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

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

12.3.6.4 Fields

Field	Function
0-31 —	Reserved

12.3.7 Device Disable Register 3 (DEVDISR3)

12.3.7.1 Offset

Register	Offset
DEVDISR3	78h

12.3.7.2 Function

A given application may not use all the peripherals on the device. In this case, it may be desirable to disable unused peripherals. DEVDISR3 provides a mechanism for gating clocks to IP blocks that are not used when running an application.

12.3.7.3 Diagram

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

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

12.3.7.4 Fields

Field	Function
0-31 —	Reserved

12.3.8 Device Disable Register 4 (DEVDISR4)

12.3.8.1 Offset

Register	Offset
DEVDISR4	7Ch

12.3.8.2 Function

A given application may not use all the peripherals on the device. In this case, it may be desirable to disable unused peripherals. DEVDISR4 provides a mechanism for gating clocks to IP blocks that are not used when running an application.

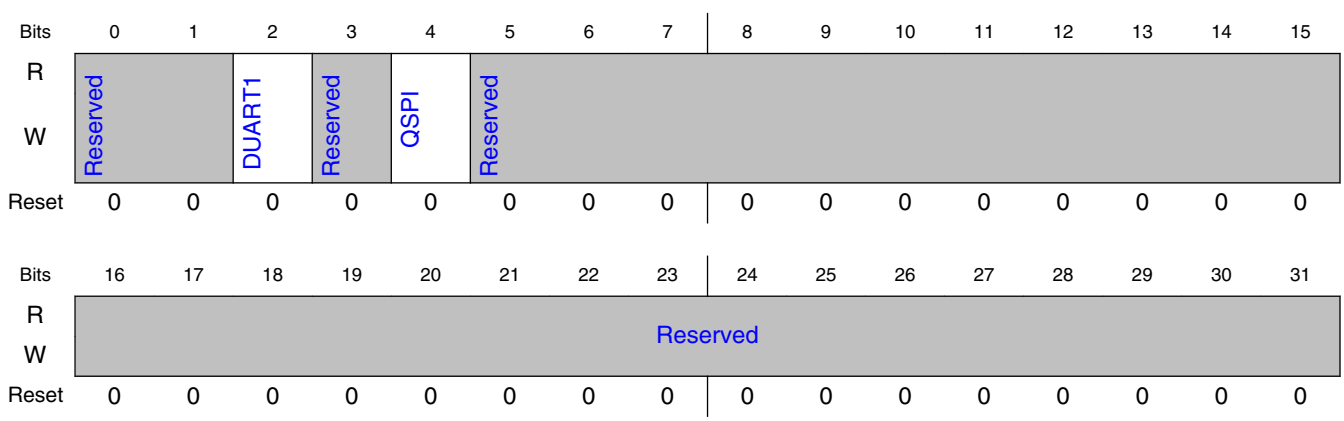
NOTE

IP Blocks disabled by setting the corresponding bit in the DEVDISR4 register must not be re-enabled.

NOTE

Power down for any module can be configured by programming the DCFG_DEVDISR register. If any module is powered down, then its data access as well as register access are not allowed.

12.3.8.3 Diagram



12.3.8.4 Fields

Field	Function
0-1 —	Reserved
2 DUART1	DUART1 module disable. 0b - Module is enabled 1b - Module is disabled
3 —	Reserved
4 QSPI	QuadSPI module disable 0b - Module is enabled 1b - Module is disabled
5-31 —	Reserved

12.3.9 Device Disable Register 5 (DEVDISR5)

12.3.9.1 Offset

Register	Offset
DEVDISR5	80h

12.3.9.2 Function

A given application may not use all the peripherals on the device. In this case, it may be desirable to disable unused peripherals. DEVDISR5 provides a mechanism for gating clocks to IP blocks that are not used when running an application.

NOTE

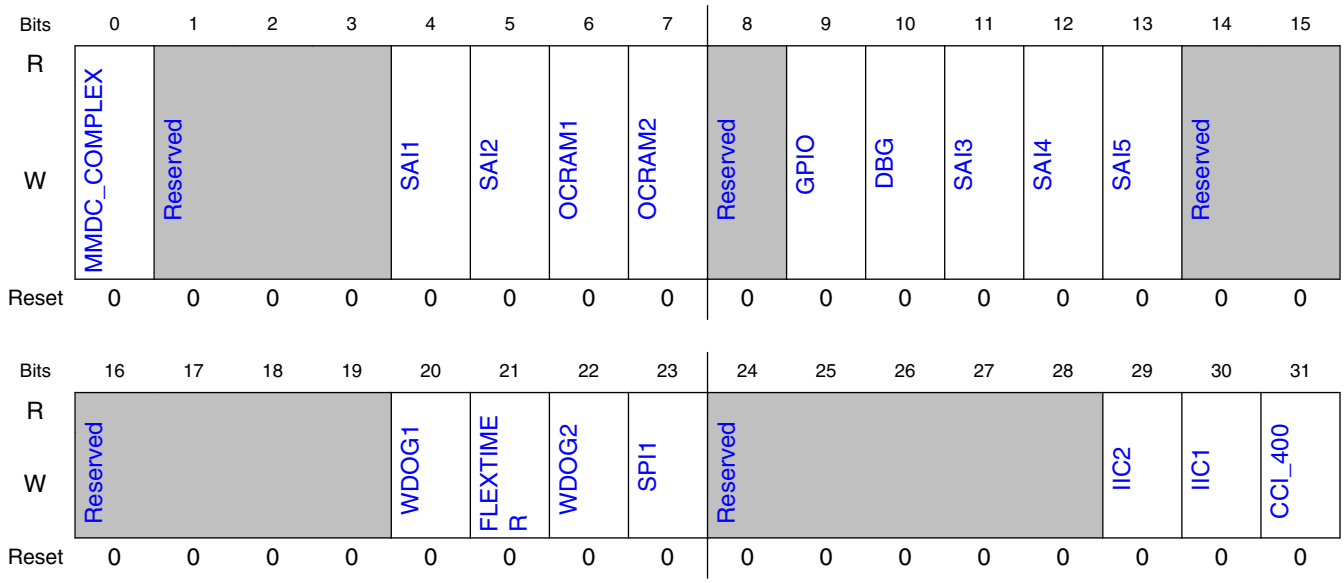
IP Blocks disabled by setting the corresponding bit in the DEVDISR5 register must not be re-enabled.

NOTE

Power down for any module can be configured by programming the DCFG_DEVDISR register. If any module is

powered down, then its data access as well as register access are not allowed.

12.3.9.3 Diagram



12.3.9.4 Fields

Field	Function
0 MMDC_COMPLEX	DDR controller disable. 0b - Module is enabled 1b - Module is disabled
1-3 —	Reserved
4 SAI1	SAI1 disable. 0b - Module is enabled 1b - Module is disabled
5 SAI2	SAI2 disable. 0b - Module is enabled 1b - Module is disabled
6 OCRAM1	OCRAM1 disable 0b - Module is enabled 1b - Module is disabled
7 OCRAM2	OCRAM2 disable 0b - Module is enabled 1b - Module is disabled

Table continues on the next page...

Field	Function
8 —	Reserved
9 GPIO	GPIO disable. **NOTE:** This field disables all GPIO modules. 0b - Module is enabled 1b - Module is disabled
10 DBG	Debug module disable. 0b - Module is enabled 1b - Module is disabled
11 SAI3	SAI3 disable. 0b - Module is enabled 1b - Module is disabled
12 SAI4	SAI4 disable. 0b - Module is enabled 1b - Module is disabled
13 SAI5	SAI5 disable. 0b - Module is enabled 1b - Module is disabled
14-19 —	Reserved
20 WDOG1	WatchDog1 disable. 0b - Module is enabled 1b - Module is disabled
21 FLEXTIMER	FlexTimer disable. 0b - Module is enabled 1b - Module is disabled
22 WDOG2	WatchDog 2 disable. 0b - Module is enabled 1b - Module is disabled
23 SPI1	SPI1 disable. 0b - Module is enabled 1b - Module is disabled
24-28 —	Reserved
29 IIC2	I^2C2 disable. 0b - Module is enabled 1b - Module is disabled
30 IIC1	I^2C1 disable. 0b - Module is enabled 1b - Module is disabled
31 CCI_400	Cache coherent interconnect disable. 0b - Module is enabled 1b - Module is disabled

12.3.10 System Version Register (SVR)

12.3.10.1 Offset

Register	Offset
SVR	A4h

12.3.10.2 Function

The SVR contains the system version number for the device. This value can also be read though the SVR SPR of the Arm Cortex- A53 core.

12.3.10.3 Diagram

Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MFR_ID				SOC_DEV_ID											
W																
Reset	1	0	0	0	0	1	1	1	0	0	0	0	0	1	0	0

Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	VAR_PER								MAJOR_REV				MINOR_REV			
W																
Reset	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0

12.3.10.4 Fields

Field	Function
0-3 MFR_ID	Manufacturer ID
4-15 SOC_DEV_ID	Chip Device ID
16-23 VAR_PER	Various Personalities 0000_0001 LS1012A (Export controlled crypto hardware enabled)

Table continues on the next page...

Field	Function
	0000_0000 LS1012A (Export controlled crypto hardware disabled)
24-27 MAJOR_REV	Major Revision Number For silicon 2.0, the major revision is 0x2. For silicon 1.0, the major revision is 0x1.
28-31 MINOR_REV	Minor Revision Number

12.3.11 Reset Control Register (RSTCR)

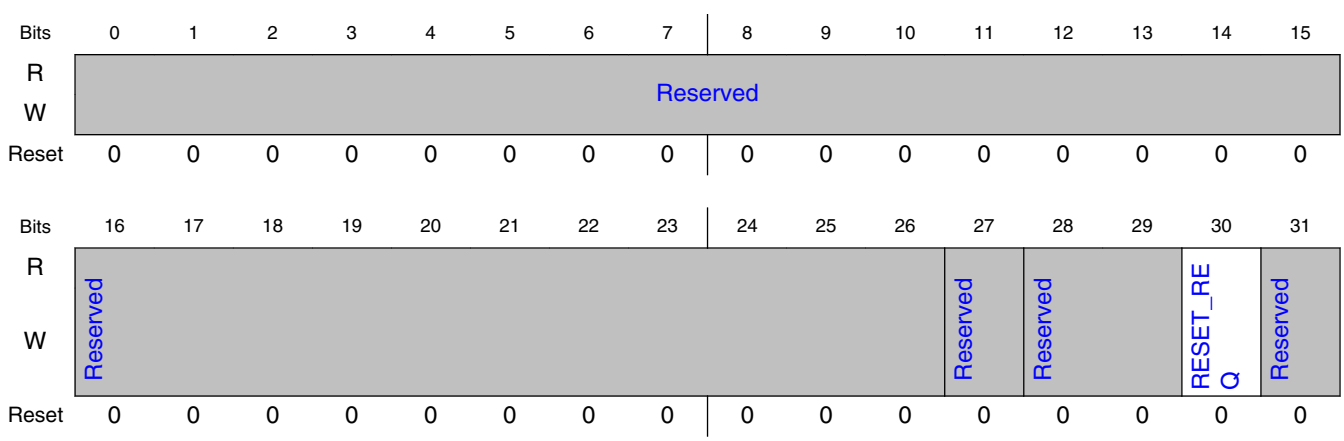
12.3.11.1 Offset

Register	Offset
RSTCR	B0h

12.3.11.2 Function

The RSTCR allows software to control reset functions.

12.3.11.3 Diagram



12.3.11.4 Fields

Field	Function
0-26 —	Reserved
27 —	Reserved
28-29 —	Reserved
30 RESET_REQ	Hardware reset request. External hardware may then decide to issue the desired reset signal (PORESET_B) to the device. It is the primary functionality only for secure device, else this needs to be selected through RCW. 0b - No reset request initiated. 1b - Hardware reset request initiated by software.
31 —	Reserved

12.3.12 Reset Request Preboot Loader Status Register (RSTR QPBLSR)

12.3.12.1 Offset

Register	Offset
RSTRQPBLSR	B4h

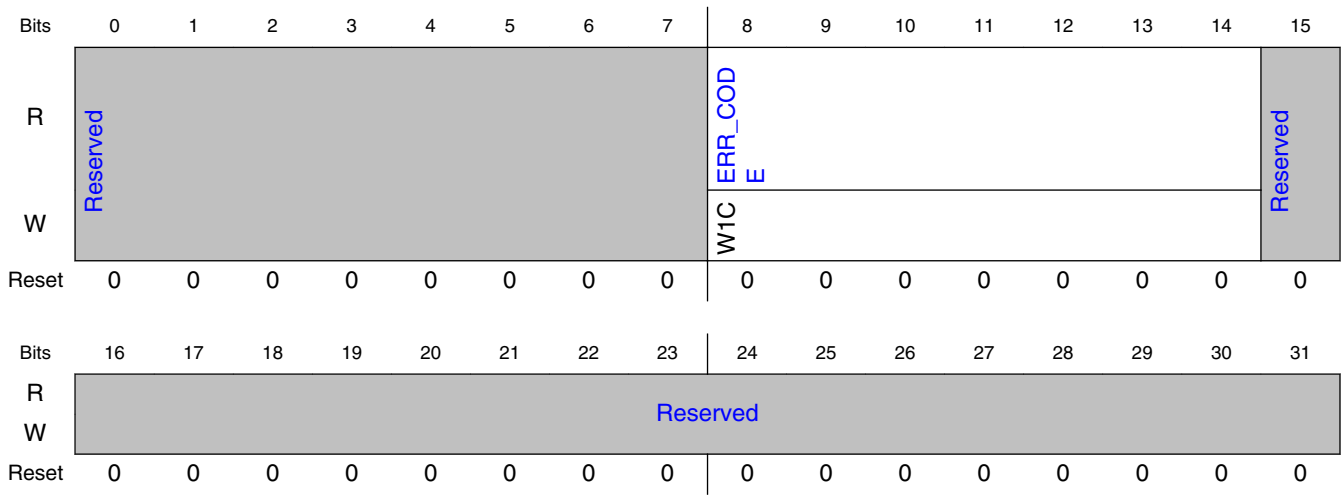
12.3.12.2 Function

The RSTRQPBLSR contains status bits to record the reasons for RESET_REQ_B assertion. It excludes core watchdog timer sources.

NOTE

This register's code is valid only when RSTRQSR[PBL_RR] is set.

12.3.12.3 Diagram



12.3.12.4 Fields

Field	Function
0-7 —	Reserved
8-14 ERR_CODE	7-bit PBL Error Code 7-bit encoded value reflects one of 128 possible PBL errors. Write 1 to each bit to clear this field. **NOTE:** See Error codes for details on the PBL error encodings.
15-31 —	Reserved

12.3.13 Reset Request Mask Register (RSTRQMR1)

12.3.13.1 Offset

Register	Offset
RSTRQMR1	C0h

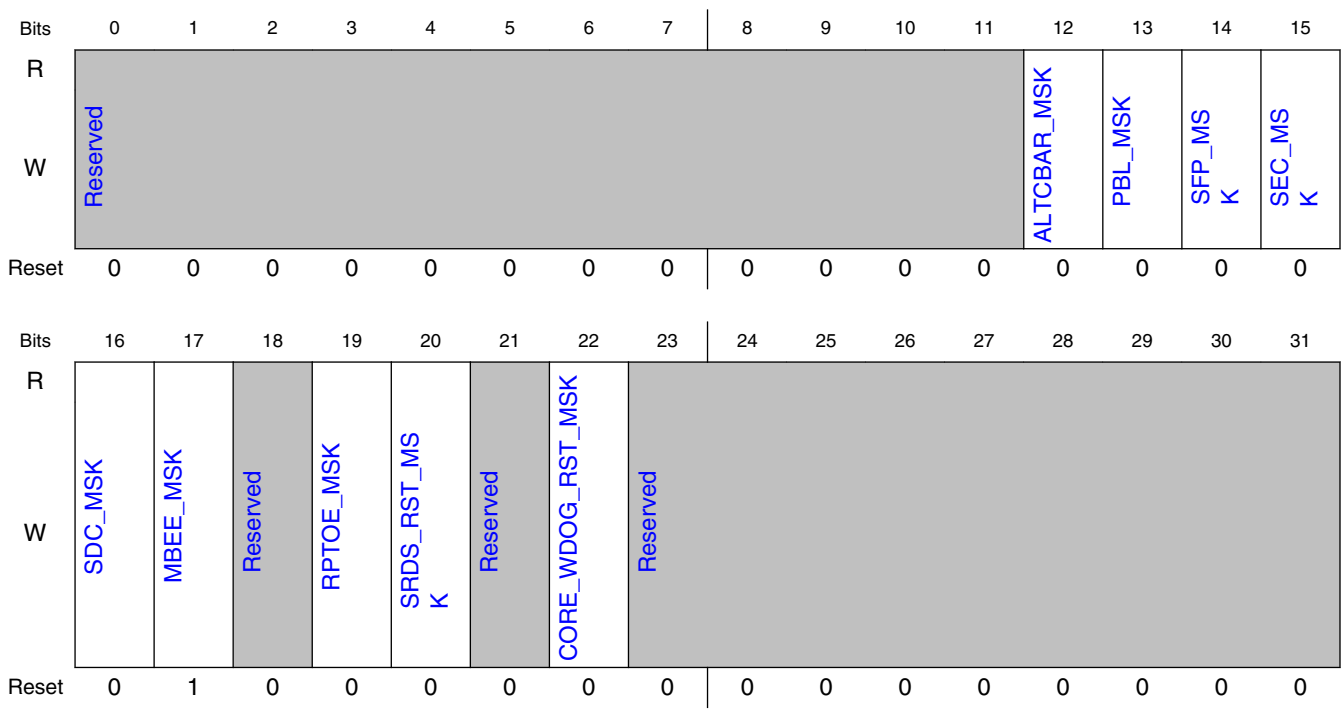
12.3.13.2 Function

The RSTRQMR contains mask bits for optional masking of RESET_REQ_B sources to prevent generation of such a reset request. It excludes core watchdog timer sources.

NOTE

When RESET_REQ_B is selected, the SFP_MSK bit is reserved and PBL_MSK is required only in PBI phase and not in the RCW loading stage.

12.3.13.3 Diagram



12.3.13.4 Fields

Field	Function
0-11 —	Reserved
12 ALTCBAR_MSK	ALTCBAR violation by PBL reset request mask
13	PBL error reset request event mask.

Table continues on the next page...

Field	Function
PBL_MSK	0b - PBL error event can cause a reset request 1b - PBL error event cannot cause a reset request
14 SFP_MSK	Security Fuse Processor error during POR fuse process reset mask. 0b - Security Fuse Processor error event can cause a reset request 1b - Security Fuse Processor error event cannot cause a reset request
15 SEC_MSK	Security monitor reset request event mask 0b - Security monitor reset request event mask can cause a reset request 1b - Security monitor reset request event mask cannot cause a reset request
16 SDC_MSK	Security Debug Controller error reset request mask 0b - SDC error event can cause a reset request 1b - SDC error event cannot cause a reset request
17 MBEE_MSK	Multi-bit ECC error reset request mask 0b - Multi-bit ECC error event can cause a reset request 1b - Multi-bit ECC error event cannot cause a reset request
18 —	Reserved
19 RPTOE_MSK	RCPM Time Out reset request event mask 0b - RCPM Time Out event can cause a reset request 1b - RCPM Time Out event cannot cause a reset request
20 SRDS_RST_MSK	SerDes reset request event mask 0b - SerDes reset event can cause a reset request 1b - SerDes reset event cannot cause a reset request
21 —	Reserved
22 CORE_WDOG_RST_MSK	Core watchdog reset request mask. 0b - Core watchdog reset request can cause a reset request 1b - Core watchdog reset request cannot cause a reset request
23-31 —	Reserved

12.3.14 Reset Request Status Register (RSTRQSR1)

12.3.14.1 Offset

Register	Offset
RSTRQSR1	C8h

12.3.14.2 Function

The RSTRQSR contains status bits to record the reasons for RESET_REQ_B assertion. The bits here are set independent of the RSTRQMR. This means if a reset reason occurs and is masked by RSTRQMR, it will still be recorded in RSTRQSR.

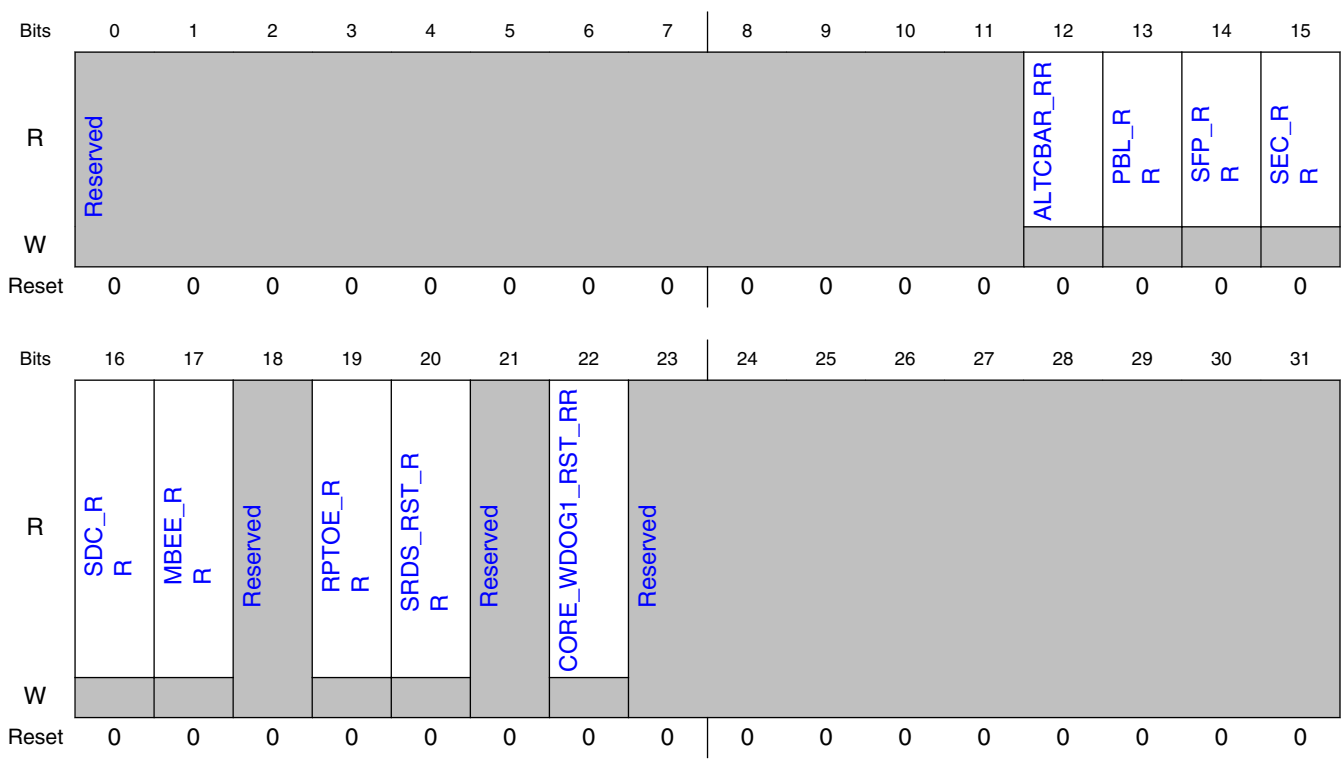
NOTE

For the different sources captured in this register, these must be serviced with PORESET_B only.

NOTE

When RESET_REQ_B is selected, the SFP_RR bit is reserved and PBL_RR is required only in PBI phase and not in the RCW loading stage.

12.3.14.3 Diagram



12.3.14.4 Fields

Field	Function
0-11 —	Reserved
12 ALTCBAR_RR	ALTCBAR violation by PBL reset request mask 0b - ALTCBAR violation by PBL can cause a reset request 1b - ALTCBAR violation by PBL can not cause a reset request
13 PBL_RR	PBL error reset request requires device level PORESET_B. 0b - PBL reset request event not active 1b - PBL reset request event active
14 SFP_RR	Security Fuse Processor error during POR fuse process caused reset request. Security Fuse Processor error requires device level PORESET._B 0b - Security Fuse Processor reset request event not active 1b - Security Fuse Processor reset request event active
15 SEC_RR	Security Monitor error during POR fuse process caused reset request. Security Monitor reached Hard Fail state and requires device level PORESET_B . 0b - Security Monitor reset request event not active 1b - Security Monitor reset request event active
16 SDC_RR	Security Debug Controller reset request Security Debug Controller requires device level PORESET_B . 0b - SDC reset request event not active 1b - SDC reset request event active
17 MBEE_RR	Multi-bit ECC reset request Platform internal memory multi-bit ECC error requires device level PORESET_B. **Note:** This bit is for multi-bit error in any of the SRAMs inside the chip including OCRAM and not for L1/L2 cache memories. 0b - Multi-bit ECC error reset request event not active 1b - Multi-bit ECC error reset request event active
18 —	Reserved Reserved
19 RPTOE_RR	RCPM Time Out reset request event RCPM Time Out event (for core halt, core stop, or core reset request) requires device level PORESET_B. 0b - RCPM Time Out event reset request event not active 1b - RCPM Time Out event reset request event active
20 SRDS_RST_RR	SerDes reset event. Occurs if any enabled SerDes PLL does not lock. 0b - SerDes reset request event not active 1b - SerDes reset request event active
21 —	Reserved
22 CORE_WDOG1_RST_RR	Core watchdog event reset request. 0b - Core watchdog reset request from WDOG1 is not active 1b - Core watchdog reset request from WDOG1 is active

Table continues on the next page...

Field	Function
23-31 —	Reserved

12.3.15 Boot Release Register (BRR)

12.3.15.1 Offset

Register	Offset
BRR	E4h

12.3.15.2 Function

The BRR contains control bits for enabling boot for each core. On exiting PORESET, the RCW BOOT_HO field optionally allows for logical core 0 to be released for booting or to remain in boot holdoff.

NOTE

The LSB, bit 31, is associated with Core 0.

NOTE

If a bit is changed from 1 to 0 outside of warm reset (at runtime), results are boundedly undefined for that core.

12.3.15.3 Diagram

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

Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Reserved															CR0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	u

12.3.15.4 Fields

Field	Function
0-30 —	Reserved
31 CR0	Core 0 Release. 0b - Core is in Boot Holdoff and not released for Booting 1b - Core released for Booting

12.3.16 Reset Control Word Status Register n (RCWSR0 - RCWSR15)

12.3.16.1 Offset

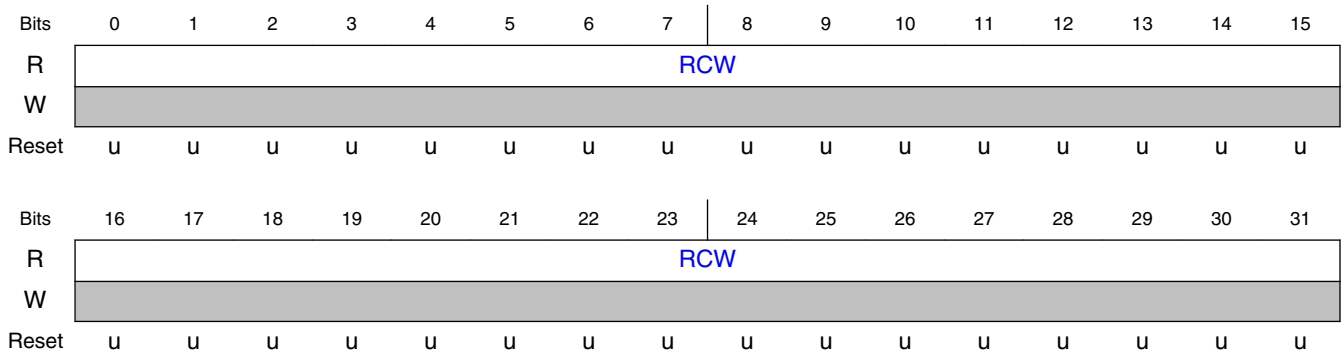
For a = 0 to 15:

Register	Offset
RCWSRa	100h + (a × 4h)

12.3.16.2 Function

RCWSR contains the Reset Configuration Word (RCW) information written with values read from flash memory by the device at power-on reset and read-only upon exiting reset.

12.3.16.3 Diagram



12.3.16.4 Fields

Field	Function
0-31 RCW	Read-only value of RCW bits (n-1)*32 : (n*32)-1 loaded in power-on reset's Reset Configuration stage.

12.3.17 Scratch Read / Write Register n (SCRATCHRW1 - SCRATCHRW4)

12.3.17.1 Offset

For a = 1 to 4:

Register	Offset
SCRATCHRWa	1FCh + (a × 4h)

12.3.17.2 Function

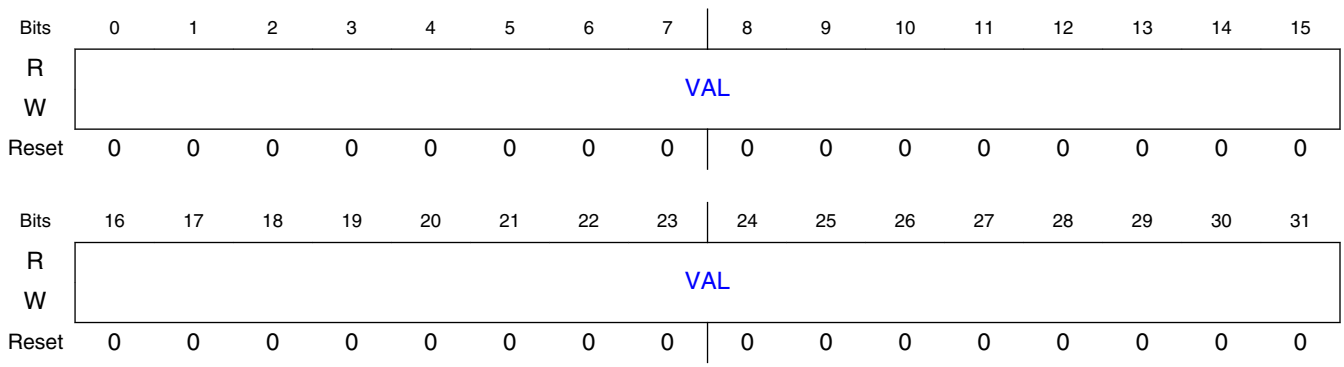
The SCRATCHRWn provides read / write scratch register locations available to the user.

NOTE

When performing secure boot, these registers are defined as follows:

- SCRATCHRW1 - Pointer to ESBC Header (Primary Boot Image)
- SCRATCHRW2 - Failure Code if Secure Boot Fails (Primary Boot Image)
- SCRATCHRW3 - Pointer to ESBC Header (Alternate Boot Image)
- SCRATCHRW4 - Failure Code if Secure Boot Fails (Alternate Boot Image)

12.3.17.3 Diagram



12.3.17.4 Fields

Field	Function
0-31 VAL	32-bit scratch contents

12.3.18 Scratch Read Register n (SCRATCHW1R1 - SCRATCHW 1R4)

12.3.18.1 Offset

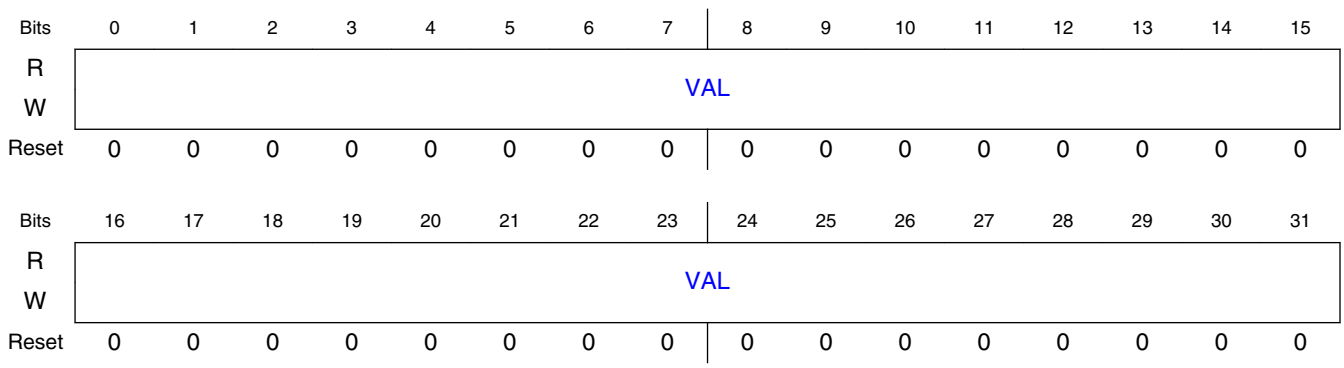
For a = 1 to 4:

Register	Offset
SCRATCHW1Ra	2FCh + (a × 4h)

12.3.18.2 Function

The SCRATCHW1Rn provides scratch register locations available to the user. These are write-once registers. After these have been written once, they can only be written after a power-on or hard reset.

12.3.18.3 Diagram



12.3.18.4 Fields

Field	Function
0-31 VAL	32-bit scratch contents

12.3.19 Core Reset Status Register n (CRSTSR0)

12.3.19.1 Offset

Register	Offset
CRSTSR0	400h

12.3.19.2 Function

The CRSTSRn contains the reset status bits for each thread on the device.

In this register (one per physical core):

- Bits 0-23 reflect per-thread conditions for this specific core
- Bits 24-31 reflect device conditions which in turn affect this specific core

NOTE

The number of CRSTSRn registers is determined by the number of physical cores.

Reset of this Register

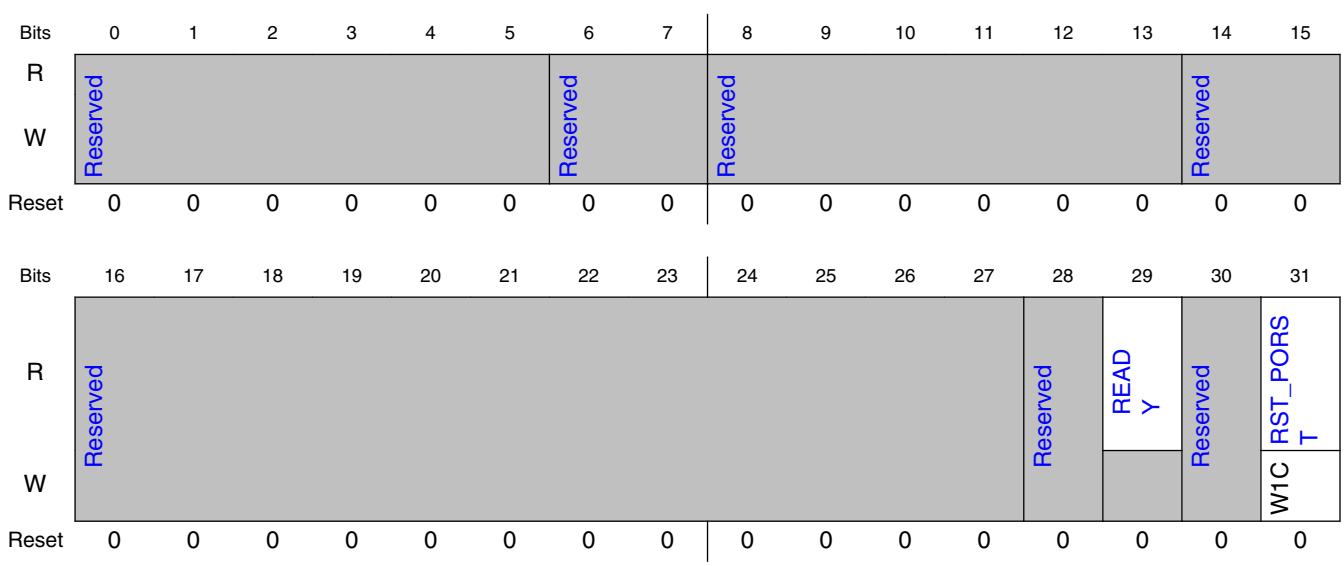
A power-on reset of the device causes the following to occur:

- RST_PORST is set
- All other bits are cleared

Ready Bit Functionality

This bit is cleared on a device power-on. Upon completion of power-on processing, this bit may be automatically set for a core if none of the conditions specified in the READY bit definition are true.

12.3.19.3 Diagram



12.3.19.4 Fields

Field	Function
0-5 —	Reserved
6-7 —	Reserved
8-13 —	Reserved
14-15 —	Reserved
16-27 —	Reserved
28 —	Reserved
29 READY	Core ready pin. Core is in the 'ready' state after device has successfully passed through the "System Ready" point and the core is currently not in any of the following states: • Core debug halted • Core debug stopped This bit reflects what is driven on the READY_P0 external signal. 0b - Core 0 not ready 1b - Core 0 ready
30 —	Reserved
31 RST_PORST	Core was reset due to a PORESET

12.3.20 DMA Control Register (DMACR1)

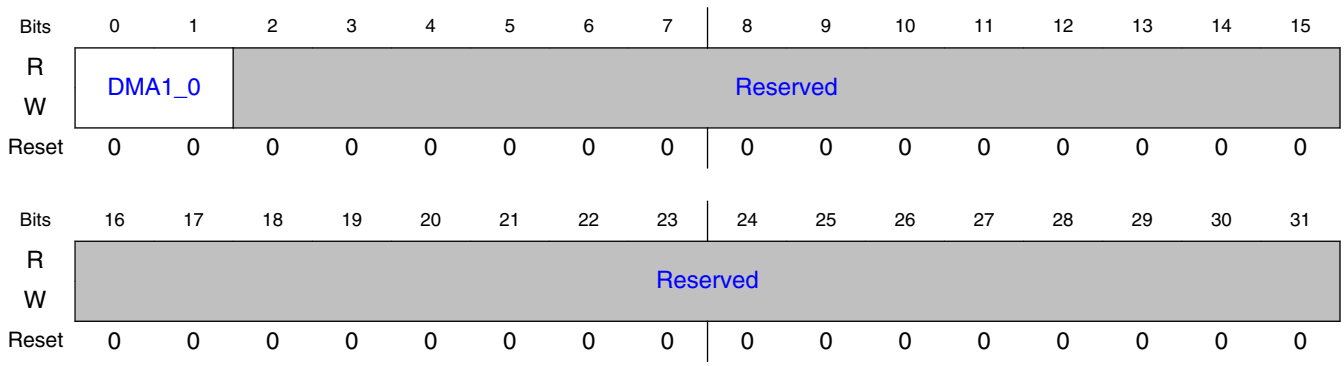
12.3.20.1 Offset

Register	Offset
DMACR1	608h

12.3.20.2 Function

The DMACR1 contains bits for allowing DMA transactions from internal sources on the device.

12.3.20.3 Diagram



12.3.20.4 Fields

Field	Function
0-1 DMA1_0	DMA 1, Channel 0. 00b - Reserved 01b - Reserved 10b - Reserved 11b - DMA's Channel may be initiated by EPU
2-31 —	Reserved

12.3.21 Topology Initiator Type n Register (TP_ITYP0 - TP_ITYP63)

12.3.21.1 Offset

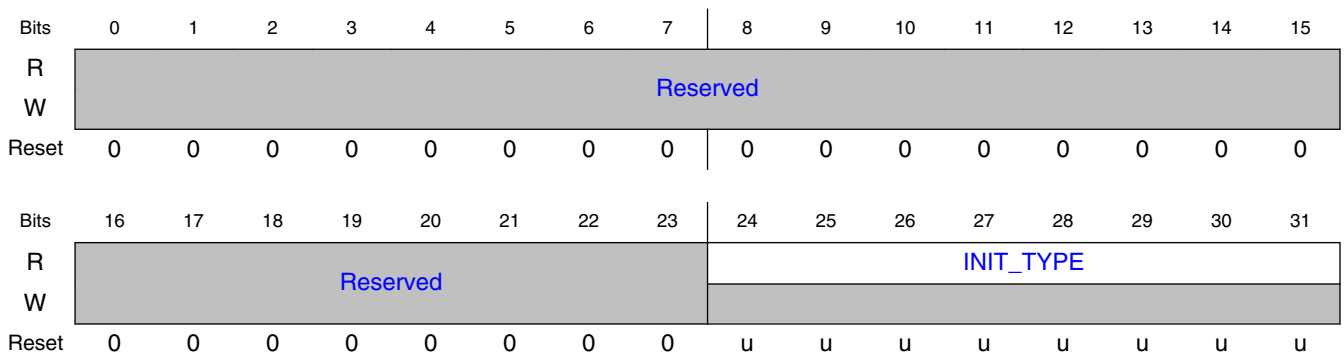
For a = 0 to 63:

Register	Offset
TP_ITYPa	740h + (a × 4h)

12.3.21.2 Function

Each Initiator Type Topology Register provides one entry of a 64 entry lookup table.

12.3.21.3 Diagram



12.3.21.4 Fields

Field	Function
0-23 —	Reserved
24-31 INIT_TYPE	Initiator Type. This field identifies the type of initiator (core or hardware accelerator) for this index. The lsb differentiates between an enabled and a disabled instance of the initiator. All encodings not listed below are reserved. 00000000b - Simple initiatior 00000010b - Arm Cortex A53 (disabled) 00000011b - Arm Cortex A53 (enabled)

12.3.22 Core Cluster n Topology Register (TP_CLUSTER1)

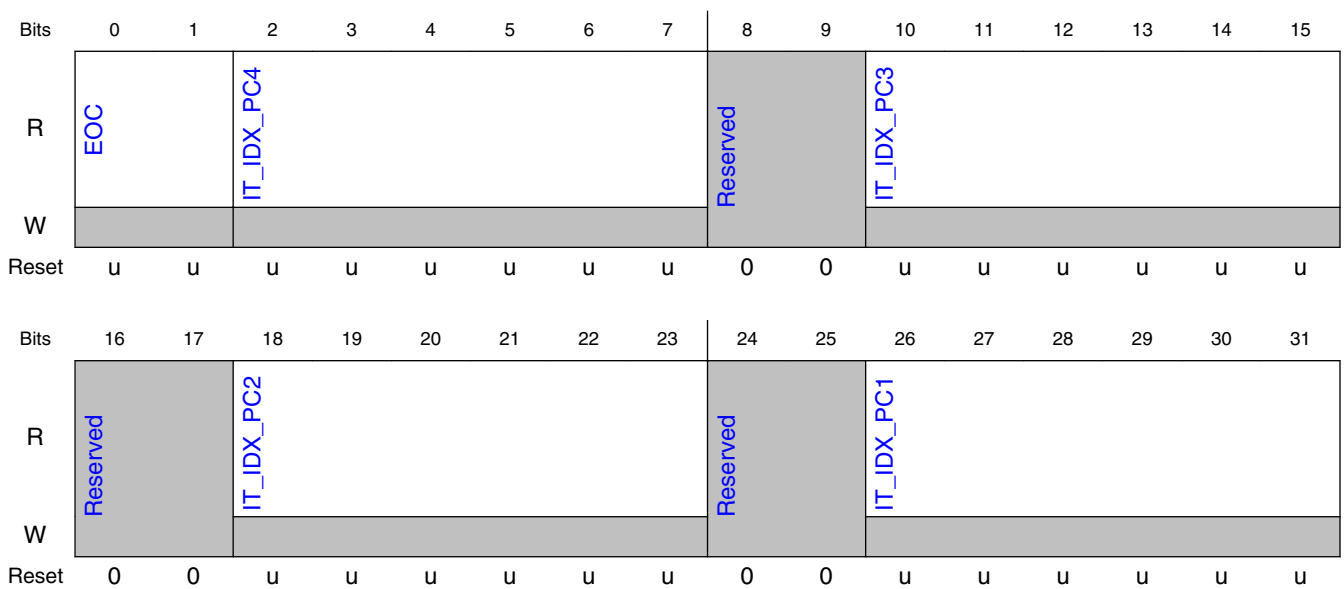
12.3.22.1 Offset

Register	Offset
TP_CLUSTER1	844h

12.3.22.2 Function

Each Topology Cluster Register (TP_CLUSTERn) contains four 6-bit fields, each of which is an index used for an initiator type lookup in a 64 entry initiator table implemented using the Topology Type Registers.

12.3.22.3 Diagram



12.3.22.4 Fields

Field	Function
0-1 EOC	End of Clusters - if the EOC field is non-zero, the register contains the information on the last cluster in the chip. 00 Not the last cluster 01,10,11 Last cluster in the chip
2-7	Initiator Type Index for this cluster's fourth initiator

Table continues on the next page...

DCFG_CCSR register descriptions

Field	Function
IT_IDX_PC4	Provides a 6-bit index for accessing an entry stored in one of the TP_ITYPn registers. This index selects which of the 64 TP_ITYPn registers contains the identification information for this initiator.
8-9 —	Reserved
10-15 IT_IDX_PC3	Initiator Type Index for this cluster third initiator Provides a 6-bit index for accessing an entry stored in one of the TP_ITYPn registers. This index selects which of the 64 TP_ITYPn registers contains the identification information for this initiator.
16-17 —	Reserved
18-23 IT_IDX_PC2	Initiator Type Index for this cluster second initiator Provides a 6-bit index for accessing an entry stored in one of the TP_ITYPn registers. This index selects which of the 64 TP_ITYPn registers contains the identification information for this initiator.
24-25 —	Reserved
26-31 IT_IDX_PC1	Initiator Type Index for this cluster first initiator Provides a 6-bit index for accessing an entry stored in one of the TP_ITYPn registers. This index selects which of the 64 TP_ITYPn registers contains the identification information for this initiator.

Chapter 13
Run Control and Power Management (RCPM)

13.1 Introduction

This chapter provides the specification and programming model for the RCPM.

13.1.1 Overview

The Run Control and Power Management (RCPM) module communicates with embedded cores, coherency modules, and other device platform module to provide run control and power management functionality. The device can be placed into a range of low power states, via the RCPM (and its associated RCPM driver), to significantly reduce dynamic power consumption at the processor core, cluster, and device level. The RCPM also provides the functionality used to return processors, clusters, and device platform module to full operation in response to wake-up events such as external signals, timers, interrupts, and network traffic.

A complete whitepaper on QorIQ Power Management can be found at QorIQ Power Management.

13.1.2 The RCPM module as implemented on the chip

This section provides details about how the RCPM module is integrated into this chip.

The RCPM module supports multithreaded core; however, the A53 core implemented on chip are single-threaded. The following table describes the mapping of the referenced RCPM thread and physical core to the Arm Cortex-A53 core:

Table 13-1. RCPM/LS1012A thread and physical core *n* mapping

RCPM thread n	LS1012A core	RCPM physical core n	LS1012A core
0	core 0	0	core 0

13.1.3 Power Management Features

- Software controlled power management, which minimizes power consumption of blocks when they are idle.
- Software-controlled power management states
  - Per core STANDBYWFI (PW15)
  - Per device LPM20
- Supports interrupt controller interrupt-based wake-up and the following wake-up sources:
  - Wake on internal timer event.
  - Wake on internal and external interrupt event.
- Enter software-controlled power state through core
- Core power management:
  - Independent power management control of core
  - STANDBYWFI (PW15) state where:
    - Entry into PW15 mode core power management state via wait for interrupt instruction execution by the core
    - STANDBYWFI (PW15) state wake up source is captured in Table 13-4
- Cluster power management:
  - WFI instruction by the core would initiate L2 Cache of the cluster to go into STANDBY mode.
  - The entry and exit mode of cluster power management is captured in Modes Entry and Exit for Power Management
- Device Power Management:
  - LPM20 state where:
    - The core is in STANDBYWFI (PW15) state.
    - Platform clock is disabled.
    - Entry into LPM20 via setting POWMGTCSR[LPM20_REQ] as well as WFI Instruction execution from the core.
    - LPM20 state wake up source is captured in Table 13-5
  - Independent Device wake up from:
    - Unmasked interrupt configured in GIC-400 and RCPM registers
    - Unmasked critical interrupt

13.1.4 Power Management States

The RCPM module supports the following Power Management modes of operation:

- Power management states
  - Core(s)
    - Core(s) Full On state
    - Core in STANDBYWFI (PW15) state
  - Device
    - Device Full On state
    - Device in LPM20 state
- Reset modes

The different states available are summarized in Table 13-2 through Table 13-5.

13.1.4.1 Power Management State Summary

The following table summarizes the core power management states. For detailed description of power management states, refer Modes Entry and Exit for Power Management.

Table 13-2. Core and Cluster Power Management State Summary

Power Management State Name	Resumable [1]	Services Snoops	Device Clocks	Core Clocks	State Retained	Description
Full On	-	Y	Y	Y	Y	The *core* referred to is currently not in the STANDBYWFI (PW15) state.
STANDBYWFI (PW15)	Y	Y	Y	N	Y	The *core* referred to is currently in the STANDBYWFI (PW15) state.

1. A resumable power state in the core indicates that the core can exit from a power managed state and return to Full On without the core being reset.

Table 13-3 summarizes the device RCPM modes.

Table 13-3. Device Power Management State Summary

Power Management State Name	Resumable [1]	Services Snoops	Device Clocks	Core Clocks	Description
Full On	-	Y	Y	Y	Platform clock to all IP modules are not gated.
LPM20	Y	N	N	N	Platform clock to all IP modules are gated off.

1. A resumable power state for the device indicates that the device can exit from a power managed state and return to Full On without the device being reset.

13.1.5 Modes Entry and Exit for Power Management

Table 13-4 summarizes the entry and exit for the core power management modes.

Table 13-4. Core and Cluster Power Management States: Entry and Exit

Power Management State Name	Entry via ...	Exit via ...	
		Temporary exit	Permanent exit
STANDBYWFI (PW15)	• Core Executes WFI (wait for interrupt) instruction.	• A snoop request that must be serviced by the core L1 data cache. • A cache or TLB maintenance operation that must be serviced by the core L1 instruction cache, data cache, or TLB. • A core debug halt request. • An APB access to the debug or trace registers residing in the core power domain.	• Core warm reset or any device-level reset • Any interrupt request.[1] • A core debug halt request.[2] • Debug interrupt without masking • Wake on software interrupt • Wake on peripheral interrupt • Wake on PORESET • Wake on debug halt • Wake on debug interrupt

1. Interrupts may still be masked in the GIC. Masking in the GIC prevents interrupt delivery to the core and therefore does not cause an exit.
2. For a core debug halt request to cause an exit from power management mode, the core must be operating at a frequency greater than the platform frequency.

NOTE

The power management states of the core can be achieved with the software sequence without the RCPM interaction. The registers given below provides the different status of the core:

- TWAITSR0: Status shows core is in the WFI state
- POWMGTCSR: LPM20 request and status

Table 13-5 summarizes the entry and exit for the device power management modes.

Table 13-5. Device RCPM Mode: Entry and Exit for Power Management

Device Power Management State Name	Entry via ...	Exit via ...
LPM20	LPM20 operation can be triggered by the following mechanism: • Set CPUECTLR[2:0] CPU retention control to a non-zero value. • Set CPUECTLR[6] SMPEN to 1. The cluster power management controller uses the state of this bit to decide whether to put the core into retention (equals to one) or powerdown(equals to zero). • Write to generic timer CNCTR register to enable the Arm counter CNTVALUEB[63:0]. • Set the SCFG_COREPMCR[WFIL2]. • Set RCPM_POWMGTCSR[LPM20_REQ]. • Execute WFI instruction on each core. After these programming ASLEEP pin should get asserted. **NOTE:** When the core is in the WFI state, the device cannot be configured in the LPM20 state until the core comes out of the WFI state. To configure the device in sleep state (LPM20) when the core is already in the WFI state, the external host should first bring the core out of WFI through software-generated interrupt. It should then wait for the interrupt to be cleared and the WFI status bits to be cleared in the SCFG_LPMCSR register before following the LPM20 sequence to put the device in sleep state. **NOTE:** When the core is in boot hold-off state, the device cannot be put in sleep state. To follow the LPM20 sequence in such cases, the core should be first brought out from the boot hold-off state.	• Wake on PFE MACs with RGMII support - Magic Packet, if programmed • Wake on IIC1, if programmed • Wake on FlexTimer1, if programmed • Wake on GPIO, if programmed • Wake on PORESET

13.1.6 Power Management States

This section provides functional explanation of the power management states summarized in the previous sections.

The privileged software can place the device in one of the following power states:

- STANDBYWFI (PW15) state
- LPM20

13.1.6.1 STANDBYWFI (PW15) State

In STANDBYWFI (PW15) state, the core internally gates clocks within the core, significantly reducing the power consumption of the core. Snooping of the L1 and L2 are still supported and thus the data in the data cache is kept coherent. Interrupts directed to the Core are monitored by the GIC and cause core to exit from STANDBYWFI (PW15) state to allow the core to recognize and process the interrupt. STANDBYWFI (PW15) state is entered through execution of wait for interrupt instruction from the core.

The watchdog timer facilities are still enabled during STANDBYWFI (PW15) state.

13.1.6.2 LPM20 State

LPM20 is a device low power state. Prior entering LPM20, software is expected to quiesce all external devices/internal IP modules through configuration. RCPM hardware is designed to implement sanity check for IP modules idle status.

In LPM20 state, all the modules in the chip are clock gated including the A53 core except I2C1, GPIO, PFE MACs, and FlexTimer1 so that only those modules which are required to wake up the device will still have a running clock.

The modules which can be used as a wake up source are internal timers, internal and external interrupts.

The following figure describes the sleep flow for the chip.

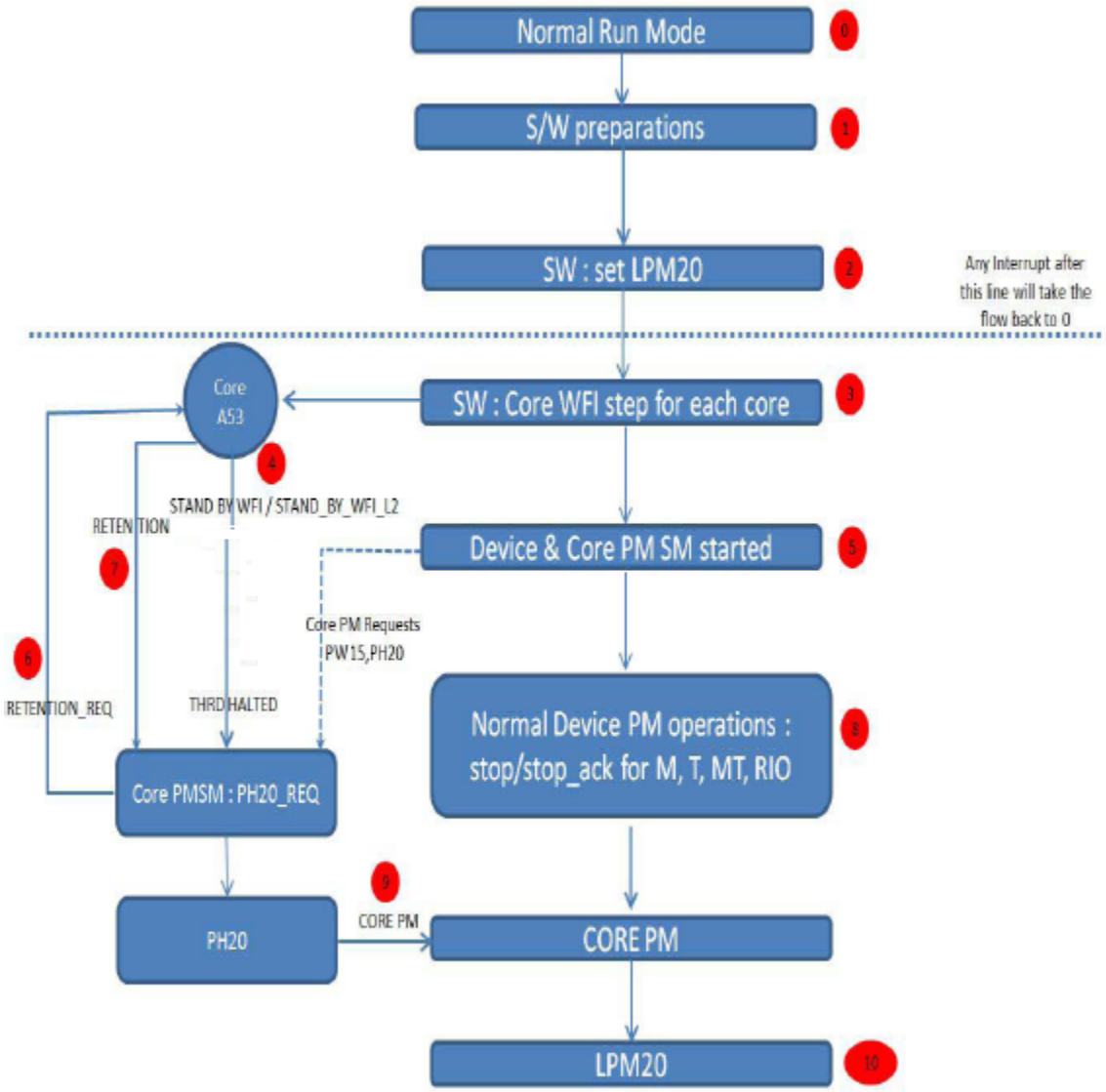


Figure 13-1. Device LPM20 operation sequence

NOTE

In the chip, the Arm cluster does not support PH20 (retention) mode. As part of LPM20 sequence, the chip requests for cluster to enter into retention mode which it immediately acknowledges without actually going into retention mode.

13.1.7 Reset Modes

13.1.7.1 Power-On Reset State

During power-on reset, the following hardware structures are reset:

- All CCSR registers
- SoC power management state machine
- Core power management state machine
- Cluster power management state machine

13.2 External Signal Description

The table below provides the signal description of power management signals:

Table 13-6. RCPM Detailed Signal Descriptions

Signal	I/O	Description	
ASLEEP (optional)	O	Asleep. After negation of PORESET, ASLEEP is asserted until the device completes its power-on reset sequence and reaches its ready state.	
		State Meaning	Asserted- Indicates that the device is either still in its power-on reset sequence or it has reached a LPM20 state after a power-down command is issued by software. Negated- The device is not in LPM20 state. (It has either awake from a power-down state, or has completed the POR sequence.)
		Timing	Assertion- May occur at any time; may be asserted asynchronously to the input clocks. Negation- Negates synchronously with SYSCLK when leaving power-on sequence; otherwise negation is asynchronous.

13.3 RCPM register descriptions

This section identifies power management resources that are not included as part of a processor core or platform IP.

13.3.1 RCPM Memory map

RCPM base address: 1EE_2000h

Offset	Register	Width (In bits)	Access	Reset value
4Ch	Thread Wait status Register (TWAITSR)	32	RO	0000_0000h
130h	Power Management Control and Status Register (POWMGTCSR)	32	RW	0000_0000h
140h	IP Powerdown Exception Control Register (IPPDEXPCR)	32	RW	0000_0000h
15Ch	nIRQOUT interrupt mask register (nIRQOUTR)	32	RW	0000_0000h
16Ch	nFIQOUT Interrupt Register (nFIQOUTR)	32	RW	0000_0000h

13.3.2 Thread Wait status Register (TWAITSR)

13.3.2.1 Offset

Register	Offset
TWAITSR	4Ch

13.3.2.2 Function

This register is used for reporting wait status per core.

13.3.2.3 Diagram

Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	T31_T0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	T31_T0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

13.3.2.4 Fields

Field	Function
0-31 T31_T0	Core STANDBYWFI (PW15) Status. **NOTE:** Bit 31 corresponds to core 0, bit 30 corresponds to core 1, and so on. 00000000000000000000000000000000b - Core is not in the STANDBYWFI (PW15) state 00000000000000000000000000000001b - Core is in the STANDBYWFI (PW15) state

13.3.3 Power Management Control and Status Register (POWMGTCSR)

13.3.3.1 Offset

Register	Offset
POWMGTCSR	130h

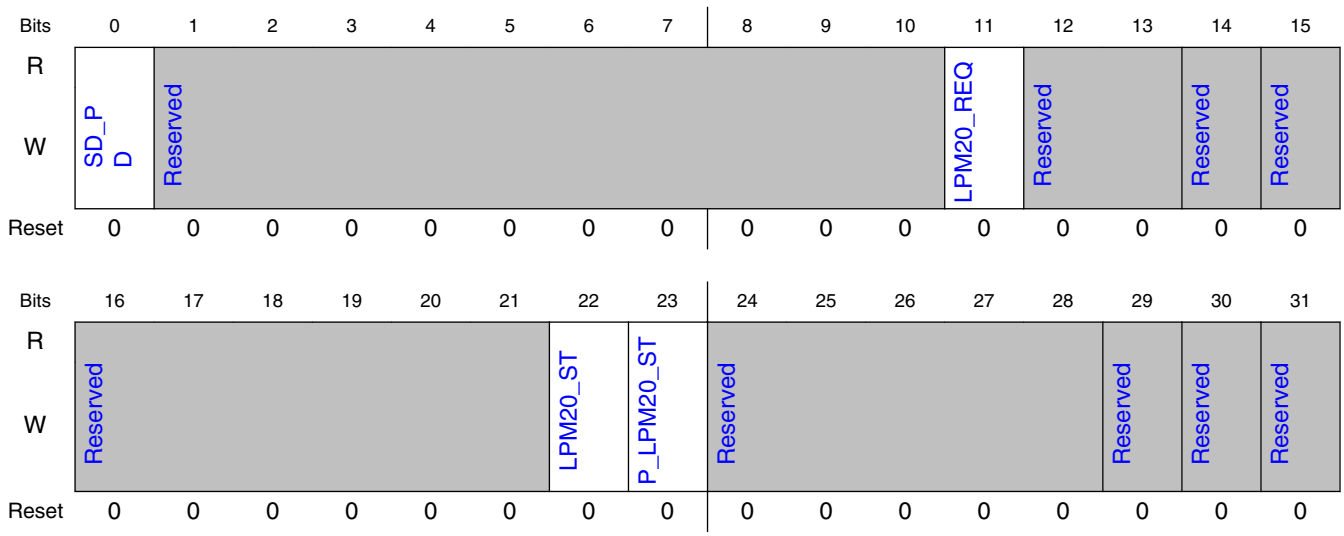
13.3.3.2 Function

This register is used for entering the chip's LPM state. In addition, it provides status indicating successful entry into the corresponding power management state.

NOTE

The following table describes this register's bit settings.

13.3.3.3 Diagram



13.3.3.4 Fields

Field	Function
0 SD_PD	SerDes and Protocol Converter Powerdown Control 0b - SerDes and Protocol Converter are not powerdown in LPM state. 1b - SerDes and Protocol Converter are powerdown in LPM state for further power saving. If this bit is set, after system wake up from LPM state, SerDes has to go through the training sequence to return back to function.
1-10 —	- Reserved
11 LPM20_REQ	LPM20 State Request. This bit is clear by interrupt event. 0b - No request to put chip in LPM20 state 1b - Request to place chip in LPM20 state.
12-13 —	- Reserved
14 —	- Reserved
15-21 —	- Reserved
22 LPM20_ST	LPM20 Status 0b - Device is not attempting to reach LPM20 state 1b - The chip is attempting to enter LPM20 state because POWMGTCSR[LPM20_REQ] is set.
23 P_LPM20_ST	Previous LPM20 Status. It is used by software to know which state the chip was in before being wake up. Before software set POWMGTCSR[LPM20_REQ], we expect software w1c [P_LPM20_ST] Bit. 0 Device was not in LPM20 state before being wake up by interrupt.

Table continues on the next page...

Field	Function
	1 Device was in LPM20 state before being wake up by interrupt. The bit is set when LPM20_ST transitions from 1 to 0 due to an interrupt. The bit is w1c.
24-28 —	- Reserved
29 —	- Reserved
30 —	- Reserved
31 —	- Reserved

13.3.4 IP Powerdown Exception Control Register (IPPDEXPCR)

13.3.4.1 Offset

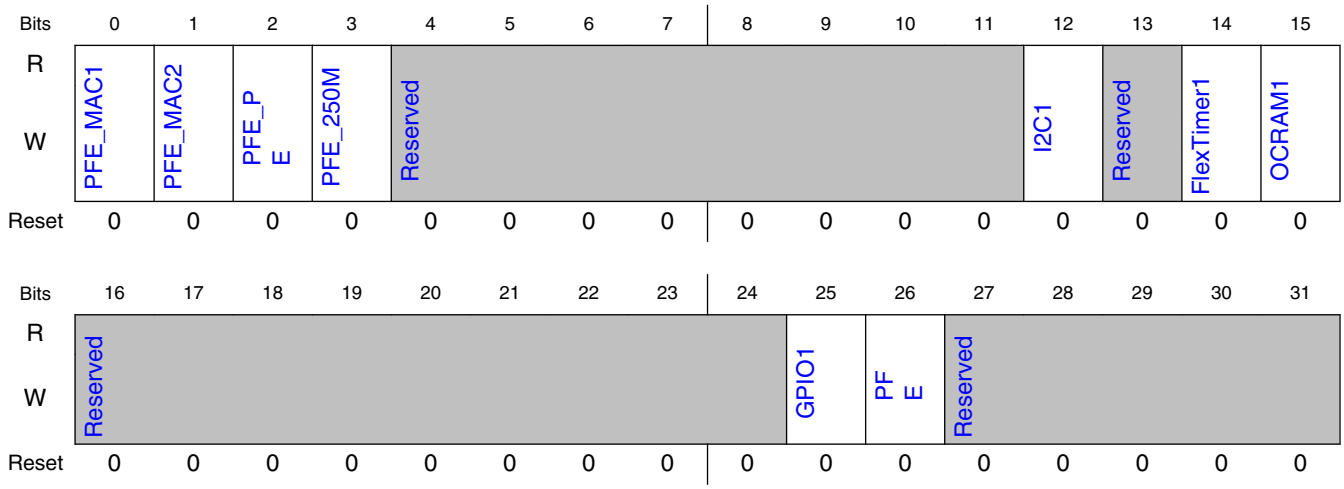
Register	Offset
IPPDEXPCR	140h

13.3.4.2 Function

IPPDEXPCR provides a mechanism for excluding certain IP blocks from device LPM20 mode to make these IP blocks available as sources for wake-up events. For example,

- Wake on LAN (magic packet) - Requires excluding the Ethernet controller from device LPM20 mode
- Wake on GPIO - Requires excluding the GPIO controller from device LPM20 mode

13.3.4.3 Diagram



13.3.4.4 Fields

Field	Function
0 PFE_MAC1	PFE MAC1 powerdown exception 0b - PFE MAC1 powerdown during device LPM20 1b - PFE MAC1 is not powerdown during device LPM20
1 PFE_MAC2	PFE MAC2 powerdown exception. This bit is required for wake on magic packet (RGMII interface). 0b - PFE MAC2 powerdown during device LPM20 1b - PFE MAC2 is not powerdown during device LPM20
2 PFE_PE	PFE PE clock. This bit is required for wake on magic packet. 0b - PFE PE clock powerdown during device LPM20 1b - PFE PE clock is not powerdown during device LPM20
3 PFE_250M	PFE logic clock (250 MHz). This bit is required for wake on magic packet. 0b - PFE logic clock powerdown during device LPM20 1b - PFE logic clock is not powerdown during device LPM20
4-11 —	- Reserved
12 I2C1	I^2C1 powerdown exception 0b - I^2C1 powerdown during device LPM20 1b - I^2C1 is not powerdown during device LPM20
13 —	- Reserved
14 FlexTimer1	FlexTimer1 powerdown exception 0b - FlexTimer1 powerdown during device LPM20 1b - FlexTimer1 is not powerdown during device LPM20
15	OCRAM1 powerdown exception

Table continues on the next page...

Field	Function
OCRAM1	0b - OCRAM 1 powerdown during device LPM20 1b - OCRAM 1 is not powerdown during device LPM20
16-24 —	- Reserved
25 GPIO1	GPIO powerdown exception 0b - GPIO powerdown during device LPM20 1b - GPIO is not powerdown during device LPM20
26 PFE	PFE powerdown exception. This bit is required for wake on magic packet. 0b - PFE powerdown during device LPM20 1b - PFE is not powerdown during device LPM20
27-31 —	- Reserved

13.3.5 nIRQOUT interrupt mask register (nIRQOUTR)

13.3.5.1 Offset

Register	Offset
nIRQOUTR	15Ch

13.3.5.2 Function

The register masks the nIRQOUT Interrupt from GIC-400 for Sleep/LPM20 mode.

13.3.5.3 Diagram

Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	IM0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	IM0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

13.3.5.4 Fields

Field	Function
0-31 IM0	Interrupt mask core 0-31. **NOTE:** Bit 31 corresponds to Core 0, bit 30 corresponds to Core 1, and so on.

13.3.6 nFIQOUT Interrupt Register (nFIQOUTR)

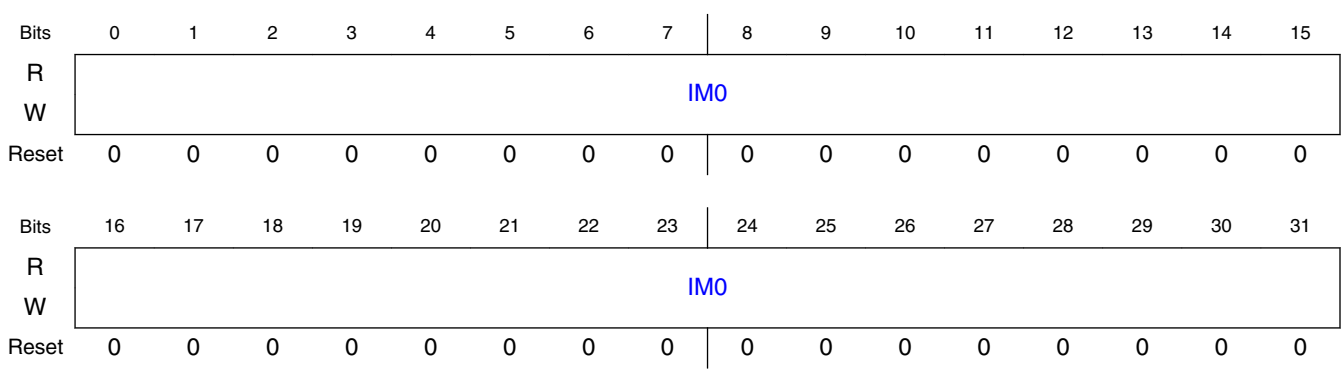
13.3.6.1 Offset

Register	Offset
nFIQOUTR	16Ch

13.3.6.2 Function

The register masks the nFIQOUT interrupt from GIC-400 for Sleep/LPM20 mode. The LPM20 FSM recognizes the corresponding core interrupt from GIC-400 and initiates a wake-up sequence provided the interrupt mask is not set.

13.3.6.3 Diagram



13.3.6.4 Fields

Field	Function
0-31 IM0	Interrupt mask core 0-31. **NOTE:** Bit 31 corresponds to Core 0, bit 30 corresponds to Core 1, and so on.

13.4 RCPM functional description

The RCPM supports minimizing the power consumption through software controlled power management states - for individual core (STANDBYWFI (PW15)) and the device (LPM20).

RCPM can handshake with various IP modules at the SoC level to gracefully stop the IP traffic and direct the memory controller to put DDR into self-refresh mode (if enabled).

The RCPM allows several wake-up event sources to exit low power state (internal timer, internal and external interrupts).

The wake-up events are mapped to interrupt controller interrupts to generate a wake-up interrupt to the core. For information on operation entry or exit, refer Modes Entry and Exit for Power Management.

Chapter 14
Packet Forwarding Engine (PFE)

14.1 Introduction

LS1012A uses a hardware packet forwarding engine to provide high performance Ethernet interfaces. The device includes two Ethernet ports. Both Ethernet ports support SGMII. One port also has the option to support RGMII (10/100M half-duplex mode or 1G full-duplex mode). These configurations conform to IEEE 802.3 and support full-duplex and half-duplex operation at 10/100/1000/2500 Mbps.

14.2 Features

The packet forwarding engine (PFE) provides the following features:

- Performs the IEEE 802.3 protocol for 10/100/1000/2500 Mbps (RGMII is up to 1000 Mbps only)
- Supports packet sizes from 64 bytes up to 10240 bytes for jumbo frames
- VLAN packet identification and tagging
- Address recognition circuit for efficient bridging. 32 specific address registers for filtering
- Capable of autonomously handling all packets belonging to a given stream, without Host CPU intervention, following stream creation
- Capable of addressing DDR, security co-processor and internal on-chip memory
- Capable of routing/bridging an aggregate of 2 Gbps of traffic, at any packet size, must support 2 Gbps for IPv4/NAT and IPv6 for a single connection
- QoS Compliant with HGI 2.0 at any packet size
- Two independent MAC ports:
  - Port 1 supports SGMII
  - Port 2 supports RGMII/SGMII
- Includes JTAG and UART interfaces for debugging

14.3 Functional description

The following sections describe the functionality of the PFE.

14.3.1 Block diagram

The following figure illustrates the PFE block diagram.

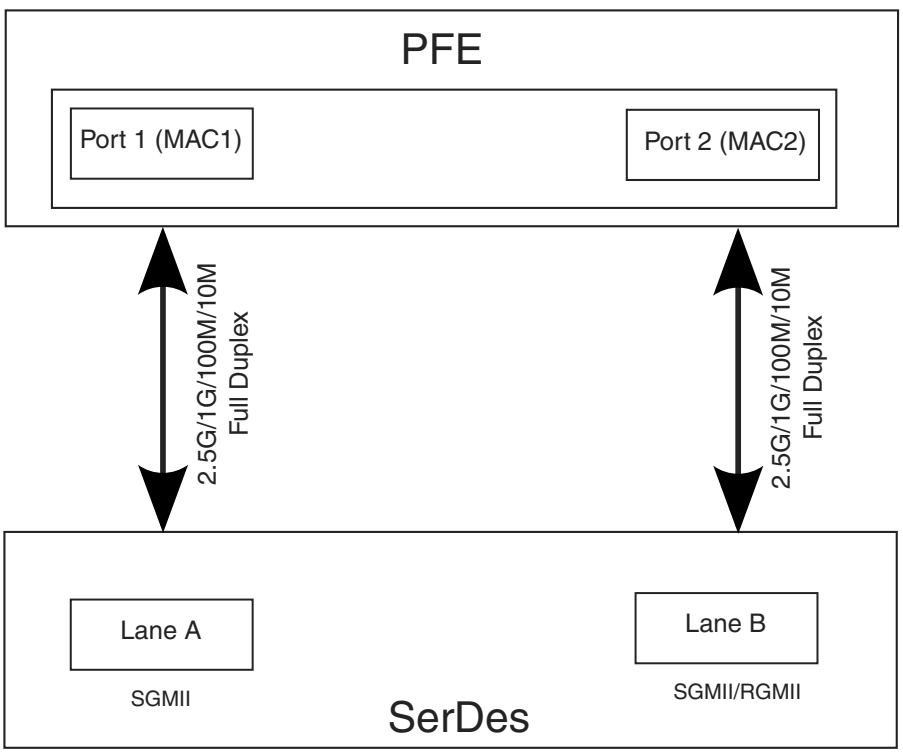


Figure 14-1. PFE block diagram

14.3.2 MDIO to PFE port connectivity

In LS1012A, there are two MDIO controllers coming out from PFE and one MDIO per MAC is provided. The MDIO ports are connected in the chip as follows:

- MDIO1: It has connectivity for external off-chip PHY registers as well as on-chip SGMII1 PCS registers. The selection occurs based on the SCFG_MDIOSELCR[MDIOSEL] bit as follows:

    - 0: MDIO from SerDes
    - 1: MDIO from external Ethernet PHY
- MDIO2: It has connectivity for on-chip SGMII2 PCS registers.

Chapter 15
DUART

15.1 The DUART module as implemented on the chip

This section provides details about how the DUART module is integrated into this chip.

The chip implements a single DUART module that contains UART1 and UART2.

In addition, UART2 module consists of clear to send (CTS) input port and request to send (RTS) output port for data flow control.

Table 15-1. DUART module as implemented on chip

Module name	Module base address
DUART1	0x21C_0000

15.2 Overview

This chapter describes the dual universal asynchronous receiver/transmitters (DUART). It describes the functional operation, the initialization sequence, and the programming details for the dual UART registers and features.

The dual UART consists of two universal asynchronous receiver/transmitters (UARTs).The UARTs act independently; all references to UART refer to one of these receiver/transmitters. Each UART is clocked by the platform clock/2. The dual UART programming model is compatible with the PC16552D.

The UART interface is point to point, meaning that only two UART devicedsscs are attached to the connecting signals. As shown in the figure below, each UART module consists of the following:

- Receive and transmit buffers

- 16-bit counter for baud rate generation
- Interrupt control logic

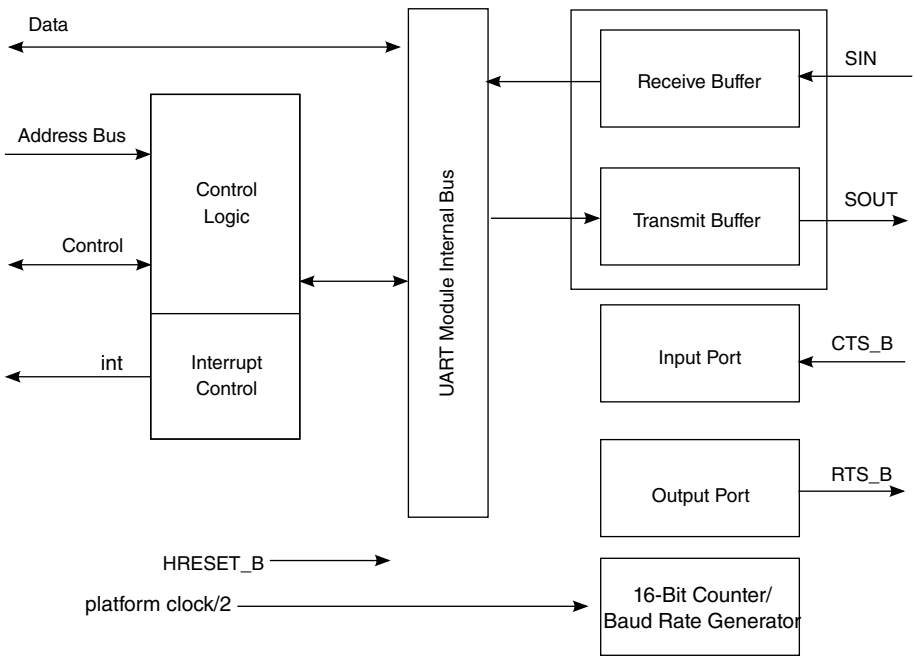


Figure 15-1. UART block diagram

NOTE

HRESET_B is not supported on this chip.

15.2.1 Features

The DUART includes these distinctive features:

- Full-duplex operation
- Programming model compatible with original PC16450 UART and PC16550D (improved version of PC16450 that also operates in FIFO mode)
- PC16450 register reset values
- Configurable FIFO mode for both transmitter and receiver, providing 64-byte FIFOs
- Serial data encapsulation and decapsulation with standard asynchronous communication bits (START, STOP, and parity)
- Maskable transmit, receive, line status, and modem status interrupts
- Software-programmable baud generators that divide the platform clock/2 by 1 to (2^{16} - 1) and generate a 16x clock for the transmitter and receiver engines
- Clear to send (CTS_B and ready to send (RTS_B) modem control functions

- Auto flow for clear to send (CTS_B) and ready to send (RTS_B) modem control functions
- Software-selectable serial interface data format (data length, parity, 1/1.5/2 STOP bit, baud rate)
- Line and modem status registers
- Line-break detection and generation
- Internal diagnostic support, local loopback, and break functions
- Prioritized interrupt reporting
- Overrun, parity, and framing error detection

15.2.2 Modes of operation

The communication channel provides a full-duplex asynchronous receiver and transmitter using an operating frequency derived from the one-half of the platform clock.

The transmitter accepts parallel data from a write to the transmitter holding register (UTHR). In FIFO mode, the data is placed directly into an internal transmitter shift register of the transmitter FIFO. The transmitter converts the data to a serial bit stream inserting the appropriate start, stop, and optional parity bits. Finally, it outputs a composite serial data stream on the channel transmitter serial data output signal (SOUT). The transmitter status may be polled or interrupt driven.

The receiver accepts serial data bits on the channel receiver serial data input signal (SIN), converts it to parallel format, checks for a start bit, parity (if any), stop bits, and transfers the assembled character (with start, stop, parity bits removed) from the receiver buffer (or FIFO) in response to a read of the UART's receiver buffer register (URBR). The receiver status may be polled or interrupt driven.

15.3 DUART external signal descriptions

The DUART signals are described in the table below.

NOTE

Although the actual device signal names are prepended with the UART_ prefix as shown in the table, the abbreviated signal names are often used throughout this chapter.

Table 15-2. DUART signals-detailed signal descriptions

Signal	I/O	Description	
UART*n*_SIN	I	Serial data in. Data is received on the receivers of UART*n* through the respective serial data input signal, with the least-significant bit received first.	
		State meaning	Asserted/Negated-Represents the data being received on the UART interface.
		Timing	Assertion/Negation-An internal logic sample signal, *rxcnt*, uses the frequency of the baud-rate generator to sample the data on SIN.
UART*n*_SOUT	O	Serial data out. The serial data output signals for the UART*n* are set ('mark' condition) when the transmitter is disabled, operating in the local loopback mode, or idle. Data is shifted out on these signals, with the least significant bit transmitted first.	
		State meaning	Asserted/Negated-Represents the data being transmitted on the respective UART interface.
		Timing	Assertion/Negation- An internal logic sample signal, *rxcnt*, uses the frequency of the baud-rate generator to update and drive the data on SOUT.

15.4 DUART register descriptions

The table below lists the DUART registers and their offsets. It lists the address, name, and a cross-reference to the complete description of each register.

The UARTs on the device are identical, except that the registers for each UART are located at different offsets. Throughout this chapter, the registers are described by a singular acronym: for example, LCR represents the line control register for each UART (see "The DUART module as implemented on the chip" section for the number of UARTs supported on chip).

The registers in each UART interface are used for configuration, control, and status. The divisor latch access bit, ULCR[DLAB], is used to access the divisor latch least- and most-significant bit registers and the alternate function register. Refer to UART line control register (ULCR1 - ULCR2) , for more information on ULCR[DLAB].

All the DUART registers are one-byte wide. Reads and writes to these registers must be byte-wide operations. The table below provides a register summary with references to the section and page that contains detailed information about each register. Undefined byte address spaces within offset 0x000-0xFFF are reserved.

15.4.1 DUART Memory map

DUART base address: 21C_0000h

Offset	Register	Width (In bits)	Access	Reset value
500h	UART divisor least significant byte register (UDLB1)	8	RW	00h
500h	UART receiver buffer register (URBR1)	8	RO	00h
500h	UART transmitter holding register (UTHR1)	8	WO	00h
501h	UART divisor most significant byte register (UDMB1)	8	RW	00h
501h	UART interrupt enable register (UIER1)	8	RW	00h
502h	UART alternate function register (UAFR1)	8	RW	00h
502h	UART FIFO control register (UFCR1)	8	WO	00h
502h	UART interrupt ID register (UIIR1)	8	RO	01h
503h	UART line control register (ULCR1)	8	RW	00h
505h	UART line status register (ULSR1)	8	RO	60h
507h	UART scratch register (USCR1)	8	RW	00h
510h	UART DMA status register (UDSR1)	8	RO	01h
600h	UART divisor least significant byte register (UDLB2)	8	RW	00h
600h	UART receiver buffer register (URBR2)	8	RO	00h
600h	UART transmitter holding register (UTHR2)	8	WO	00h
601h	UART divisor most significant byte register (UDMB2)	8	RW	00h
601h	UART interrupt enable register (UIER2)	8	RW	00h
602h	UART alternate function register (UAFR2)	8	RW	00h
602h	UART FIFO control register (UFCR2)	8	WO	00h
602h	UART interrupt ID register (UIIR2)	8	RO	01h
603h	UART line control register (ULCR2)	8	RW	00h
605h	UART line status register (ULSR2)	8	RO	60h
607h	UART scratch register (USCR2)	8	RW	00h
610h	UART DMA status register (UDSR2)	8	RO	01h

15.4.2 UART divisor least significant byte register (UDLB1 - UDLB2)

15.4.2.1 Offset

For a = 1 to 2:

Register	Offset
UDLBa	400h + (a × 100h)

15.4.2.2 Function

This register is accessible when ULCR[DLAB] = 1.

The divisor least significant byte register (UDLB) is concatenated with the divisor most significant byte register (UDMB) to create the divisor used to divide the input clock into the DUART. The output frequency of the baud generator is 16 times the baud rate; therefore the desired baud rate = platform clock frequency/2 ÷ (16 x [UDMB||UDLB]). Equivalently, [UDMB||UDLB:0b0000] = platform clock frequency/2 ÷ desired baud rate. Baud rates that can be generated by specific input clock frequencies are shown in the table below.

The following table shows examples of baud rate generation based on common input clock frequencies. Many other target baud rates are also possible.

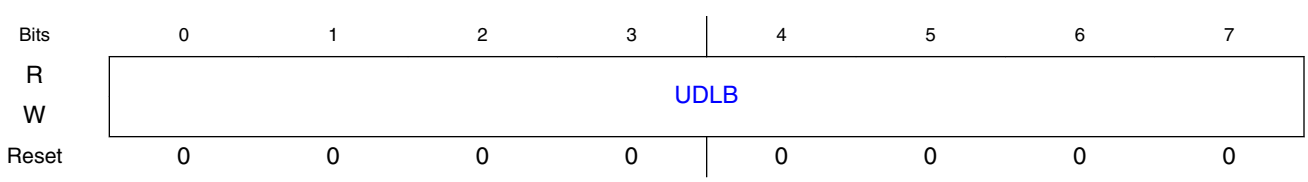
NOTE

Because only integer values can be used as divisors, the actual baud rate differs slightly from the desired (target) baud rate; for this reason, both target and actual baud rates are given, along with the percentage of error.

Table 15-3. Baud Rate Examples

Target Baud Rate (Decimal)	Divisor		Platform Cock/2 Frequency (MHz)	Actual Baud Rate (Decimal)	Percent Error (Decimal)
	Decimal	Hex			
9,600	814	032E	125	9597.666	0.0243
19,200	407	0197	125	19,195.33	0.0243
38,400	203	00CB	125	38,485.22	0.2219
57,600	136	0088	125	57,444.85	0.2694
115,200	68	0044	125	114,889.7	0.2694
230,400	34	0022	125	229,779.4	0.2694

15.4.2.3 Diagram



15.4.2.4 Fields

Field	Function
0-7 UDLB	Divisor least significant byte. This is concatenated with UDMB.

15.4.3 UART receiver buffer register (URBR1 - URBR2)

15.4.3.1 Offset

For a = 1 to 2:

Register	Offset
URBRa	400h + (a × 100h)

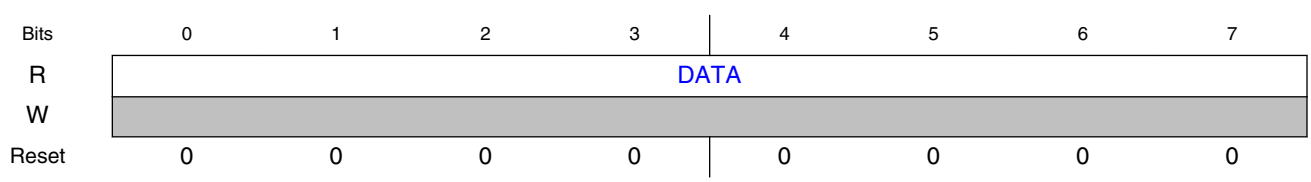
15.4.3.2 Function

This register is accessible when ULCR[DLAB] = 0.

These registers contain the data received from the transmitter on the UART buses. In FIFO mode, when read, they return the first byte received. For FIFO status information, refer to the UDSR[RXRDY] description.

Except for the case when there is an overrun, URBR returns the data in the order it was received from the transmitter. Refer to the ULSR[OE] description, UART line status register (ULSR1 - ULSR2) . Note that these registers have same offset as the UTHRs.

15.4.3.3 Diagram



15.4.3.4 Fields

Field	Function
0-7 DATA	Data received from the transmitter on the UART bus (read only)

15.4.4 UART transmitter holding register (UTHR1 - UTHR2)

15.4.4.1 Offset

For a = 1 to 2:

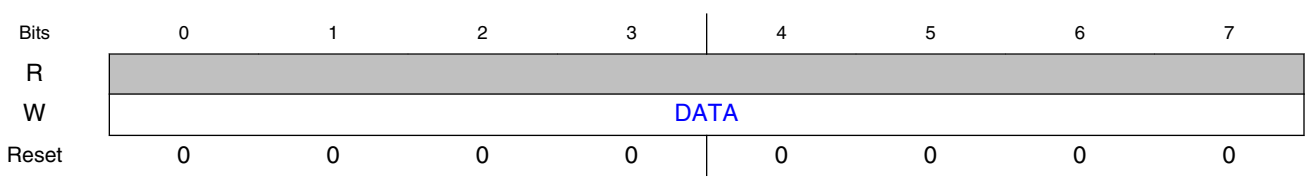
Register	Offset
UTHRa	400h + (a × 100h)

15.4.4.2 Function

This register is accessible when ULCR[DLAB] = 0.

A write to these 8-bit registers causes the UART devices to transfer 5-8 data bits on the UART bus in the format set up in the ULCR (line control register). In FIFO mode, data written to UTHR is placed into the FIFO. The data written to UTHR is the data sent onto the UART bus, and the first byte written to UTHR is the first byte onto the bus. UDSR[TXRDY_B] indicates when the FIFO is full. See UART DMA status register (UDSR1 - UDSR2) for more details.

15.4.4.3 Diagram



15.4.4.4 Fields

Field	Function
0-7 DATA	Data that is written to UTHR (write only)

15.4.5 UART divisor most significant byte register (UDMB1 - UDMB2)

15.4.5.1 Offset

For a = 1 to 2:

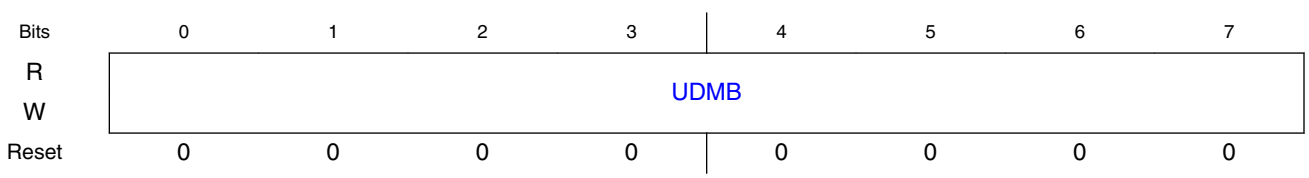
Register	Offset
UDMBa	401h + (a × 100h)

15.4.5.2 Function

This register is accessible when ULCR[DLAB] = 1.

The divisor least significant byte register (UDLB) is concatenated with the divisor most significant byte register (UDMB) to create the divisor used to divide the input clock into the DUART. The output frequency of the baud generator is 16 times the baud rate; therefore the desired baud rate = platform clock frequency/2 ÷ (16 x [UDMB||UDLB]). Equivalently, [UDMB||UDLB:0b0000] = platform clock frequency/2 ÷ desired baud rate. Baud rates that can be generated by specific input clock frequencies are shown in UART divisor least significant byte register (UDLB1 - UDLB2).

15.4.5.3 Diagram



15.4.5.4 Fields

Field	Function
0-7 UDMB	Divisor most significant byte

15.4.6 UART interrupt enable register (UIER1 - UIER2)

15.4.6.1 Offset

For a = 1 to 2:

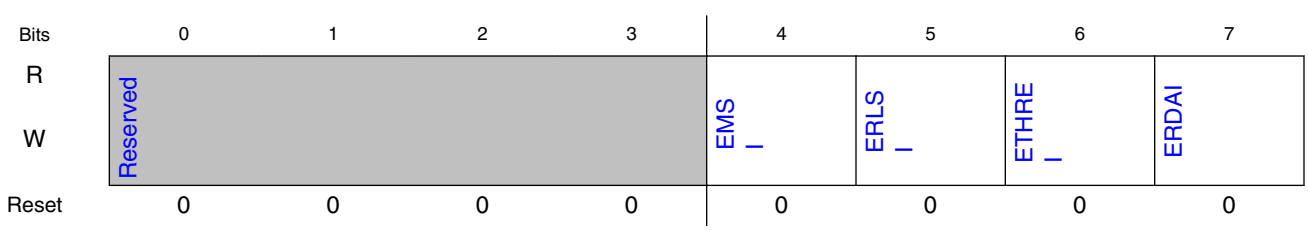
Register	Offset
UIERa	401h + (a × 100h)

15.4.6.2 Function

This register is accessible when ULCR[DLAB] = 0.

The UIER gives the user the ability to mask specific UART interrupts to the programmable interrupt controller (PIC).

15.4.6.3 Diagram



15.4.6.4 Fields

Field	Function
0-3 —	Reserved.
4 EMSI	Enable modem status interrupt. 0b - Mask interrupts caused by UMSR[DCTS] being set 1b - Enable and assert interrupts when the clear-to-send bit in the UART modem status register (UMSR) changes state
5 ERLSI	Enable receiver line status interrupt. 0b - Mask interrupts when ULSR's overrun, parity error, framing error or break interrupt bits are set 1b - Enable and assert interrupts when ULSR's overrun, parity error, framing error or break interrupt bits are set
6 ETHREI	Enable transmitter holding register empty interrupt. 0b - Mask interrupt when ULSR[THRE] is set 1b - Enable and assert interrupts when ULSR[THRE] is set
7 ERDAI	Enable received data available interrupt. 0b - Mask interrupt when new receive data is available or receive data time out has occurred 1b - Enable and assert interrupts when a new data character is received from the external device and/or a time-out interrupt occurs in the FIFO mode

15.4.7 UART alternate function register (UAFR1 - UAFR2)

15.4.7.1 Offset

For a = 1 to 2:

Register	Offset
UAFRa	402h + (a × 100h)

15.4.7.2 Function

This register is accessible when ULCR[DLAB] = 1.

The UAFRs give software the ability to gate off the baud clock and write to each UARTn registers simultaneously with the same write operation.

15.4.7.3 Diagram

Bits	0	1	2	3	4	5	6	7
R W	Reserved						BO	CW
Reset	0	0	0	0	0	0	0	0

15.4.7.4 Fields

Field	Function
0-5 —	Reserved.
6 BO	Baud clock select. 0b - The baud clock is not gated off. 1b - The baud clock is gated off.
7 CW	Concurrent write enable. 0b - Disables writing to each UART*n* 1b - Enables concurrent writes to corresponding UART registers. A write to a register in UART*n* is also a write to the corresponding register in UART*n+1* and vice versa for each DUART where *n* refers to the number of UART controller. The user needs to ensure that the ULCR[DLAB] of each UART is in the same state before executing a concurrent write to register addresses 0x*m*00, 0x*m*01 and 0x*m*02, where *m* is the base address of the corresponding UART.

15.4.8 UART FIFO control register (UFCR1 - UFCR2)

15.4.8.1 Offset

For a = 1 to 2:

Register	Offset
UFCRa	402h + (a × 100h)

15.4.8.2 Function

This register is accessible when ULCR[DLAB] = 0.

The UFCR, a write-only register, is used to enable and clear the receiver and transmitter FIFOs, set a receiver FIFO trigger level to control the received data available interrupt, and select the type of DMA signaling.

When the UFCR bits are written, the FIFO enable bit must also be set or else the UFCR bits are not programmed. When changing from FIFO mode to 16450 mode (non-FIFO mode) and vice versa, data is automatically cleared from the FIFOs.

After all the bytes in the receiver FIFO are cleared, the receiver internal shift register is not cleared. Similarly, the bytes are cleared in the transmitter FIFO, but the transmitter internal shift register is not cleared. Both TFR and RFR are self-clearing bits.

15.4.8.3 Diagram

Bits	0	1	2	3	4	5	6	7
R								
W	RTL		EN64	Reserved	DMS	TFR	RFR	FEN
Reset	0	0	0	0	0	0	0	0

15.4.8.4 Fields

Field	Function
0-1 RTL	Receiver trigger level. A received data available interrupt occurs when UIER[ERDAI] is set and the number of bytes in the receiver FIFO equals the designated interrupt trigger level as follows: 00b - 1 byte, if EN64 1 byte 01b - 4 bytes, if EN64 16 bytes 10b - 8 bytes, if EN64 32 bytes 11b - 14 bytes, if EN64 56 bytes
2 EN64	Enable 64-byte FIFO 0b - Disables the 64-byte FIFOs 1b - Enables the 64-byte FIFOs
3 —	Reserved
4 DMS	DMA mode select. See DMA mode select for more information. 0b - UDSR[RXRDY] and UDSR[TXRDY] bits are in mode 0. 1b - UDSR[RXRDY] and UDSR[TXRDY] bits are in mode 1 if UFCR[FEN] = 1.
5 TFR	Transmitter FIFO reset 0b - No action 1b - Clears all bytes in the transmitter FIFO and resets the FIFO counter/pointer to 0
6	Receiver FIFO reset

Table continues on the next page...

Field	Function
RFR	0b - No action 1b - Clears all bytes in the receiver FIFO and resets the FIFO counter/pointer to 0
7 FEN	FIFO enable 0b - FIFOs are disabled and cleared 1b - Enables the transmitter and receiver FIFOs

15.4.9 UART interrupt ID register (UIIR1 - UIIR2)

15.4.9.1 Offset

For a = 1 to 2:

Register	Offset
UIIRa	402h + (a × 100h)

15.4.9.2 Function

This register is accessible when ULCR[DLAB] = 0.

The UIIRs indicate when an interrupt is pending from the corresponding UART and what type of interrupt is active. They also indicate if the FIFOs are enabled.

The DUART prioritizes interrupts into four levels and records these in the corresponding UIIR. The four levels of interrupt conditions in order of priority are:

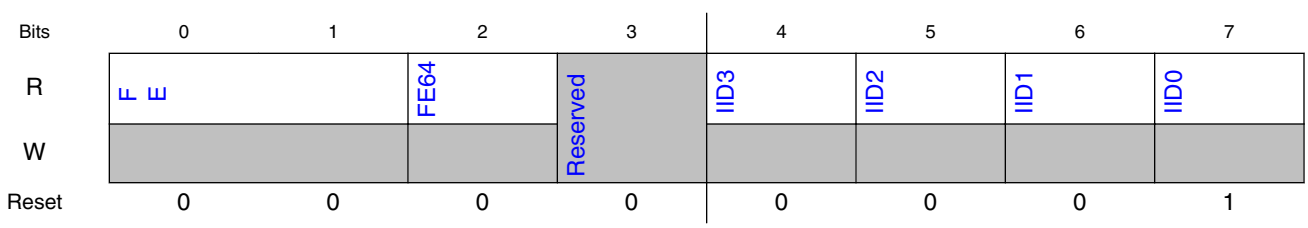
1. Receiver line status
2. Received data ready/character time-out
3. Transmitter holding register empty
4. Modem status

When the UIIR is read, the associated DUART serial channel freezes all interrupts and indicates the highest priority pending interrupt. While this read transaction is occurring, the associated DUART serial channel records new interrupts, but does not change the contents of UIIR until the read access is complete.

Table 15-4. UIIR IID Bits Summary

IID Bits IID[3-0]	Priority Level	Interrupt Type	Interrupt Description	How To Reset Interrupt
0b0001	-	-	-	-
0b0110	Highest	Receiver line status	Overrun error, parity error, framing error, or break interrupt	Read the line status register.
0b0100	Second	Received data available	Receiver data available or trigger level reached in FIFO mode	Read the receiver buffer register or interrupt is automatically reset if the number of bytes in the receiver FIFO drops below the trigger level.
0b1100	Second	Character time-out	No characters have been removed from or input to the receiver FIFO during the last 4 character times and there is at least one character in the receiver FIFO during this time.	Read the receiver buffer register.
0b0010	Third	UTHR empty	Transmitter holding register is empty	Read the UIIR or write to the UTHR.
0b0000	Fourth	Modem status	CTS_B input value changed since last read of UMSR	Read the UMSR.

15.4.9.3 Diagram



15.4.9.4 Fields

Field	Function
0-1 FE	FIFOs enabled. Reflects the setting of UFCR[FEN]

Table continues on the next page...

Field	Function
2 FE64	64-byte FIFOs enabled. Reflects the setting of UFCR[EN64]. 0b - 64-byte FIFOs disabled 1b - 64-byte FIFOs enabled
3 —	Reserved
4 IID3	Interrupt ID bits identify the highest priority interrupt that is pending as indicated in the table above. IID3 is set along with IID2 only when a timeout interrupt is pending for FIFO mode.
5 IID2	Interrupt ID bits identify the highest priority interrupt that is pending as indicated in the table above.
6 IID1	Interrupt ID bits identify the highest priority interrupt that is pending as indicated in the table above.
7 IID0	IID0 indicates when an interrupt is pending. 0b - The UART has an active interrupt ready to be serviced. 1b - No interrupt is pending.

15.4.10 UART line control register (ULCR1 - ULCR2)

15.4.10.1 Offset

For a = 1 to 2:

Register	Offset
ULCRa	403h + (a × 100h)

15.4.10.2 Function

This register is accessible when ULCR[DLAB] = x.

The ULCRs specify the data format for the UART bus and set the divisor latch access bit ULCR[DLAB], which controls the ability to access the divisor latch least and most significant bit registers and the alternate function register.

After initializing the ULCR, the software should not re-write the ULCR when valid transfers on the UART bus are active. The software should not re-write the ULCR until the last STOP bit has been received and there are no new characters being transferred on the bus.

The stick parity bit, ULCR[SP], assigns a set parity value for the parity bit time slot sent on the UART bus. The set value is defined as mark parity (logic 1) or space parity (logic 0). ULCR[PEN] and ULCR[EPS] help determine the set parity value. See the table below for more information. ULCR[NSTB], defines the number of STOP bits to be sent at the end of the data transfer. The receiver only checks the first STOP bit, regardless of the number of STOP bits selected. The word length select bits (1 and 0) define the number of data bits that are transmitted or received as a serial character. The word length does not include START, parity, and STOP bits.

Table 15-5. Parity Selection Using ULCR[PEN], ULCR[SP], and ULCR[EPS]

PEN	SP	EPS	Parity Selected
0	0	0	No parity
0	0	1	No parity
0	1	0	No parity
0	1	1	No parity
1	0	0	Odd parity
1	0	1	Even parity
1	1	0	Mark parity
1	1	1	Space parity

15.4.10.3 Diagram

Bits	0	1	2	3	4	5	6	7
R W	DLAB	SB	SP	EPS	PEN	NSTB	WLS	
Reset	0	0	0	0	0	0	0	0

15.4.10.4 Fields

Field	Function
0 DLAB	Divisor latch access bit. 0b - Access to all registers except UDLB, UAFR, and UDMB 1b - Ability to access divisor latch least and most significant byte registers and alternate function register (UAFR)
1 SB	Set break. 0b - Send normal UTHR data onto the serial output (SOUT) signal 1b - Force logic 0 to be on the SOUT signal. Data in the UTHR is not affected

Table continues on the next page...

Field	Function
2 SP	Stick parity. 0b - Stick parity is disabled. 1b - If PEN = 1 and EPS = 1, space parity is selected. And if PEN = 1 and EPS = 0, mark parity is selected.
3 EPS	Even parity select. See the table above for more information. 0b - If PEN = 1 and SP = 0, odd parity is selected. 1b - If PEN = 1 and SP = 0, even parity is selected.
4 PEN	Parity enable. 0b - No parity generation and checking 1b - Generate parity bit as a transmitter, and check parity as a receiver
5 NSTB	Number of STOP bits. 0b - One STOP bit is generated in the transmitted data. 1b - When a 5-bit data length is selected, 1½ STOP bits are generated. When either a 6-, 7-, or 8-bit word length is selected, two STOP bits are generated.
6-7 WLS	Word length select. Number of bits that comprise the character length. The word length select values are as follows: 00b - 5 bits 01b - 6 bits 10b - 7 bits 11b - 8 bits

15.4.11 UART line status register (ULSR1 - ULSR2)

15.4.11.1 Offset

For a = 1 to 2:

Register	Offset
ULSRa	405h + (a × 100h)

15.4.11.2 Function

This register is accessible when ULCR[DLAB] = x.

The ULSRs are read-only registers that monitor the status of the data transfer on the UART buses. To isolate the status bits from the proper character received through the UART bus, software should read the ULSR and then the URBR.

15.4.11.3 Diagram

Bits	0	1	2	3	4	5	6	7
R	RFE	TEMT	THRE	BI	FE	PE	OE	DR
W								
Reset	0	1	1	0	0	0	0	0

15.4.11.4 Fields

Field	Function
0 RFE	Receiver FIFO error. 0b - This bit is cleared when there are no errors in the receiver FIFO or on a read of the ULSR with no remaining receiver FIFO errors. 1b - Set to one when one of the characters in the receiver FIFO encounters an error (framing, parity, or break interrupt)
1 TEMT	Transmitter empty. 0b - Either or both the UTHR or the internal transmitter shift register has a data character. In FIFO mode, a data character is in the transmitter FIFO or the internal transmitter shift register. 1b - Both the UTHR and the internal transmitter shift register are empty. In FIFO mode, both the transmitter FIFO and the internal transmitter shift register are empty.
2 THRE	Transmitter holding register empty. 0b - The UTHR is not empty. 1b - A data character has transferred from the UTHR into the internal transmitter shift register. In FIFO mode, the transmitter FIFO contains no data character.
3 BI	Break interrupt. **NOTE:** For a single break signal, BI and DR are set multiple times, approximately once every character period. The BI and DR bits continue to be set each character period after they are cleared. This continues for the entire duration of the break signal. To accommodate this behavior, read URBR, which returns zeros and clears DR. Then delay one character period and read URBR again. Note that at the end of the break signal, a random character may be falsely detected and received in the URBR, with ULSR[DR] being set. 0b - This bit is cleared when the ULSR is read or when a valid data transfer is detected (that is, STOP bit is received). 1b - Received data of logic 0 for more than START bit + Data bits + Parity bit + one STOP bits length of time. A new character is not loaded until SIN returns to the mark state (logic 1) and a valid START is detected. In FIFO mode, a zero character is encountered in the FIFO (the zero character is at the top of the FIFO). In FIFO mode, only one zero character is stored.
4 FE	Framing error. 0b - This bit is cleared when ULSR is read or when a new character is loaded into the URBR from the receiver shift register. 1b - Invalid STOP bit for receive data (only the first STOP bit is checked). In FIFO mode, this bit is set when the character that detected a framing error is encountered in the FIFO (that is the character at the top of the FIFO). An attempt to resynchronize occurs after a framing error. The UART assumes that the framing error (due to a logic 0 being read when a logic 1 (STOP) was

Table continues on the next page...

Field	Function
	expected) was due to a STOP bit overlapping with the next START bit, so it assumes this logic 0 sample is a true START bit and then receives the following new data.
5 PE	Parity error. 0b - This bit is cleared when ULSR is read or when a new character is loaded into the URBR. 1b - Unexpected parity value encountered when receiving data. In FIFO mode, the character with the error is at the top of the FIFO .
6 OE	Overrun error. 0b - This bit is cleared when ULSR is read. 1b - Before the URBR is read, the URBR was overwritten with a new character. The old character is loss. In FIFO mode, the receiver FIFO is full (regardless of the receiver FIFO trigger level setting) and a new character has been received into the internal receiver shift register. The old character was overwritten by the new character. Data in the receiver FIFO was not overwritten.
7 DR	Data ready. 0b - This bit is cleared when URBR is read or when all of the data in the receiver FIFO is read. 1b - A character has been received in the URBR or the receiver FIFO.

15.4.12 UART scratch register (USCR1 - USCR2)

15.4.12.1 Offset

For a = 1 to 2:

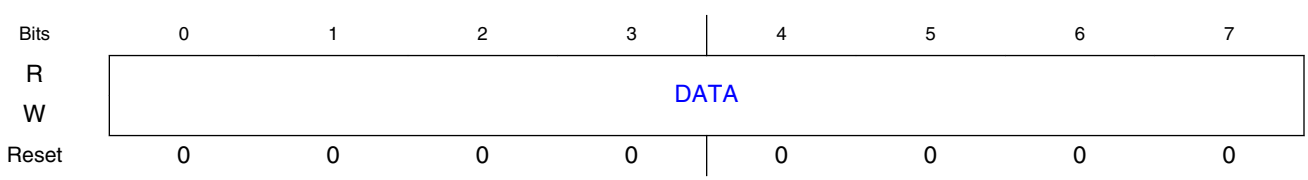
Register	Offset
USCRa	407h + (a × 100h)

15.4.12.2 Function

This register is accessible when ULCR[DLAB] = x.

The USCR registers are for debugging software or the DUART hardware. The USCRs do not affect the operation of the DUART.

15.4.12.3 Diagram



15.4.12.4 Fields

Field	Function
0-7 DATA	Data

15.4.13 UART DMA status register (UDSR1 - UDSR2)

15.4.13.1 Offset

For a = 1 to 2:

Register	Offset
UDSRa	410h + (a × 100h)

15.4.13.2 Function

This register is accessible when ULCR[DLAB] = x.

The DMA status registers (UDSRs) are read-only registers that return transmitter and receiver FIFO status. UDSRs also provide the ability to assist DMA data operations to and from the FIFOs.

Table 15-6. UDSR[TXRDY] Set Conditions

DMS	FEN	DMA Mode	Meaning
0	0	0	TXRDY is set after the first character is loaded into the transmitter FIFO or UTHR.
0	1	0	
1	0	0	
1	1	1	TXRDY is set when the transmitter FIFO is full.

Table 15-7. UDSR[TXRDY] Cleared Conditions

DMS	FEN	DMA Mode	Meaning
0	0	0	TXRDY is cleared when there are no characters in the transmitter FIFO or UTHR.
0	1	0	
1	0	0	
1	1	1	TXRDY is cleared when there are no characters in the transmitter FIFO or UTHR. TXRDY remains clear when the transmitter FIFO is not yet full.

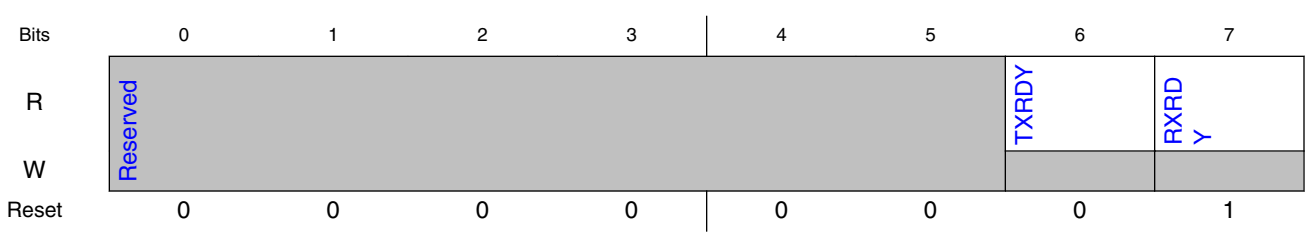
Table 15-8. UDSR[RXRDY] Set Conditions

DMS	FEN	DMA Mode	Meaning
0	0	0	RXRDY is set when there are no characters in the receiver FIFO or URBR.
0	1	0	
1	0	0	
1	1	1	RXRDY is set when the trigger level has not been reached and there has been no time out.

Table 15-9. UDSR[RXRDY] Cleared Conditions

DMS	FEN	DMA Mode	Meaning
0	0	0	RXRDY is cleared when there is at least one character in the receiver FIFO or URBR.
0	1	0	
1	0	0	
1	1	1	RXRDY is cleared when the trigger level or a time-out has been reached. RXRDY remains cleared until the receiver FIFO is empty.

15.4.13.3 Diagram



15.4.13.4 Fields

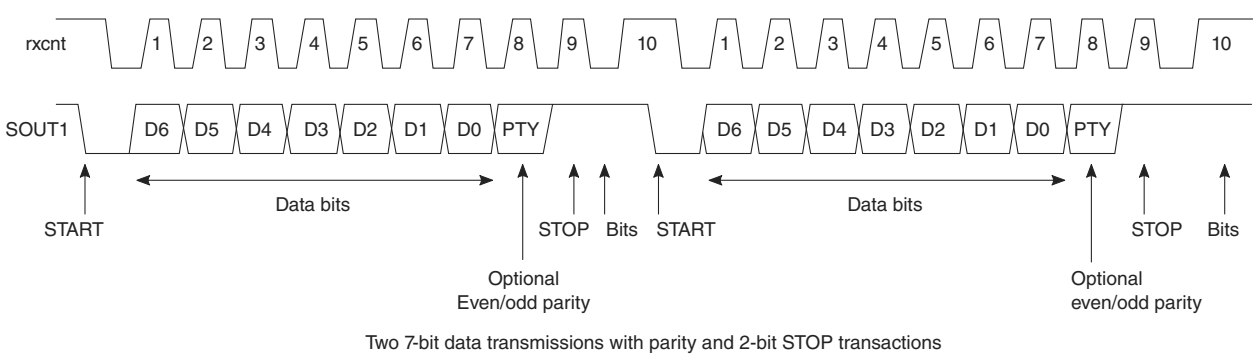
Field	Function
0-5 —	Reserved
6 TXRDY	Transmitter ready. This read-only bit reflects the status of the transmitter FIFO or the UTHR. The status depends on the DMA mode selected, which is determined by the DMS and FEN bits in the UFCR. 0b - This bit is cleared, as shown in Table 15-7 . 1b - This bit is set, as shown in Table 15-6 .
7 RXRDY	Receiver ready. This read-only bit reflects the status of the receiver FIFO or URBR. The status depends on the DMA mode selected, which is determined by the DMS and FEN bits in the UFCR. 0b - This bit is cleared, as shown in Table 15-9 . 1b - This bit is set, as shown in Table 15-8 .

15.5 Functional description

The following sections provide the function of the DUART controller.

15.5.1 Serial interface

The UART bus is a serial, full-duplex, point-to-point bus as shown in the following figure. Therefore, only two devices are attached to the same signals and there is no need for address or arbitration bus cycles.



Two 7-bit data transmissions with parity and 2-bit STOP transactions

Figure 15-2. UART bus interface transaction protocol example

A standard UART bus transfer is composed of either three or four parts:

- START bit
- Data transfer bits (least-significant bit is first data bit on the bus)

- Parity bit (optional)
- STOP bits

An internal logic sample signal, *rxcnt*, uses the frequency of the baud-rate generator to drive the bits on SOUT.

The following sections describe the four components of the serial interface, the baud-rate generator, local loopback mode, different errors, and FIFO mode.

15.5.1.1 START bit

A write to the transmitter holding register (UTHR) generates a START bit on the SOUT signal.

Figure 15-2 shows that the START bit is defined as a logic 0. The START bit denotes the beginning of a new data transfer which is limited to the bit length programmed in the UART line control register (ULCR). When the bus is idle, SOUT is high.

15.5.1.2 Data transfer

Each data transfer contains 5-8 bits of data.

The ULCR data bit length for the transmitter and receiver UART devices must agree before a transfer begins; otherwise, a parity or framing error may occur. A transfer begins when UTHR is written. At that time a START bit is generated followed by 5-8 of the data bits previously written to the UTHR. The data bits are driven from the least significant to the most significant bits. After the parity and STOP bits, a new data transfer can begin if new data is written to the UTHR.

15.5.1.3 Parity bit

The user has the option of using even, odd, no parity, or stick parity.

See UART line control register (ULCR1 - ULCR2). Both the receiver and transmitter parity definition must agree before attempting to transfer data. When receiving data, a parity error can occur if an unexpected parity value is detected. See UART line status register (ULSR1 - ULSR2).

15.5.1.4 STOP bit

The transmitter device ends the write transfer by generating a STOP bit.

The STOP bit is always high. The length of the STOP bit(s) can be programmed in the ULCR. Both the receiver and transmitter STOP bit length must agree before attempting to transfer data. A framing error can occur if an invalid STOP bit is detected.

15.5.2 Baud-rate generator logic

Each UART contains an independent programmable baud-rate generator, that is capable of taking the platform clock frequency/2as input and dividing the input by any divisor from 1 to 2^{16} - 1.

The baud rate is defined as the number of bits per second that can be sent over the UART bus. The formula for calculating baud rate is as follows:

Baud rate = (1/16) x ((platform clock/2) frequency ÷ divisor value)

Therefore, the output frequency of the baud-rate generator is 16 times the baud rate.

The divisor value is determined by the following two 8-bit registers to form a 16-bit binary number:

- UART divisor most significant byte register (UDMB)
- UART divisor least significant byte register (UDLB)

Upon loading either of the divisor latches, a 16-bit baud-rate counter is loaded.

The divisor latches must be loaded during initialization to ensure proper operation of the baud-rate generator. Both UART devices on the same bus must be programmed for the same baud-rate before starting a transfer.

The baud clock can be passed to the performance monitor by enabling the UAFR[BO] bit. This can be used to determine baud rate errors.

15.5.3 Local loopback mode

Local loopback mode is provided for diagnostic testing.

The data written to UTHR can be read from the receiver buffer register (URBR) of the same UART. In this mode, the modem control register UMCR[RTS] is internally tied to the modem status register UMSR[CTS]. The transmitter SOUT is set to a logic 1 and the receiver SIN is disconnected. The output of the transmitter shift register is looped back into the receiver shift register input. The CTS_B (input signal) is disconnected, RTS_B is

internally connected to CTS_B, and the RTS_B (output signal) becomes inactive. In this diagnostic mode, data that is transmitted is immediately received. In local loopback mode, the transmit and receive data paths of the DUART can be verified.

NOTE

In local loopback mode, the transmit/receive interrupts are fully operational and can be controlled by the interrupt enable register (UIER).

15.5.4 Errors

The following sections describe framing, parity, and overrun errors which may occur while data is transferred on the UART bus.

Each of the error bits are usually cleared, as described below, when the line status register (ULSR) is read.

15.5.4.1 Framing error

When an invalid STOP bit is detected, a framing error occurs and ULSR[FE] is set.

Note that only the first STOP bit is checked. In FIFO mode, ULSR[FE] is set when the character at the top of the FIFO detects a framing error. An attempt to re-synchronize occurs after a framing error. The UART assumes that the framing error (due to a logic 0 being read when a logic 1 (STOP) was expected) was due to a STOP bit overlapping with the next START bit. ULSR[FE] is cleared when ULSR is read or when a new character is loaded into the URBR from the receiver shift register.

15.5.4.2 Parity error

A parity error occurs, and ULSR[PE] is set, when unexpected parity values are encountered while receiving data.

In FIFO mode, ULSR[PE] is set when the character with the error is at the top of the FIFO. ULSR[PE] is cleared when ULSR is read or when a new character is loaded into the URBR.

15.5.4.3 Overrun error

When a new (overwriting character) STOP bit is detected and the old character is lost, an overrun error occurs and ULSR[OE] is set.

In FIFO mode, ULSR[OE] is set after the receiver FIFO is full (despite the receiver FIFO trigger level setting) and a new character has been received into the internal receiver shift register. Data in the FIFO is not overwritten; only the shift register data is overwritten. Therefore, the interrupt occurs immediately. ULSR[OE] is cleared when ULSR is read.

15.5.5 FIFO mode

The UARTs use an alternate mode (FIFO mode) to relieve the processor core from excessive software overhead.

The FIFO control register (UFCR) is used to enable and clear the receiver and transmitter FIFOs and set the FIFO receiver trigger level UFCR[RTL] to control the received data available interrupt UIER[ERDAI].

The UFCR also selects the type of DMA signaling. The UDSR[RXRDY] indicates the status of the receiver FIFO. The DMA status registers (UDSR[TXRDY]) indicate when the transmitter FIFO is full. When in FIFO mode, data written to UTHR is placed into the transmitter FIFO. The first byte written to UTHR is the first byte onto the UART bus.

15.5.5.1 FIFO interrupts

In FIFO mode, the UIER[ERDAI] is set when a time-out interrupt occurs. When a receive data time-out occurs there is a maskable interrupt condition (through UIER[ERDAI]). See UART interrupt enable register (UIER1 - UIER2) for more details on interrupt enables.

The interrupt ID register (UIIR) indicates if the FIFOs are enabled. The interrupt ID3 bit UIIR[IID3] is only set for FIFO mode interrupts. The character time-out interrupt occurs when no characters have been removed from or input to the receiver FIFO during the last four character times and there is at least one character in the receiver FIFO during this time. The character time-out interrupt (controlled by UIIR[IID*n*]) is cleared when the URBR is read. See UART interrupt ID register (UIIR1 - UIIR2) for more information.

The UIIR[FE] bit indicates if FIFO mode is enabled.

15.5.5.2 Interrupt control logic

An interrupt is active when DUART interrupt ID register bit 7 (UIIR[IID0]), is cleared.

The interrupt enable register (UIER) is used to mask specific interrupt types. See UART interrupt enable register (UIER1 - UIER2) for more details.

When the interrupts are disabled in UIER, polling software cannot use UIIR[IID0] to determine whether the UART is ready for service. The software must monitor the appropriate bits in the line status (ULSR) and/or the modem status registers (UMSR). UIIR[IID0] can be used for polling if the interrupts are enabled in UIER.

15.6 Initialization/Application information

The following requirement must be met for DUART accesses:

- All DUART registers are 1 byte wide. Reads and writes to these registers must be byte-wide operations.

A system reset puts the DUART registers to a default state. Before the interface can transfer serial data, the following initialization steps are recommended:

1. Update the programmable interrupt controller (PIC) DUART channel interrupt vector source registers.
2. Set data attributes and control bits in the ULCR, UFCR, UAFR, UMCR, UDLB, and UDMB.
3. Set the data attributes and control bits of the external modem or peripheral device.
4. Set the interrupt enable register (UIER).
5. To start a write transfer, write to the UTHR.
6. Poll UIIR, if the interrupts generated by the DUART are masked.

Chapter 16
Direct Memory Access Multiplexer (DMAMUX)

16.1 The DMAMUX module as implemented on the chip

This section provides details about how the DMAMUX module is implemented on the chip.

16.1.1 LS1012A DMAMUX module integration

The following table describes the DMAMUX module integration into this chip:

Table 16-1. DMAMUX module integration

Module	Base address
DMAMUX1	2C1_0000
DMAMUX2	2C2_0000

The remainder of this chapter refers to a single DMAMUX module. Notes are included to indicate variations for multiple instantiations.

16.1.2 LS1012A DMAMUX module special consideration

16.1.2.1 eDMA and DMAMUX

The device contains one eDMA and two DMAMUX modules.

The eDMA module implements the following parameter settings in the chip:

Table 16-2. LS1012A eDMA parameter settings

eDMA parameters	LS1012A parameter value
Stop mode support	Refers to LPM20 low power mode of the chip
Debug mode support	No

16.1.2.1.1 eDMA and DMAMUX channel assignment

The table below provides the channel assignments of eDMA and DMAMUX:

Table 16-3. eDMA and DMAMUX channel assignment

eDMA channel assignment	Parameter value
15-0	DMAMUX1
31-16	DMAMUX2
63-32	Reserved

16.1.2.1.2 DMAMUX settings

The table below provides the DMAMUX settings used in the chip:

Table 16-4. DMAMUX settings

DMAMUX settings	DMAMUX1	DMAMUX2
NUMBER_OF_DMA_CHANNELS	16	16
NUMBER_OF_PERIPHERAL_SLOTS	62	62

16.1.2.1.3 DMAMUX request sources

This device includes a DMA request multiplexer that allows up to 64 DMA request signals to be mapped to any of the 16 DMA channels.

As shown in the figure below, request from MUX1 can be mapped to any of the first 16 DMA channels and from MUX2, to any of the lower 16 DMA channels.

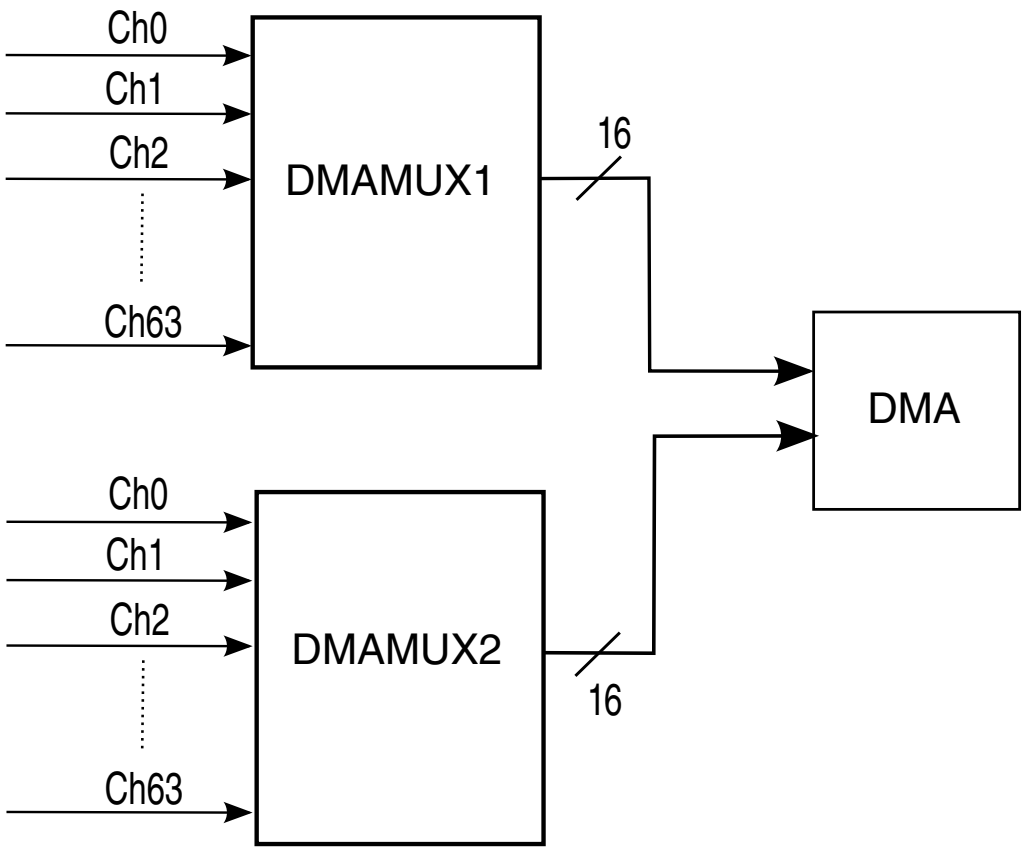


Figure 16-1. DMA Request MUX/DMA Structure

The table below provides the source mapping for DMAMUX1 and DMAMUX2:

Table 16-5. DMAMUX request sources

DMAMUX source #	DMAMUX1	DMAMUX2
1	DMA transfer request from EPU[1]	Reserved
2	Reserved	Reserved
3	Reserved	Reserved
4	Reserved	Reserved
5	Reserved	Reserved
6	Reserved	Reserved
7	Reserved	Reserved
8	Reserved	Reserved
9	Reserved	Reserved
10	Reserved	Reserved
11	Reserved	Reserved
12	Reserved	Reserved

Table continues on the next page...

Table 16-5. DMAMUX request sources (continued)

DMAMUX source #	DMAMUX1	DMAMUX2
13	Reserved	Reserved
14	Reserved	Reserved
15	Reserved	Reserved
16	Reserved	Reserved
17	Reserved	Reserved
18	Reserved	Reserved
19	Reserved	QuadSPI RFDF
20	Reserved	Reserved
21	Reserved	Reserved
22	Reserved	Reserved
23	Reserved	Reserved
24	Reserved	Reserved
25	Reserved	Reserved
26	Reserved	Reserved
27	Reserved	Reserved
28	Reserved	Reserved
29	Reserved	Reserved
30	Reserved	Reserved
31	Reserved	Reserved
32	Reserved	Reserved
33	Reserved	Reserved
34	Reserved	SAI5 Rx
35	FlexTimer2[0]	SAI5 Tx
36	FlexTimer2[1]	IIC2 Rx
37	FlexTimer2[2]	IIC2 Tx
38	FlexTimer2[3]	IIC1 Rx
39	Reserved	IIC1 Tx
40	Reserved	SAI4 Rx
41	Reserved	SAI4 Tx
42	Reserved	SAI3 Rx
43	FlexTimer1[0]	SAI3 Tx
44	FlexTimer1[1]	SAI2 Rx
45	FlexTimer1[2]	SAI2 Tx
46	FlexTimer1[3]	SAI1 Rx
47	Reserved	SAI1 Tx
48	Reserved	Reserved
49	Reserved	Reserved
50	Reserved	Reserved
51	Reserved	Reserved

Table continues on the next page...

Table 16-5. DMAMUX request sources (continued)

DMAMUX source #	DMAMUX1	DMAMUX2
52	Reserved	Reserved
53	Reserved	Reserved
54	Reserved	Reserved
55	Reserved	Reserved
56	Reserved	Reserved
57	Reserved	Reserved
58	Reserved	Reserved
59	Reserved	Reserved
60	SPI1 RFDF	Reserved
61	SPI1 CMD	Reserved
62	SPI1 TF	Reserved

1. Trigger requests from EPU are further qualified to be the bits of DMACR1.

16.2 Introduction

16.2.1 Overview

The Direct Memory Access Multiplexer (DMAMUX) routes DMA sources, called slots, to any of the 32 DMA channels. See the chip-specific information to know the detailed source numbers. This process is illustrated in the following figure.

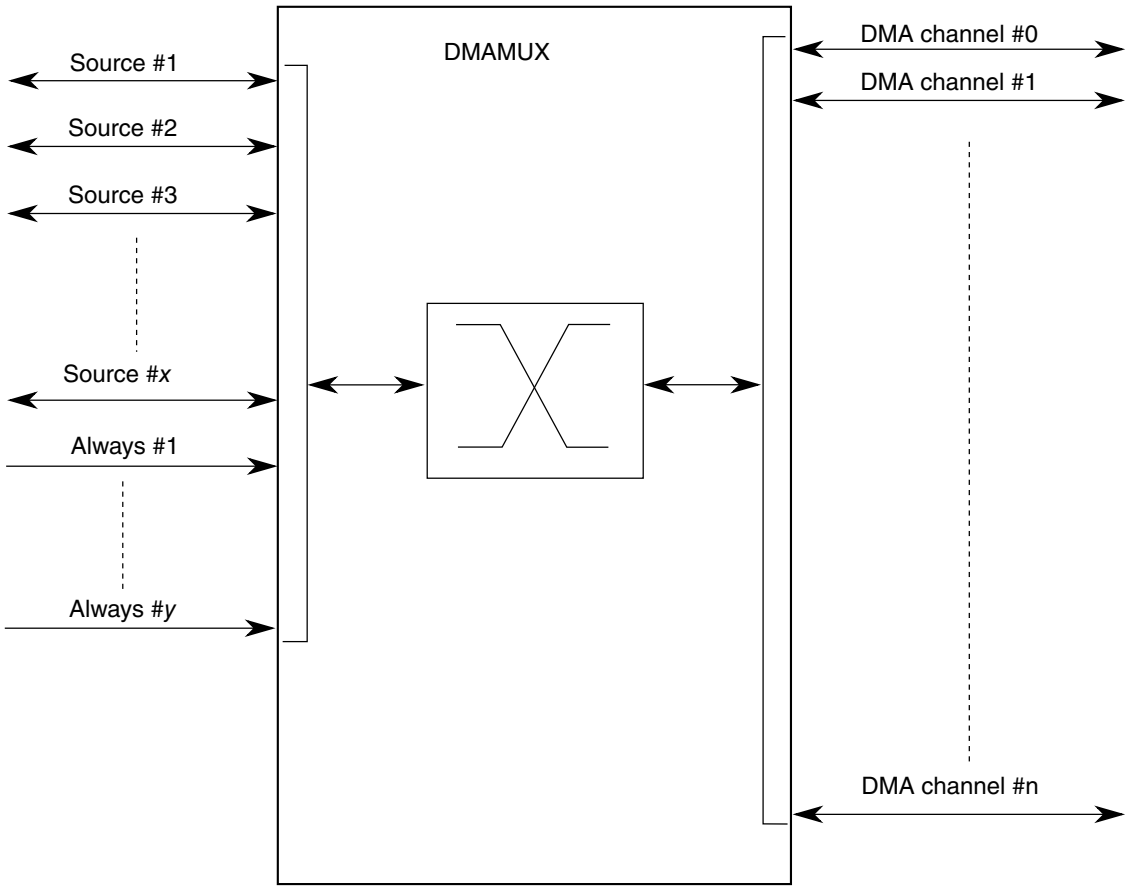


Figure 16-2. DMAMUX block diagram

16.2.2 Features

The DMAMUX module provides these features:

- Up to 62 peripheral slots and up to 1 always-on slots can be routed to 32 channels.
- 32 independently selectable DMA channel routers.
- Each channel router can be assigned to one of the possible peripheral DMA slots or to one of the always-on slots.

16.2.3 Modes of operation

The following operating modes are available:

- Disabled mode

In this mode, the DMA channel is disabled. Because disabling and enabling of DMA channels is done primarily via the DMA configuration registers, this mode is used mainly as the reset state for a DMA channel in the DMA channel MUX. It may also be used to temporarily suspend a DMA channel while reconfiguration of the system takes place.

- Normal mode

  In this mode, a DMA source is routed directly to the specified DMA channel. The operation of the DMAMUX in this mode is completely transparent to the system.

16.3 External signal description

The DMAMUX has no external pins.

16.4 Memory map/register definition

This section provides a detailed description of all memory-mapped registers in the DMAMUX.

16.4.1 DMAMUX register descriptions

16.4.1.1 DMAMUX Memory map

DMAMUX1 base address: 2C1_0000h

DMAMUX2 base address: 2C2_0000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Channel Configuration register (CHCFG3)	8	RW	00h
1h	Channel Configuration register (CHCFG2)	8	RW	00h
2h	Channel Configuration register (CHCFG1)	8	RW	00h
3h	Channel Configuration register (CHCFG0)	8	RW	00h
4h	Channel Configuration register (CHCFG7)	8	RW	00h
5h	Channel Configuration register (CHCFG6)	8	RW	00h

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
6h	Channel Configuration register (CHCFG5)	8	RW	00h
7h	Channel Configuration register (CHCFG4)	8	RW	00h
8h	Channel Configuration register (CHCFG11)	8	RW	00h
9h	Channel Configuration register (CHCFG10)	8	RW	00h
Ah	Channel Configuration register (CHCFG9)	8	RW	00h
Bh	Channel Configuration register (CHCFG8)	8	RW	00h
Ch	Channel Configuration register (CHCFG15)	8	RW	00h
Dh	Channel Configuration register (CHCFG14)	8	RW	00h
Eh	Channel Configuration register (CHCFG13)	8	RW	00h
Fh	Channel Configuration register (CHCFG12)	8	RW	00h
10h	Channel Configuration register (CHCFG19)	8	RW	00h
11h	Channel Configuration register (CHCFG18)	8	RW	00h
12h	Channel Configuration register (CHCFG17)	8	RW	00h
13h	Channel Configuration register (CHCFG16)	8	RW	00h
14h	Channel Configuration register (CHCFG23)	8	RW	00h
15h	Channel Configuration register (CHCFG22)	8	RW	00h
16h	Channel Configuration register (CHCFG21)	8	RW	00h
17h	Channel Configuration register (CHCFG20)	8	RW	00h
18h	Channel Configuration register (CHCFG27)	8	RW	00h
19h	Channel Configuration register (CHCFG26)	8	RW	00h
1Ah	Channel Configuration register (CHCFG25)	8	RW	00h
1Bh	Channel Configuration register (CHCFG24)	8	RW	00h
1Ch	Channel Configuration register (CHCFG31)	8	RW	00h
1Dh	Channel Configuration register (CHCFG30)	8	RW	00h
1Eh	Channel Configuration register (CHCFG29)	8	RW	00h
1Fh	Channel Configuration register (CHCFG28)	8	RW	00h

16.4.1.2 Channel Configuration register (CHCFG0 - CHCFG31)

16.4.1.2.1 Offset

Register	Offset
CHCFG3	0h
CHCFG2	1h
CHCFG1	2h
CHCFG0	3h
CHCFG7	4h

Table continues on the next page...

Register	Offset
CHCFG6	5h
CHCFG5	6h
CHCFG4	7h
CHCFG11	8h
CHCFG10	9h
CHCFG9	Ah
CHCFG8	Bh
CHCFG15	Ch
CHCFG14	Dh
CHCFG13	Eh
CHCFG12	Fh
CHCFG19	10h
CHCFG18	11h
CHCFG17	12h
CHCFG16	13h
CHCFG23	14h
CHCFG22	15h
CHCFG21	16h
CHCFG20	17h
CHCFG27	18h
CHCFG26	19h
CHCFG25	1Ah
CHCFG24	1Bh
CHCFG31	1Ch
CHCFG30	1Dh
CHCFG29	1Eh
CHCFG28	1Fh

16.4.1.2.2 Function

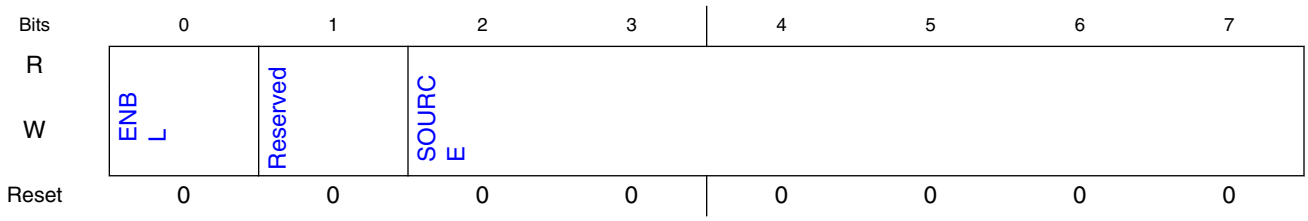
Each of the DMA channels can be independently enabled/disabled and associated with one of the DMA slots (peripheral slots or always-on slots) in the system.

NOTE

Setting multiple CHCFG registers with the same source value will result in unpredictable behavior. This is true, even if a channel is disabled (ENBL==0).

Before changing the source settings, a DMA channel must be disabled via CHCFGn[ENBL].

16.4.1.2.3 Diagram



16.4.1.2.4 Fields

Field	Function
0 ENBL	DMA Channel Enable Enables the DMA channel. 0b - DMA channel is disabled. This mode is primarily used during configuration of the DMAMux. The DMA has separate channel enables/disables, which should be used to disable or reconfigure a DMA channel. 1b - DMA channel is enabled
1 —	Reserved This read/write field does not affect the functionality of the device and should not be used.
2-7 SOURCE	DMA Channel Source (Slot) Specifies which DMA source, if any, is routed to a particular DMA channel. See the chip-specific DMAMUX information for details about the peripherals and their slot numbers.

16.5 Functional description

The primary purpose of the DMAMUX is to provide flexibility in the system's use of the available DMA channels.

As such, configuration of the DMAMUX is intended to be a static procedure done during execution of the system boot code. However, if the procedure outlined in Enabling and configuring sources is followed, the configuration of the DMAMUX may be changed during the normal operation of the system.

The DMAMUX channels implement only the normal routing functionality.

16.5.1 Always-enabled DMA sources

In addition to the peripherals that can be used as DMA sources, there are 1 additional DMA sources that are always enabled. Unlike the peripheral DMA sources, where the peripheral controls the flow of data during DMA transfers, the sources that are always enabled provide no such "throttling" of the data transfers. These sources are most useful in the following cases:

- Performing DMA transfers to/from GPIO—Moving data from/to one or more GPIO pins.
- Performing DMA transfers from memory to memory—Moving data from memory to memory, typically as fast as possible, sometimes with software activation.
- Performing DMA transfers from memory to the external bus, or vice-versa—Similar to memory to memory transfers, this is typically done as quickly as possible.
- Any DMA transfer that requires software activation—Any DMA transfer that should be explicitly started by software.

In cases where software should initiate the start of a DMA transfer, an always-enabled DMA source can be used to provide maximum flexibility. When activating a DMA channel via software, subsequent executions of the minor loop require that a new start event be sent. This can either be a new software activation, or a transfer request from the DMA channel MUX. The options for doing this are:

- Transfer all data in a single minor loop.

  By configuring the DMA to transfer all of the data in a single minor loop (that is, major loop counter = 1), no reactivation of the channel is necessary. The disadvantage to this option is the reduced granularity in determining the load that the DMA transfer will impose on the system. For this option, the DMA channel must be disabled in the DMA channel MUX.

- Use explicit software reactivation.

  In this option, the DMA is configured to transfer the data using both minor and major loops, but the processor is required to reactivate the channel by writing to the DMA registers *after every minor loop*. For this option, the DMA channel must be disabled in the DMA channel MUX.

- Use an always-enabled DMA source.

  In this option, the DMA is configured to transfer the data using both minor and major loops, and the DMA channel MUX does the channel reactivation. For this option, the DMA channel should be enabled and pointing to an "always enabled" source.

16.6 Initialization/application information

This section provides instructions for initializing the DMA channel MUX.

16.6.1 Reset

The reset state of each individual bit is shown in Memory map/register definition. In summary, after reset, all channels are disabled and must be explicitly enabled before use.

16.6.2 Enabling and configuring sources

To enable a source, the following steps can be used:

1. Determine with which DMA channel the source will be associated.
2. Clear the CHCFG[ENBL] field of the DMA channel.
3. Ensure that the DMA channel is properly configured in the DMA. The DMA channel may be enabled at this point.
4. Select the source to be routed to the DMA channel. Write to the corresponding CHCFG register, ensuring that CHCFG[ENBL] is set.

To configure source #5 transmit for use with DMA channel 1, as an example, the following steps can be used:

1. Write 0x00 to CHCFG1.
2. Configure channel 1 in the DMA, including enabling the channel.
3. Write 0x85 to CHCFG1.

The following code example illustrates steps 1 and 3 above:

```
In File registers.h:

#define DMAMUX_BASE_ADDR     0x40021000/* Example only ! */

/* Following example assumes char is 8-bits */

volatile unsigned char *CHCFG0 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0003);

volatile unsigned char *CHCFG1 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0002);

volatile unsigned char *CHCFG2 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0001);

volatile unsigned char *CHCFG3 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0000);

volatile unsigned char *CHCFG4 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0007);

volatile unsigned char *CHCFG5 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0006);

volatile unsigned char *CHCFG6 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0005);
```

```
volatile unsigned char *CHCFG7 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0004);
volatile unsigned char *CHCFG8 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000B);
volatile unsigned char *CHCFG9 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000A);
volatile unsigned char *CHCFG10= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0009);
volatile unsigned char *CHCFG11= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0008);
volatile unsigned char *CHCFG12= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000F);
volatile unsigned char *CHCFG13= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000E);
volatile unsigned char *CHCFG14= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000D);
volatile unsigned char *CHCFG15= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000C);


In File main.c:
#include "registers.h"
:
:
*CHCFG1 = 0x00;
*CHCFG1 = 0x85;
```

To disable a source:

A particular DMA source may be disabled by not writing the corresponding source value into any of the CHCFG registers. Additionally, some module-specific configuration may be necessary. See the appropriate section for more details.

To switch the source of a DMA channel:

1. Disable the DMA channel in the DMA and reconfigure the channel for the new source.
2. Clear the CHCFG[ENBL] bit of the DMA channel.
3. Select the source to be routed to the DMA channel. Write to the corresponding CHCFG register, ensuring that the CHCFG[ENBL] field is set.

To switch DMA channel 8 from source #5 transmit to source #7 transmit:

1. In the DMA configuration registers, disable DMA channel 8 and reconfigure it to handle the transfers to peripheral slot 7. This example assumes channel 8 doesn't have triggering capability.
2. Write 0x00 to CHCFG8.
3. Write 0x87 to CHCFG8.

The following code example illustrates steps 2 and 3 above:

```
In File registers.h:
```

Initialization/application information

```
#define DMAMUX_BASE_ADDR     0x40021000/* Example only ! */

/* Following example assumes char is 8-bits */

volatile unsigned char *CHCFG0 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0003);

volatile unsigned char *CHCFG1 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0002);

volatile unsigned char *CHCFG2 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0001);

volatile unsigned char *CHCFG3 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0000);

volatile unsigned char *CHCFG4 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0007);

volatile unsigned char *CHCFG5 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0006);

volatile unsigned char *CHCFG6 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0005);

volatile unsigned char *CHCFG7 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0004);

volatile unsigned char *CHCFG8 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000B);

volatile unsigned char *CHCFG9 = (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000A);

volatile unsigned char *CHCFG10= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0009);

volatile unsigned char *CHCFG11= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x0008);

volatile unsigned char *CHCFG12= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000F);

volatile unsigned char *CHCFG13= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000E);

volatile unsigned char *CHCFG14= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000D);

volatile unsigned char *CHCFG15= (volatile unsigned char *) (DMAMUX_BASE_ADDR+0x000C);


In File main.c:

#include "registers.h"

:

:

*CHCFG8 = 0x00;

*CHCFG8 = 0x87;
```

Chapter 17
Enhanced Direct Memory Access Controller (eDMA)

17.1 The eDMA module as implemented on the chip

This section provides details about how the eDMA module is implemented on the chip.

17.1.1 LS1012A eDMA module special consideration

The eDMA module implements the following parameter settings in the chip:

Table 17-1. LS1012A eDMA parameter settings

eDMA parameters	LS1012A parameter value
Stop mode support	Yes. Refers to the LPM20 low power mode of the chip.
Debug mode support	No

17.2 Introduction

The enhanced direct memory access (eDMA) controller is a second-generation module capable of performing complex data transfers with minimal intervention from a host processor. The hardware microarchitecture includes:

- A DMA engine that performs:
  - Source address and destination address calculations
  - Data-movement operations
- Local memory containing transfer control descriptors for each of the 32 channels

17.2.1 eDMA system block diagram

Figure 17-1 illustrates the components of the eDMA system, including the eDMA module ("engine").

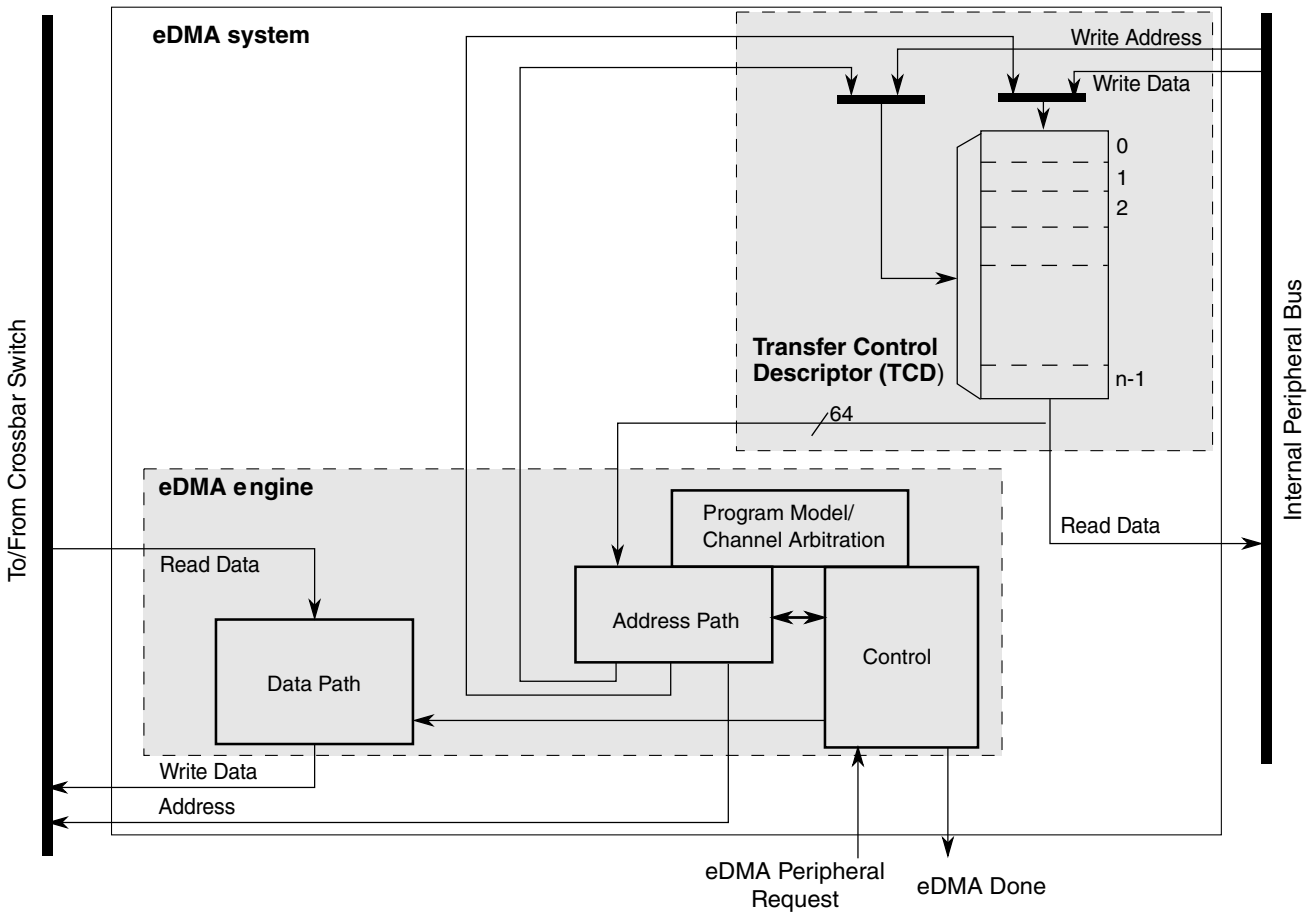


Figure 17-1. eDMA system block diagram

17.2.2 Block parts

The eDMA module is partitioned into two major modules: the eDMA engine and the transfer-control descriptor local memory.

The eDMA engine is further partitioned into four submodules:

Table 17-2. eDMA engine submodules

Submodule	Function
Address path	This block implements registered versions of two channel transfer control descriptors, channel x and channel y, and manages all master bus-address calculations. All the channels provide the same functionality. This structure allows data transfers associated with one channel to be preempted after the completion of a read/write sequence if a higher priority channel activation is asserted while the first channel is active. After a channel is activated, it runs until the minor loop is

Table continues on the next page...

Table 17-2. eDMA engine submodules (continued)

Submodule	Function
	completed, unless preempted by a higher priority channel. This provides a mechanism (enabled by DCHPRI*n*[ECP]) where a large data move operation can be preempted to minimize the time another channel is blocked from execution. When any channel is selected to execute, the contents of its TCD are read from local memory and loaded into the address path channel x registers for a normal start and into channel y registers for a preemption start. After the minor loop completes execution, the address path hardware writes the new values for the TCD*n*_{SADDR, DADDR, CITER} back to local memory. If the major iteration count is exhausted, additional processing is performed, including the final address pointer updates, reloading the TCD*n*_CITER field, and a possible fetch of the next TCD*n* from memory as part of a scatter/gather operation.
Data path	This block implements the bus master read/write datapath. It includes a data buffer and the necessary multiplex logic to support any required data alignment. The internal read data bus is the primary input, and the internal write data bus is the primary output. The address and data path modules directly support the 2-stage pipelined internal bus. The address path module represents the 1st stage of the bus pipeline (address phase), while the data path module implements the 2nd stage of the pipeline (data phase).
Program model/channel arbitration	This block implements the first section of the eDMA programming model as well as the channel arbitration logic. The programming model registers are connected to the internal peripheral bus. The eDMA peripheral request inputs and interrupt request outputs are also connected to this block (via control logic).
Control	This block provides all the control functions for the eDMA engine. For data transfers where the source and destination sizes are equal, the eDMA engine performs a series of source read/ destination write operations until the number of bytes specified in the minor loop byte count has moved. For descriptors where the sizes are not equal, multiple accesses of the smaller size data are required for each reference of the larger size. As an example, if the source size references 16-bit data and the destination is 32-bit data, two reads are performed, then one 32-bit write.

The transfer-control descriptor local memory is further partitioned into:

Table 17-3. Transfer control descriptor memory

Submodule	Description
Memory controller	This logic implements the required dual-ported controller, managing accesses from the eDMA engine as well as references from the internal peripheral bus. As noted earlier, in the event of simultaneous accesses, the eDMA engine is given priority and the peripheral transaction is stalled.
Memory array	TCD storage for each channel's transfer profile.

17.2.3 Features

The eDMA is a highly programmable data-transfer engine optimized to minimize any required intervention from the host processor. It is intended for use in applications where the data size to be transferred is statically known and not defined within the transferred data itself. The eDMA module features:

- All data movement via dual-address transfers: read from source, write to destination
  - Programmable source and destination addresses and transfer size
  - Support for enhanced addressing modes
- 32-channel implementation that performs complex data transfers with minimal intervention from a host processor
  - Internal data buffer, used as temporary storage to support 16- and 32-byte transfers
  - Connections to the crossbar switch for bus mastering the data movement
- Transfer control descriptor (TCD) organized to support two-deep, nested transfer operations
  - 32-byte TCD stored in local memory for each channel
  - An inner data transfer loop defined by a minor byte transfer count
  - An outer data transfer loop defined by a major iteration count
- Channel activation via one of three methods:
  - Explicit software initiation
  - Initiation via a channel-to-channel linking mechanism for continuous transfers
  - Peripheral-paced hardware requests, one per channel
- Fixed-priority and round-robin channel arbitration
- Channel completion reported via programmable interrupt requests
  - One interrupt per channel, which can be asserted at completion of major iteration count
  - Programmable error terminations per channel and logically summed together to form one error interrupt to the interrupt controller
- Programmable support for scatter/gather DMA processing
- Support for complex data structures

In the discussion of this module, *n* is used to reference the channel number.

17.3 Modes of operation

The eDMA operates in the following modes:

Table 17-4. Modes of operation

Mode	Description
Normal	In Normal mode, the eDMA transfers data between a source and a destination. The source and destination can be a memory block or an I/O block capable of operation with the eDMA. A service request initiates a transfer of a specific number of bytes (NBYTES) as specified in the transfer control descriptor (TCD). The minor loop is the sequence of read-write operations that transfers these NBYTES per service request. Each service request executes one iteration of the major loop, which transfers NBYTES of data.
Debug	DMA operation is configurable in Debug mode via the control register: • If CR[EDBG] is cleared, the DMA continues to operate. • If CR[EDBG] is set, the eDMA stops transferring data. If Debug mode is entered while a channel is active, the eDMA continues operation until the channel retires.

17.4 Memory map/register definition

The eDMA's programming model is partitioned into two regions:

- The first region defines a number of registers providing control functions
- The second region corresponds to the local transfer control descriptor (TCD) memory

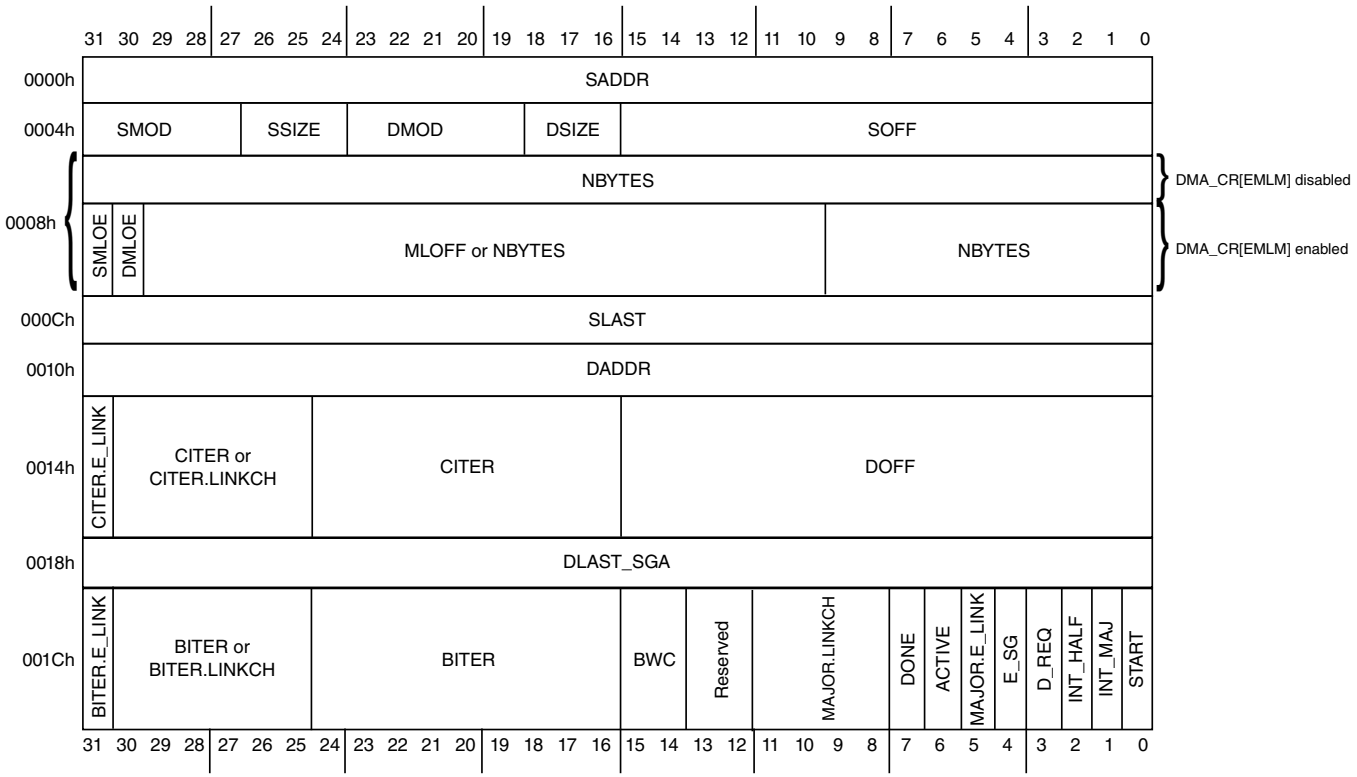
17.4.1 TCD memory

Each channel requires a 32-byte transfer control descriptor for defining the desired data movement operation. The channel descriptors are stored in the local memory in sequential order: channel 0, channel 1, ... channel 31. Each TCD*n* definition is presented as 11 registers of 16 or 32 bits.

17.4.2 TCD initialization

Prior to activating a channel, you must initialize its TCD with the appropriate transfer profile.

17.4.3 TCD structure



17.4.4 Reserved memory and bit fields

- Reading reserved bits in a register returns the value of zero.
- Writes to reserved bits in a register are ignored.
- Reading or writing a reserved memory location generates a bus error.

17.4.5 Endianness

This module's memory map uses big-endian ordering. This means:

- For 8-bit registers, the lower address byte is read as the most significant byte.
- For 16-bit registers, the lower address word is read as the most significant word.

The following figure provides examples of this.

Example 1: 8-bit register structure

Address	Register Data
00h	AAh
01h	BBh
02h	CCh
03h	DDh

For this structure, an 8-bit read of address 00h will yield DDh.

Example 2: 16-bit register structure

Address	Register Data
00h	AABBh
02h	CCDDh

For this structure, a 16-bit read of address 00h will yield CCDDh.

Figure 17-2. Examples of big-endian register access results

17.4.6 DMA register descriptions

17.4.6.1 DMA Memory map

DMA base address: 2C0_0000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Control Register (CR)	32	RW	0000_0400h
4h	Error Status Register (ES)	32	RO	0000_0000h
Ch	Enable Request Register (ERQ)	32	RW	0000_0000h
14h	Enable Error Interrupt Register (EEI)	32	RW	0000_0000h
18h	Set Enable Request Register (SERQ)	8	WORZ	00h
19h	Clear Enable Request Register (CERQ)	8	WORZ	00h
1Ah	Set Enable Error Interrupt Register (SEEI)	8	WORZ	00h
1Bh	Clear Enable Error Interrupt Register (CEEI)	8	WORZ	00h
1Ch	Clear Interrupt Request Register (CINT)	8	WORZ	00h
1Dh	Clear Error Register (CERR)	8	WORZ	00h
1Eh	Set START Bit Register (SSRT)	8	WORZ	00h
1Fh	Clear DONE Status Bit Register (CDNE)	8	WORZ	00h
24h	Interrupt Request Register (INT)	32	W1C	0000_0000h
2Ch	Error Register (ERR)	32	W1C	0000_0000h
34h	Hardware Request Status Register (HRS)	32	RO	0000_0000h
100h	Channel Priority Register (DCHPRI3)	8	RW	03h
101h	Channel Priority Register (DCHPRI2)	8	RW	02h
102h	Channel Priority Register (DCHPRI1)	8	RW	01h

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
103h	Channel Priority Register (DCHPRI0)	8	RW	00h
104h	Channel Priority Register (DCHPRI7)	8	RW	07h
105h	Channel Priority Register (DCHPRI6)	8	RW	06h
106h	Channel Priority Register (DCHPRI5)	8	RW	05h
107h	Channel Priority Register (DCHPRI4)	8	RW	04h
108h	Channel Priority Register (DCHPRI11)	8	RW	0Bh
109h	Channel Priority Register (DCHPRI10)	8	RW	0Ah
10Ah	Channel Priority Register (DCHPRI9)	8	RW	09h
10Bh	Channel Priority Register (DCHPRI8)	8	RW	08h
10Ch	Channel Priority Register (DCHPRI15)	8	RW	0Fh
10Dh	Channel Priority Register (DCHPRI14)	8	RW	0Eh
10Eh	Channel Priority Register (DCHPRI13)	8	RW	0Dh
10Fh	Channel Priority Register (DCHPRI12)	8	RW	0Ch
110h	Channel Priority Register (DCHPRI19)	8	RW	13h
111h	Channel Priority Register (DCHPRI18)	8	RW	12h
112h	Channel Priority Register (DCHPRI17)	8	RW	11h
113h	Channel Priority Register (DCHPRI16)	8	RW	10h
114h	Channel Priority Register (DCHPRI23)	8	RW	17h
115h	Channel Priority Register (DCHPRI22)	8	RW	16h
116h	Channel Priority Register (DCHPRI21)	8	RW	15h
117h	Channel Priority Register (DCHPRI20)	8	RW	14h
118h	Channel Priority Register (DCHPRI27)	8	RW	1Bh
119h	Channel Priority Register (DCHPRI26)	8	RW	1Ah
11Ah	Channel Priority Register (DCHPRI25)	8	RW	19h
11Bh	Channel Priority Register (DCHPRI24)	8	RW	18h
11Ch	Channel Priority Register (DCHPRI31)	8	RW	1Fh
11Dh	Channel Priority Register (DCHPRI30)	8	RW	1Eh
11Eh	Channel Priority Register (DCHPRI29)	8	RW	1Dh
11Fh	Channel Priority Register (DCHPRI28)	8	RW	1Ch
140h - 15Fh	Channel n Master ID Register (DCHMID0 - DCHMID31)	8	RW	00h
1000h	TCD Source Address (TCD0_SADDR)	32	RW	See description.
1004h	TCD Transfer Attributes (TCD0_ATTR)	16	RW	See description.
1006h	TCD Signed Source Address Offset (TCD0_SOFF)	16	RW	See description.
1008h	TCD Minor Byte Count (Minor Loop Mapping Disabled) (TCD0_NBYTES_MLNO)	32	RW	See description.
1008h	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (TCD0_NBYTES_MLOFFNO)	32	RW	See description.

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
1008h	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (TCD0_NBYTES_MLOFFYES)	32	RW	See description.
100Ch	TCD Last Source Address Adjustment (TCD0_SLAST)	32	RW	See description.
1010h	TCD Destination Address (TCD0_DADDR)	32	RW	See description.
1014h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD0_CITER_ELINKNO)	16	RW	See description.
1014h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD0_CITER_ELINKYES)	16	RW	See description.
1016h	TCD Signed Destination Address Offset (TCD0_DOFF)	16	RW	See description.
1018h	TCD Last Destination Address Adjustment/Scatter Gather Address (TCD0_DLASTSGA)	32	RW	See description.
101Ch	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD0_BITER_ELINKNO)	16	RW	See description.
101Ch	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD0_BITER_ELINKYES)	16	RW	See description.
101Eh	TCD Control and Status (TCD0_CSR)	16	RW	See description.
1020h	TCD Source Address (TCD1_SADDR)	32	RW	See description.
1024h	TCD Transfer Attributes (TCD1_ATTR)	16	RW	See description.
1026h	TCD Signed Source Address Offset (TCD1_SOFF)	16	RW	See description.
1028h	TCD Minor Byte Count (Minor Loop Mapping Disabled) (TCD1_NBYTES_MLNO)	32	RW	See description.
1028h	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (TCD1_NBYTES_MLOFFNO)	32	RW	See description.
1028h	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (TCD1_NBYTES_MLOFFYES)	32	RW	See description.
102Ch	TCD Last Source Address Adjustment (TCD1_SLAST)	32	RW	See description.
1030h	TCD Destination Address (TCD1_DADDR)	32	RW	See description.
1034h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD1_CITER_ELINKNO)	16	RW	See description.
1034h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD1_CITER_ELINKYES)	16	RW	See description.
1036h	TCD Signed Destination Address Offset (TCD1_DOFF)	16	RW	See description.
1038h	TCD Last Destination Address Adjustment/Scatter Gather Address (TCD1_DLASTSGA)	32	RW	See description.
103Ch	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD1_BITER_ELINKNO)	16	RW	See description.

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
103Ch	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD1_BITER_ELINKYES)	16	RW	See description.
103Eh	TCD Control and Status (TCD1_CSR)	16	RW	See description.
1040h	TCD Source Address (TCD2_SADDR)	32	RW	See description.
1044h	TCD Transfer Attributes (TCD2_ATTR)	16	RW	See description.
1046h	TCD Signed Source Address Offset (TCD2_SOFF)	16	RW	See description.
1048h	TCD Minor Byte Count (Minor Loop Mapping Disabled) (TCD2_NBYTES_MLNO)	32	RW	See description.
1048h	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (TCD2_NBYTES_MLOFFNO)	32	RW	See description.
1048h	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (TCD2_NBYTES_MLOFFYES)	32	RW	See description.
104Ch	TCD Last Source Address Adjustment (TCD2_SLAST)	32	RW	See description.
1050h	TCD Destination Address (TCD2_DADDR)	32	RW	See description.
1054h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD2_CITER_ELINKNO)	16	RW	See description.
1054h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD2_CITER_ELINKYES)	16	RW	See description.
1056h	TCD Signed Destination Address Offset (TCD2_DOFF)	16	RW	See description.
1058h	TCD Last Destination Address Adjustment/Scatter Gather Address (TCD2_DLASTSGA)	32	RW	See description.
105Ch	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD2_BITER_ELINKNO)	16	RW	See description.
105Ch	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD2_BITER_ELINKYES)	16	RW	See description.
105Eh	TCD Control and Status (TCD2_CSR)	16	RW	See description.
1060h	TCD Source Address (TCD3_SADDR)	32	RW	See description.
1064h	TCD Transfer Attributes (TCD3_ATTR)	16	RW	See description.
1066h	TCD Signed Source Address Offset (TCD3_SOFF)	16	RW	See description.
1068h	TCD Minor Byte Count (Minor Loop Mapping Disabled) (TCD3_NBYTES_MLNO)	32	RW	See description.
1068h	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (TCD3_NBYTES_MLOFFNO)	32	RW	See description.
1068h	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (TCD3_NBYTES_MLOFFYES)	32	RW	See description.

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
106Ch	TCD Last Source Address Adjustment (TCD3_SLAST)	32	RW	See description.
1070h	TCD Destination Address (TCD3_DADDR)	32	RW	See description.
1074h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD3_CITER_ELINKNO)	16	RW	See description.
1074h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD3_CITER_ELINKYES)	16	RW	See description.
1076h	TCD Signed Destination Address Offset (TCD3_DOFF)	16	RW	See description.
1078h	TCD Last Destination Address Adjustment/Scatter Gather Address (TCD3_DLASTSGA)	32	RW	See description.
107Ch	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD3_BITER_ELINKNO)	16	RW	See description.
107Ch	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD3_BITER_ELINKYES)	16	RW	See description.
107Eh	TCD Control and Status (TCD3_CSR)	16	RW	See description.
1080h	TCD Source Address (TCD4_SADDR)	32	RW	See description.
1084h	TCD Transfer Attributes (TCD4_ATTR)	16	RW	See description.
1086h	TCD Signed Source Address Offset (TCD4_SOFF)	16	RW	See description.
1088h	TCD Minor Byte Count (Minor Loop Mapping Disabled) (TCD4_NBYTES_MLNO)	32	RW	See description.
1088h	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (TCD4_NBYTES_MLOFFNO)	32	RW	See description.
1088h	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (TCD4_NBYTES_MLOFFYES)	32	RW	See description.
108Ch	TCD Last Source Address Adjustment (TCD4_SLAST)	32	RW	See description.
1090h	TCD Destination Address (TCD4_DADDR)	32	RW	See description.
1094h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD4_CITER_ELINKNO)	16	RW	See description.
1094h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD4_CITER_ELINKYES)	16	RW	See description.
1096h	TCD Signed Destination Address Offset (TCD4_DOFF)	16	RW	See description.
1098h	TCD Last Destination Address Adjustment/Scatter Gather Address (TCD4_DLASTSGA)	32	RW	See description.
109Ch	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD4_BITER_ELINKNO)	16	RW	See description.
109Ch	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD4_BITER_ELINKYES)	16	RW	See description.

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
109Eh	TCD Control and Status (TCD4_CSR)	16	RW	See description.
10A0h	TCD Source Address (TCD5_SADDR)	32	RW	See description.
10A4h	TCD Transfer Attributes (TCD5_ATTR)	16	RW	See description.
10A6h	TCD Signed Source Address Offset (TCD5_SOFF)	16	RW	See description.
10A8h	TCD Minor Byte Count (Minor Loop Mapping Disabled) (TCD5_NBYTES_MLNO)	32	RW	See description.
10A8h	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (TCD5_NBYTES_MLOFFNO)	32	RW	See description.
10A8h	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (TCD5_NBYTES_MLOFFYES)	32	RW	See description.
10ACh	TCD Last Source Address Adjustment (TCD5_SLAST)	32	RW	See description.
10B0h	TCD Destination Address (TCD5_DADDR)	32	RW	See description.
10B4h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD5_CITER_ELINKNO)	16	RW	See description.
10B4h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD5_CITER_ELINKYES)	16	RW	See description.
10B6h	TCD Signed Destination Address Offset (TCD5_DOFF)	16	RW	See description.
10B8h	TCD Last Destination Address Adjustment/Scatter Gather Address (TCD5_DLASTSGA)	32	RW	See description.
10BCh	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD5_BITER_ELINKNO)	16	RW	See description.
10BCh	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD5_BITER_ELINKYES)	16	RW	See description.
10BEh	TCD Control and Status (TCD5_CSR)	16	RW	See description.
10C0h	TCD Source Address (TCD6_SADDR)	32	RW	See description.
10C4h	TCD Transfer Attributes (TCD6_ATTR)	16	RW	See description.
10C6h	TCD Signed Source Address Offset (TCD6_SOFF)	16	RW	See description.
10C8h	TCD Minor Byte Count (Minor Loop Mapping Disabled) (TCD6_NBYTES_MLNO)	32	RW	See description.
10C8h	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (TCD6_NBYTES_MLOFFNO)	32	RW	See description.
10C8h	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (TCD6_NBYTES_MLOFFYES)	32	RW	See description.
10CCh	TCD Last Source Address Adjustment (TCD6_SLAST)	32	RW	See description.

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
10D0h	TCD Destination Address (TCD6_DADDR)	32	RW	See description.
10D4h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD6_CITER_ELINKNO)	16	RW	See description.
10D4h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD6_CITER_ELINKYES)	16	RW	See description.
10D6h	TCD Signed Destination Address Offset (TCD6_DOFF)	16	RW	See description.
10D8h	TCD Last Destination Address Adjustment/Scatter Gather Address (TCD6_DLASTSGA)	32	RW	See description.
10DCh	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD6_BITER_ELINKNO)	16	RW	See description.
10DCh	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD6_BITER_ELINKYES)	16	RW	See description.
10DEh	TCD Control and Status (TCD6_CSR)	16	RW	See description.
10E0h	TCD Source Address (TCD7_SADDR)	32	RW	See description.
10E4h	TCD Transfer Attributes (TCD7_ATTR)	16	RW	See description.
10E6h	TCD Signed Source Address Offset (TCD7_SOFF)	16	RW	See description.
10E8h	TCD Minor Byte Count (Minor Loop Mapping Disabled) (TCD7_NBYTES_MLNO)	32	RW	See description.
10E8h	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (TCD7_NBYTES_MLOFFNO)	32	RW	See description.
10E8h	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (TCD7_NBYTES_MLOFFYES)	32	RW	See description.
10ECh	TCD Last Source Address Adjustment (TCD7_SLAST)	32	RW	See description.
10F0h	TCD Destination Address (TCD7_DADDR)	32	RW	See description.
10F4h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD7_CITER_ELINKNO)	16	RW	See description.
10F4h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD7_CITER_ELINKYES)	16	RW	See description.
10F6h	TCD Signed Destination Address Offset (TCD7_DOFF)	16	RW	See description.
10F8h	TCD Last Destination Address Adjustment/Scatter Gather Address (TCD7_DLASTSGA)	32	RW	See description.
10FCh	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD7_BITER_ELINKNO)	16	RW	See description.
10FCh	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD7_BITER_ELINKYES)	16	RW	See description.
10FEh	TCD Control and Status (TCD7_CSR)	16	RW	See description.

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
1100h	TCD Source Address (TCD8_SADDR)	32	RW	See description.
1104h	TCD Transfer Attributes (TCD8_ATTR)	16	RW	See description.
1106h	TCD Signed Source Address Offset (TCD8_SOFF)	16	RW	See description.
1108h	TCD Minor Byte Count (Minor Loop Mapping Disabled) (TCD8_NBYTES_MLNO)	32	RW	See description.
1108h	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (TCD8_NBYTES_MLOFFNO)	32	RW	See description.
1108h	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (TCD8_NBYTES_MLOFFYES)	32	RW	See description.
110Ch	TCD Last Source Address Adjustment (TCD8_SLAST)	32	RW	See description.
1110h	TCD Destination Address (TCD8_DADDR)	32	RW	See description.
1114h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD8_CITER_ELINKNO)	16	RW	See description.
1114h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD8_CITER_ELINKYES)	16	RW	See description.
1116h	TCD Signed Destination Address Offset (TCD8_DOFF)	16	RW	See description.
1118h	TCD Last Destination Address Adjustment/Scatter Gather Address (TCD8_DLASTSGA)	32	RW	See description.
111Ch	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD8_BITER_ELINKNO)	16	RW	See description.
111Ch	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD8_BITER_ELINKYES)	16	RW	See description.
111Eh	TCD Control and Status (TCD8_CSR)	16	RW	See description.
1120h	TCD Source Address (TCD9_SADDR)	32	RW	See description.
1124h	TCD Transfer Attributes (TCD9_ATTR)	16	RW	See description.
1126h	TCD Signed Source Address Offset (TCD9_SOFF)	16	RW	See description.
1128h	TCD Minor Byte Count (Minor Loop Mapping Disabled) (TCD9_NBYTES_MLNO)	32	RW	See description.
1128h	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (TCD9_NBYTES_MLOFFNO)	32	RW	See description.
1128h	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (TCD9_NBYTES_MLOFFYES)	32	RW	See description.
112Ch	TCD Last Source Address Adjustment (TCD9_SLAST)	32	RW	See description.
1130h	TCD Destination Address (TCD9_DADDR)	32	RW	See description.

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
1134h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD9_CITER_ELINKNO)	16	RW	See description.
1134h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD9_CITER_ELINKYES)	16	RW	See description.
1136h	TCD Signed Destination Address Offset (TCD9_DOFF)	16	RW	See description.
1138h	TCD Last Destination Address Adjustment/Scatter Gather Address (TCD9_DLASTSGA)	32	RW	See description.
113Ch	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD9_BITER_ELINKNO)	16	RW	See description.
113Ch	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD9_BITER_ELINKYES)	16	RW	See description.
113Eh	TCD Control and Status (TCD9_CSR)	16	RW	See description.
1140h	TCD Source Address (TCD10_SADDR)	32	RW	See description.
1144h	TCD Transfer Attributes (TCD10_ATTR)	16	RW	See description.
1146h	TCD Signed Source Address Offset (TCD10_SOFF)	16	RW	See description.
1148h	TCD Minor Byte Count (Minor Loop Mapping Disabled) (TCD10_NBYTES_MLNO)	32	RW	See description.
1148h	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (TCD10_NBYTES_MLOFFNO)	32	RW	See description.
1148h	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (TCD10_NBYTES_MLOFFYES)	32	RW	See description.
114Ch	TCD Last Source Address Adjustment (TCD10_SLAST)	32	RW	See description.
1150h	TCD Destination Address (TCD10_DADDR)	32	RW	See description.
1154h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD10_CITER_ELINKNO)	16	RW	See description.
1154h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD10_CITER_ELINKYES)	16	RW	See description.
1156h	TCD Signed Destination Address Offset (TCD10_DOFF)	16	RW	See description.
1158h	TCD Last Destination Address Adjustment/Scatter Gather Address (TCD10_DLASTSGA)	32	RW	See description.
115Ch	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD10_BITER_ELINKNO)	16	RW	See description.
115Ch	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD10_BITER_ELINKYES)	16	RW	See description.
115Eh	TCD Control and Status (TCD10_CSR)	16	RW	See description.
1160h	TCD Source Address (TCD11_SADDR)	32	RW	See description.

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
1164h	TCD Transfer Attributes (TCD11_ATTR)	16	RW	See description.
1166h	TCD Signed Source Address Offset (TCD11_SOFF)	16	RW	See description.
1168h	TCD Minor Byte Count (Minor Loop Mapping Disabled) (TCD11_NBYTES_MLNO)	32	RW	See description.
1168h	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (TCD11_NBYTES_MLOFFNO)	32	RW	See description.
1168h	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (TCD11_NBYTES_MLOFFYES)	32	RW	See description.
116Ch	TCD Last Source Address Adjustment (TCD11_SLAST)	32	RW	See description.
1170h	TCD Destination Address (TCD11_DADDR)	32	RW	See description.
1174h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD11_CITER_ELINKNO)	16	RW	See description.
1174h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD11_CITER_ELINKYES)	16	RW	See description.
1176h	TCD Signed Destination Address Offset (TCD11_DOFF)	16	RW	See description.
1178h	TCD Last Destination Address Adjustment/Scatter Gather Address (TCD11_DLASTSGA)	32	RW	See description.
117Ch	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD11_BITER_ELINKNO)	16	RW	See description.
117Ch	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD11_BITER_ELINKYES)	16	RW	See description.
117Eh	TCD Control and Status (TCD11_CSR)	16	RW	See description.
1180h	TCD Source Address (TCD12_SADDR)	32	RW	See description.
1184h	TCD Transfer Attributes (TCD12_ATTR)	16	RW	See description.
1186h	TCD Signed Source Address Offset (TCD12_SOFF)	16	RW	See description.
1188h	TCD Minor Byte Count (Minor Loop Mapping Disabled) (TCD12_NBYTES_MLNO)	32	RW	See description.
1188h	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (TCD12_NBYTES_MLOFFNO)	32	RW	See description.
1188h	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (TCD12_NBYTES_MLOFFYES)	32	RW	See description.
118Ch	TCD Last Source Address Adjustment (TCD12_SLAST)	32	RW	See description.
1190h	TCD Destination Address (TCD12_DADDR)	32	RW	See description.
1194h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD12_CITER_ELINKNO)	16	RW	See description.

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
1194h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD12_CITER_ELINKYES)	16	RW	See description.
1196h	TCD Signed Destination Address Offset (TCD12_DOFF)	16	RW	See description.
1198h	TCD Last Destination Address Adjustment/Scatter Gather Address (TCD12_DLASTSGA)	32	RW	See description.
119Ch	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD12_BITER_ELINKNO)	16	RW	See description.
119Ch	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD12_BITER_ELINKYES)	16	RW	See description.
119Eh	TCD Control and Status (TCD12_CSR)	16	RW	See description.
11A0h	TCD Source Address (TCD13_SADDR)	32	RW	See description.
11A4h	TCD Transfer Attributes (TCD13_ATTR)	16	RW	See description.
11A6h	TCD Signed Source Address Offset (TCD13_SOFF)	16	RW	See description.
11A8h	TCD Minor Byte Count (Minor Loop Mapping Disabled) (TCD13_NBYTES_MLNO)	32	RW	See description.
11A8h	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (TCD13_NBYTES_MLOFFNO)	32	RW	See description.
11A8h	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (TCD13_NBYTES_MLOFFYES)	32	RW	See description.
11ACh	TCD Last Source Address Adjustment (TCD13_SLAST)	32	RW	See description.
11B0h	TCD Destination Address (TCD13_DADDR)	32	RW	See description.
11B4h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD13_CITER_ELINKNO)	16	RW	See description.
11B4h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD13_CITER_ELINKYES)	16	RW	See description.
11B6h	TCD Signed Destination Address Offset (TCD13_DOFF)	16	RW	See description.
11B8h	TCD Last Destination Address Adjustment/Scatter Gather Address (TCD13_DLASTSGA)	32	RW	See description.
11BCh	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD13_BITER_ELINKNO)	16	RW	See description.
11BCh	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD13_BITER_ELINKYES)	16	RW	See description.
11BEh	TCD Control and Status (TCD13_CSR)	16	RW	See description.
11C0h	TCD Source Address (TCD14_SADDR)	32	RW	See description.
11C4h	TCD Transfer Attributes (TCD14_ATTR)	16	RW	See description.

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
11C6h	TCD Signed Source Address Offset (TCD14_SOFF)	16	RW	See description.
11C8h	TCD Minor Byte Count (Minor Loop Mapping Disabled) (TCD14_NBYTES_MLNO)	32	RW	See description.
11C8h	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (TCD14_NBYTES_MLOFFNO)	32	RW	See description.
11C8h	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (TCD14_NBYTES_MLOFFYES)	32	RW	See description.
11CCh	TCD Last Source Address Adjustment (TCD14_SLAST)	32	RW	See description.
11D0h	TCD Destination Address (TCD14_DADDR)	32	RW	See description.
11D4h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD14_CITER_ELINKNO)	16	RW	See description.
11D4h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD14_CITER_ELINKYES)	16	RW	See description.
11D6h	TCD Signed Destination Address Offset (TCD14_DOFF)	16	RW	See description.
11D8h	TCD Last Destination Address Adjustment/Scatter Gather Address (TCD14_DLASTSGA)	32	RW	See description.
11DCh	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD14_BITER_ELINKNO)	16	RW	See description.
11DCh	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD14_BITER_ELINKYES)	16	RW	See description.
11DEh	TCD Control and Status (TCD14_CSR)	16	RW	See description.
11E0h	TCD Source Address (TCD15_SADDR)	32	RW	See description.
11E4h	TCD Transfer Attributes (TCD15_ATTR)	16	RW	See description.
11E6h	TCD Signed Source Address Offset (TCD15_SOFF)	16	RW	See description.
11E8h	TCD Minor Byte Count (Minor Loop Mapping Disabled) (TCD15_NBYTES_MLNO)	32	RW	See description.
11E8h	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (TCD15_NBYTES_MLOFFNO)	32	RW	See description.
11E8h	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (TCD15_NBYTES_MLOFFYES)	32	RW	See description.
11ECh	TCD Last Source Address Adjustment (TCD15_SLAST)	32	RW	See description.
11F0h	TCD Destination Address (TCD15_DADDR)	32	RW	See description.
11F4h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD15_CITER_ELINKNO)	16	RW	See description.
11F4h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD15_CITER_ELINKYES)	16	RW	See description.

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
11F6h	TCD Signed Destination Address Offset (TCD15_DOFF)	16	RW	See description.
11F8h	TCD Last Destination Address Adjustment/Scatter Gather Address (TCD15_DLASTSGA)	32	RW	See description.
11FCh	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD15_BITER_ELINKNO)	16	RW	See description.
11FCh	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD15_BITER_ELINKYES)	16	RW	See description.
11FEh	TCD Control and Status (TCD15_CSR)	16	RW	See description.
1200h	TCD Source Address (TCD16_SADDR)	32	RW	See description.
1204h	TCD Transfer Attributes (TCD16_ATTR)	16	RW	See description.
1206h	TCD Signed Source Address Offset (TCD16_SOFF)	16	RW	See description.
1208h	TCD Minor Byte Count (Minor Loop Mapping Disabled) (TCD16_NBYTES_MLNO)	32	RW	See description.
1208h	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (TCD16_NBYTES_MLOFFNO)	32	RW	See description.
1208h	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (TCD16_NBYTES_MLOFFYES)	32	RW	See description.
120Ch	TCD Last Source Address Adjustment (TCD16_SLAST)	32	RW	See description.
1210h	TCD Destination Address (TCD16_DADDR)	32	RW	See description.
1214h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD16_CITER_ELINKNO)	16	RW	See description.
1214h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD16_CITER_ELINKYES)	16	RW	See description.
1216h	TCD Signed Destination Address Offset (TCD16_DOFF)	16	RW	See description.
1218h	TCD Last Destination Address Adjustment/Scatter Gather Address (TCD16_DLASTSGA)	32	RW	See description.
121Ch	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD16_BITER_ELINKNO)	16	RW	See description.
121Ch	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD16_BITER_ELINKYES)	16	RW	See description.
121Eh	TCD Control and Status (TCD16_CSR)	16	RW	See description.
1220h	TCD Source Address (TCD17_SADDR)	32	RW	See description.
1224h	TCD Transfer Attributes (TCD17_ATTR)	16	RW	See description.
1226h	TCD Signed Source Address Offset (TCD17_SOFF)	16	RW	See description.

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
1228h	TCD Minor Byte Count (Minor Loop Mapping Disabled) (TCD17_NBYTES_MLNO)	32	RW	See description.
1228h	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (TCD17_NBYTES_MLOFFNO)	32	RW	See description.
1228h	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (TCD17_NBYTES_MLOFFYES)	32	RW	See description.
122Ch	TCD Last Source Address Adjustment (TCD17_SLAST)	32	RW	See description.
1230h	TCD Destination Address (TCD17_DADDR)	32	RW	See description.
1234h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD17_CITER_ELINKNO)	16	RW	See description.
1234h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD17_CITER_ELINKYES)	16	RW	See description.
1236h	TCD Signed Destination Address Offset (TCD17_DOFF)	16	RW	See description.
1238h	TCD Last Destination Address Adjustment/Scatter Gather Address (TCD17_DLASTSGA)	32	RW	See description.
123Ch	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD17_BITER_ELINKNO)	16	RW	See description.
123Ch	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD17_BITER_ELINKYES)	16	RW	See description.
123Eh	TCD Control and Status (TCD17_CSR)	16	RW	See description.
1240h	TCD Source Address (TCD18_SADDR)	32	RW	See description.
1244h	TCD Transfer Attributes (TCD18_ATTR)	16	RW	See description.
1246h	TCD Signed Source Address Offset (TCD18_SOFF)	16	RW	See description.
1248h	TCD Minor Byte Count (Minor Loop Mapping Disabled) (TCD18_NBYTES_MLNO)	32	RW	See description.
1248h	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (TCD18_NBYTES_MLOFFNO)	32	RW	See description.
1248h	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (TCD18_NBYTES_MLOFFYES)	32	RW	See description.
124Ch	TCD Last Source Address Adjustment (TCD18_SLAST)	32	RW	See description.
1250h	TCD Destination Address (TCD18_DADDR)	32	RW	See description.
1254h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD18_CITER_ELINKNO)	16	RW	See description.
1254h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD18_CITER_ELINKYES)	16	RW	See description.
1256h	TCD Signed Destination Address Offset (TCD18_DOFF)	16	RW	See description.

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
1258h	TCD Last Destination Address Adjustment/Scatter Gather Address (TCD18_DLASTSGA)	32	RW	See description.
125Ch	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD18_BITER_ELINKNO)	16	RW	See description.
125Ch	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD18_BITER_ELINKYES)	16	RW	See description.
125Eh	TCD Control and Status (TCD18_CSR)	16	RW	See description.
1260h	TCD Source Address (TCD19_SADDR)	32	RW	See description.
1264h	TCD Transfer Attributes (TCD19_ATTR)	16	RW	See description.
1266h	TCD Signed Source Address Offset (TCD19_SOFF)	16	RW	See description.
1268h	TCD Minor Byte Count (Minor Loop Mapping Disabled) (TCD19_NBYTES_MLNO)	32	RW	See description.
1268h	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (TCD19_NBYTES_MLOFFNO)	32	RW	See description.
1268h	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (TCD19_NBYTES_MLOFFYES)	32	RW	See description.
126Ch	TCD Last Source Address Adjustment (TCD19_SLAST)	32	RW	See description.
1270h	TCD Destination Address (TCD19_DADDR)	32	RW	See description.
1274h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD19_CITER_ELINKNO)	16	RW	See description.
1274h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD19_CITER_ELINKYES)	16	RW	See description.
1276h	TCD Signed Destination Address Offset (TCD19_DOFF)	16	RW	See description.
1278h	TCD Last Destination Address Adjustment/Scatter Gather Address (TCD19_DLASTSGA)	32	RW	See description.
127Ch	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD19_BITER_ELINKNO)	16	RW	See description.
127Ch	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD19_BITER_ELINKYES)	16	RW	See description.
127Eh	TCD Control and Status (TCD19_CSR)	16	RW	See description.
1280h	TCD Source Address (TCD20_SADDR)	32	RW	See description.
1284h	TCD Transfer Attributes (TCD20_ATTR)	16	RW	See description.
1286h	TCD Signed Source Address Offset (TCD20_SOFF)	16	RW	See description.
1288h	TCD Minor Byte Count (Minor Loop Mapping Disabled) (TCD20_NBYTES_MLNO)	32	RW	See description.

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
1288h	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (TCD20_NBYTES_MLOFFNO)	32	RW	See description.
1288h	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (TCD20_NBYTES_MLOFFYES)	32	RW	See description.
128Ch	TCD Last Source Address Adjustment (TCD20_SLAST)	32	RW	See description.
1290h	TCD Destination Address (TCD20_DADDR)	32	RW	See description.
1294h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD20_CITER_ELINKNO)	16	RW	See description.
1294h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD20_CITER_ELINKYES)	16	RW	See description.
1296h	TCD Signed Destination Address Offset (TCD20_DOFF)	16	RW	See description.
1298h	TCD Last Destination Address Adjustment/Scatter Gather Address (TCD20_DLASTSGA)	32	RW	See description.
129Ch	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD20_BITER_ELINKNO)	16	RW	See description.
129Ch	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD20_BITER_ELINKYES)	16	RW	See description.
129Eh	TCD Control and Status (TCD20_CSR)	16	RW	See description.
12A0h	TCD Source Address (TCD21_SADDR)	32	RW	See description.
12A4h	TCD Transfer Attributes (TCD21_ATTR)	16	RW	See description.
12A6h	TCD Signed Source Address Offset (TCD21_SOFF)	16	RW	See description.
12A8h	TCD Minor Byte Count (Minor Loop Mapping Disabled) (TCD21_NBYTES_MLNO)	32	RW	See description.
12A8h	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (TCD21_NBYTES_MLOFFNO)	32	RW	See description.
12A8h	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (TCD21_NBYTES_MLOFFYES)	32	RW	See description.
12ACh	TCD Last Source Address Adjustment (TCD21_SLAST)	32	RW	See description.
12B0h	TCD Destination Address (TCD21_DADDR)	32	RW	See description.
12B4h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD21_CITER_ELINKNO)	16	RW	See description.
12B4h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD21_CITER_ELINKYES)	16	RW	See description.
12B6h	TCD Signed Destination Address Offset (TCD21_DOFF)	16	RW	See description.
12B8h	TCD Last Destination Address Adjustment/Scatter Gather Address (TCD21_DLASTSGA)	32	RW	See description.

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
12BCh	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD21_BITER_ELINKNO)	16	RW	See description.
12BCh	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD21_BITER_ELINKYES)	16	RW	See description.
12BEh	TCD Control and Status (TCD21_CSR)	16	RW	See description.
12C0h	TCD Source Address (TCD22_SADDR)	32	RW	See description.
12C4h	TCD Transfer Attributes (TCD22_ATTR)	16	RW	See description.
12C6h	TCD Signed Source Address Offset (TCD22_SOFF)	16	RW	See description.
12C8h	TCD Minor Byte Count (Minor Loop Mapping Disabled) (TCD22_NBYTES_MLNO)	32	RW	See description.
12C8h	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (TCD22_NBYTES_MLOFFNO)	32	RW	See description.
12C8h	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (TCD22_NBYTES_MLOFFYES)	32	RW	See description.
12CCh	TCD Last Source Address Adjustment (TCD22_SLAST)	32	RW	See description.
12D0h	TCD Destination Address (TCD22_DADDR)	32	RW	See description.
12D4h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD22_CITER_ELINKNO)	16	RW	See description.
12D4h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD22_CITER_ELINKYES)	16	RW	See description.
12D6h	TCD Signed Destination Address Offset (TCD22_DOFF)	16	RW	See description.
12D8h	TCD Last Destination Address Adjustment/Scatter Gather Address (TCD22_DLASTSGA)	32	RW	See description.
12DCh	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD22_BITER_ELINKNO)	16	RW	See description.
12DCh	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD22_BITER_ELINKYES)	16	RW	See description.
12DEh	TCD Control and Status (TCD22_CSR)	16	RW	See description.
12E0h	TCD Source Address (TCD23_SADDR)	32	RW	See description.
12E4h	TCD Transfer Attributes (TCD23_ATTR)	16	RW	See description.
12E6h	TCD Signed Source Address Offset (TCD23_SOFF)	16	RW	See description.
12E8h	TCD Minor Byte Count (Minor Loop Mapping Disabled) (TCD23_NBYTES_MLNO)	32	RW	See description.
12E8h	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (TCD23_NBYTES_MLOFFNO)	32	RW	See description.

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
12E8h	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (TCD23_NBYTES_MLOFFYES)	32	RW	See description.
12ECh	TCD Last Source Address Adjustment (TCD23_SLAST)	32	RW	See description.
12F0h	TCD Destination Address (TCD23_DADDR)	32	RW	See description.
12F4h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD23_CITER_ELINKNO)	16	RW	See description.
12F4h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD23_CITER_ELINKYES)	16	RW	See description.
12F6h	TCD Signed Destination Address Offset (TCD23_DOFF)	16	RW	See description.
12F8h	TCD Last Destination Address Adjustment/Scatter Gather Address (TCD23_DLASTSGA)	32	RW	See description.
12FCh	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD23_BITER_ELINKNO)	16	RW	See description.
12FCh	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD23_BITER_ELINKYES)	16	RW	See description.
12FEh	TCD Control and Status (TCD23_CSR)	16	RW	See description.
1300h	TCD Source Address (TCD24_SADDR)	32	RW	See description.
1304h	TCD Transfer Attributes (TCD24_ATTR)	16	RW	See description.
1306h	TCD Signed Source Address Offset (TCD24_SOFF)	16	RW	See description.
1308h	TCD Minor Byte Count (Minor Loop Mapping Disabled) (TCD24_NBYTES_MLNO)	32	RW	See description.
1308h	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (TCD24_NBYTES_MLOFFNO)	32	RW	See description.
1308h	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (TCD24_NBYTES_MLOFFYES)	32	RW	See description.
130Ch	TCD Last Source Address Adjustment (TCD24_SLAST)	32	RW	See description.
1310h	TCD Destination Address (TCD24_DADDR)	32	RW	See description.
1314h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD24_CITER_ELINKNO)	16	RW	See description.
1314h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD24_CITER_ELINKYES)	16	RW	See description.
1316h	TCD Signed Destination Address Offset (TCD24_DOFF)	16	RW	See description.
1318h	TCD Last Destination Address Adjustment/Scatter Gather Address (TCD24_DLASTSGA)	32	RW	See description.
131Ch	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD24_BITER_ELINKNO)	16	RW	See description.

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
131Ch	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD24_BITER_ELINKYES)	16	RW	See description.
131Eh	TCD Control and Status (TCD24_CSR)	16	RW	See description.
1320h	TCD Source Address (TCD25_SADDR)	32	RW	See description.
1324h	TCD Transfer Attributes (TCD25_ATTR)	16	RW	See description.
1326h	TCD Signed Source Address Offset (TCD25_SOFF)	16	RW	See description.
1328h	TCD Minor Byte Count (Minor Loop Mapping Disabled) (TCD25_NBYTES_MLNO)	32	RW	See description.
1328h	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (TCD25_NBYTES_MLOFFNO)	32	RW	See description.
1328h	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (TCD25_NBYTES_MLOFFYES)	32	RW	See description.
132Ch	TCD Last Source Address Adjustment (TCD25_SLAST)	32	RW	See description.
1330h	TCD Destination Address (TCD25_DADDR)	32	RW	See description.
1334h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD25_CITER_ELINKNO)	16	RW	See description.
1334h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD25_CITER_ELINKYES)	16	RW	See description.
1336h	TCD Signed Destination Address Offset (TCD25_DOFF)	16	RW	See description.
1338h	TCD Last Destination Address Adjustment/Scatter Gather Address (TCD25_DLASTSGA)	32	RW	See description.
133Ch	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD25_BITER_ELINKNO)	16	RW	See description.
133Ch	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD25_BITER_ELINKYES)	16	RW	See description.
133Eh	TCD Control and Status (TCD25_CSR)	16	RW	See description.
1340h	TCD Source Address (TCD26_SADDR)	32	RW	See description.
1344h	TCD Transfer Attributes (TCD26_ATTR)	16	RW	See description.
1346h	TCD Signed Source Address Offset (TCD26_SOFF)	16	RW	See description.
1348h	TCD Minor Byte Count (Minor Loop Mapping Disabled) (TCD26_NBYTES_MLNO)	32	RW	See description.
1348h	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (TCD26_NBYTES_MLOFFNO)	32	RW	See description.
1348h	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (TCD26_NBYTES_MLOFFYES)	32	RW	See description.

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
134Ch	TCD Last Source Address Adjustment (TCD26_SLAST)	32	RW	See description.
1350h	TCD Destination Address (TCD26_DADDR)	32	RW	See description.
1354h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD26_CITER_ELINKNO)	16	RW	See description.
1354h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD26_CITER_ELINKYES)	16	RW	See description.
1356h	TCD Signed Destination Address Offset (TCD26_DOFF)	16	RW	See description.
1358h	TCD Last Destination Address Adjustment/Scatter Gather Address (TCD26_DLASTSGA)	32	RW	See description.
135Ch	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD26_BITER_ELINKNO)	16	RW	See description.
135Ch	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD26_BITER_ELINKYES)	16	RW	See description.
135Eh	TCD Control and Status (TCD26_CSR)	16	RW	See description.
1360h	TCD Source Address (TCD27_SADDR)	32	RW	See description.
1364h	TCD Transfer Attributes (TCD27_ATTR)	16	RW	See description.
1366h	TCD Signed Source Address Offset (TCD27_SOFF)	16	RW	See description.
1368h	TCD Minor Byte Count (Minor Loop Mapping Disabled) (TCD27_NBYTES_MLNO)	32	RW	See description.
1368h	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (TCD27_NBYTES_MLOFFNO)	32	RW	See description.
1368h	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (TCD27_NBYTES_MLOFFYES)	32	RW	See description.
136Ch	TCD Last Source Address Adjustment (TCD27_SLAST)	32	RW	See description.
1370h	TCD Destination Address (TCD27_DADDR)	32	RW	See description.
1374h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD27_CITER_ELINKNO)	16	RW	See description.
1374h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD27_CITER_ELINKYES)	16	RW	See description.
1376h	TCD Signed Destination Address Offset (TCD27_DOFF)	16	RW	See description.
1378h	TCD Last Destination Address Adjustment/Scatter Gather Address (TCD27_DLASTSGA)	32	RW	See description.
137Ch	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD27_BITER_ELINKNO)	16	RW	See description.
137Ch	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD27_BITER_ELINKYES)	16	RW	See description.

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
137Eh	TCD Control and Status (TCD27_CSR)	16	RW	See description.
1380h	TCD Source Address (TCD28_SADDR)	32	RW	See description.
1384h	TCD Transfer Attributes (TCD28_ATTR)	16	RW	See description.
1386h	TCD Signed Source Address Offset (TCD28_SOFF)	16	RW	See description.
1388h	TCD Minor Byte Count (Minor Loop Mapping Disabled) (TCD28_NBYTES_MLNO)	32	RW	See description.
1388h	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (TCD28_NBYTES_MLOFFNO)	32	RW	See description.
1388h	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (TCD28_NBYTES_MLOFFYES)	32	RW	See description.
138Ch	TCD Last Source Address Adjustment (TCD28_SLAST)	32	RW	See description.
1390h	TCD Destination Address (TCD28_DADDR)	32	RW	See description.
1394h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD28_CITER_ELINKNO)	16	RW	See description.
1394h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD28_CITER_ELINKYES)	16	RW	See description.
1396h	TCD Signed Destination Address Offset (TCD28_DOFF)	16	RW	See description.
1398h	TCD Last Destination Address Adjustment/Scatter Gather Address (TCD28_DLASTSGA)	32	RW	See description.
139Ch	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD28_BITER_ELINKNO)	16	RW	See description.
139Ch	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD28_BITER_ELINKYES)	16	RW	See description.
139Eh	TCD Control and Status (TCD28_CSR)	16	RW	See description.
13A0h	TCD Source Address (TCD29_SADDR)	32	RW	See description.
13A4h	TCD Transfer Attributes (TCD29_ATTR)	16	RW	See description.
13A6h	TCD Signed Source Address Offset (TCD29_SOFF)	16	RW	See description.
13A8h	TCD Minor Byte Count (Minor Loop Mapping Disabled) (TCD29_NBYTES_MLNO)	32	RW	See description.
13A8h	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (TCD29_NBYTES_MLOFFNO)	32	RW	See description.
13A8h	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (TCD29_NBYTES_MLOFFYES)	32	RW	See description.
13ACh	TCD Last Source Address Adjustment (TCD29_SLAST)	32	RW	See description.

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
13B0h	TCD Destination Address (TCD29_DADDR)	32	RW	See description.
13B4h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD29_CITER_ELINKNO)	16	RW	See description.
13B4h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD29_CITER_ELINKYES)	16	RW	See description.
13B6h	TCD Signed Destination Address Offset (TCD29_DOFF)	16	RW	See description.
13B8h	TCD Last Destination Address Adjustment/Scatter Gather Address (TCD29_DLASTSGA)	32	RW	See description.
13BCh	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD29_BITER_ELINKNO)	16	RW	See description.
13BCh	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD29_BITER_ELINKYES)	16	RW	See description.
13BEh	TCD Control and Status (TCD29_CSR)	16	RW	See description.
13C0h	TCD Source Address (TCD30_SADDR)	32	RW	See description.
13C4h	TCD Transfer Attributes (TCD30_ATTR)	16	RW	See description.
13C6h	TCD Signed Source Address Offset (TCD30_SOFF)	16	RW	See description.
13C8h	TCD Minor Byte Count (Minor Loop Mapping Disabled) (TCD30_NBYTES_MLNO)	32	RW	See description.
13C8h	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (TCD30_NBYTES_MLOFFNO)	32	RW	See description.
13C8h	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (TCD30_NBYTES_MLOFFYES)	32	RW	See description.
13CCh	TCD Last Source Address Adjustment (TCD30_SLAST)	32	RW	See description.
13D0h	TCD Destination Address (TCD30_DADDR)	32	RW	See description.
13D4h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD30_CITER_ELINKNO)	16	RW	See description.
13D4h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD30_CITER_ELINKYES)	16	RW	See description.
13D6h	TCD Signed Destination Address Offset (TCD30_DOFF)	16	RW	See description.
13D8h	TCD Last Destination Address Adjustment/Scatter Gather Address (TCD30_DLASTSGA)	32	RW	See description.
13DCh	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD30_BITER_ELINKNO)	16	RW	See description.
13DCh	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD30_BITER_ELINKYES)	16	RW	See description.
13DEh	TCD Control and Status (TCD30_CSR)	16	RW	See description.

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
13E0h	TCD Source Address (TCD31_SADDR)	32	RW	See description.
13E4h	TCD Transfer Attributes (TCD31_ATTR)	16	RW	See description.
13E6h	TCD Signed Source Address Offset (TCD31_SOFF)	16	RW	See description.
13E8h	TCD Minor Byte Count (Minor Loop Mapping Disabled) (TCD31_NBYTES_MLNO)	32	RW	See description.
13E8h	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (TCD31_NBYTES_MLOFFNO)	32	RW	See description.
13E8h	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (TCD31_NBYTES_MLOFFYES)	32	RW	See description.
13ECh	TCD Last Source Address Adjustment (TCD31_SLAST)	32	RW	See description.
13F0h	TCD Destination Address (TCD31_DADDR)	32	RW	See description.
13F4h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD31_CITER_ELINKNO)	16	RW	See description.
13F4h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD31_CITER_ELINKYES)	16	RW	See description.
13F6h	TCD Signed Destination Address Offset (TCD31_DOFF)	16	RW	See description.
13F8h	TCD Last Destination Address Adjustment/Scatter Gather Address (TCD31_DLASTSGA)	32	RW	See description.
13FCh	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD31_BITER_ELINKNO)	16	RW	See description.
13FCh	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD31_BITER_ELINKYES)	16	RW	See description.
13FEh	TCD Control and Status (TCD31_CSR)	16	RW	See description.

17.4.6.2 Control Register (CR)

17.4.6.2.1 Offset

Register	Offset
CR	0h

17.4.6.2.2 Function

The CR defines the basic operating configuration of the DMA. The DMA arbitrates channel service requests in two groups of 16 channels each:

- Group 1 contains channels 31-16
- Group 0 contains channels 15-0

Arbitration within a group can be configured to use either a fixed-priority or a round-robin scheme. For fixed-priority arbitration, the highest priority channel requesting service is selected to execute. The channel priority registers assign the priorities; see the DCHPRIn registers. For round-robin arbitration, the channel priorities are ignored and channels within each group are cycled through (from high to low channel number) without regard to priority.

NOTE

For correct operation, writes to the CR register must be performed only when the DMA channels are inactive; that is, when TCDn_CSR[ACTIVE] bits are cleared.

The group priorities operate in a similar fashion. In group fixed priority arbitration mode, channel service requests in the highest priority group are executed first, where priority level 1 is the highest and priority level 0 is the lowest. The group priorities are assigned in the GRPnPRI fields of the DMA Control Register (CR). All group priorities must have unique values prior to any channel service requests occurring; otherwise, a configuration error will be reported. For group round robin arbitration, the group priorities are ignored and the groups are cycled through (from high to low group number) without regard to priority.

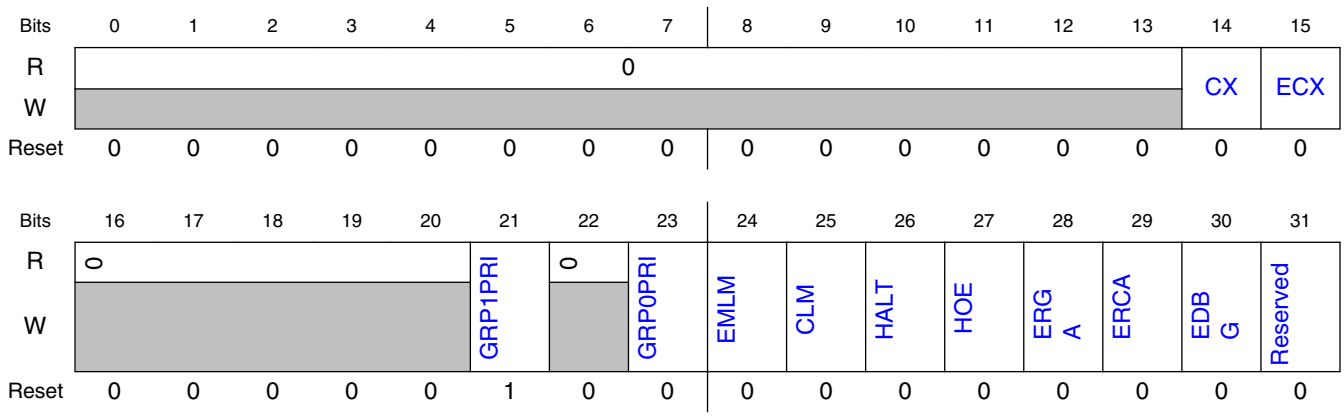
Minor loop offsets are address offset values added to the final source address (TCDn_SADDR) or destination address (TCDn_DADDR) upon minor loop completion. When minor loop offsets are enabled, the minor loop offset (MLOFF) is added to the final source address (TCDn_SADDR), to the final destination address (TCDn_DADDR), or to both prior to the addresses being written back into the TCD. If the major loop is complete, the minor loop offset is ignored and the major loop address offsets (TCDn_SLAST and TCDn_DLAST_SGA) are used to compute the next TCDn_SADDR and TCDn_DADDR values.

When minor loop mapping is enabled (EMLM is 1), TCDn word2 is redefined. A portion of TCDn word2 is used to specify multiple fields: a source enable bit (SMLOE) to specify the minor loop offset should be applied to the source address (TCDn_SADDR) upon minor loop completion, a destination enable bit (DMLOE) to specify the minor loop offset should be applied to the destination address (TCDn_DADDR) upon minor loop completion, and the sign extended minor loop offset value (MLOFF). The same offset value (MLOFF) is used for both source and destination minor loop offsets. When either

minor loop offset is enabled (SMLOE set or DMLOE set), the NBYTES field is reduced to 10 bits. When both minor loop offsets are disabled (SMLOE cleared and DMLOE cleared), the NBYTES field is a 30-bit vector.

When minor loop mapping is disabled (EMLM is 0), all 32 bits of TCDn word2 are assigned to the NBYTES field.

17.4.6.2.3 Diagram



17.4.6.2.4 Fields

Field	Function
0-13 —	Reserved
14 CX	Cancel Transfer 0b - Normal operation 1b - Cancel the remaining data transfer. Stop the executing channel and force the minor loop to finish. The cancel takes effect after the last write of the current read/write sequence. The CX bit clears itself after the cancel has been honored. This cancel retires the channel normally as if the minor loop was completed.
15 ECX	Error Cancel Transfer 0b - Normal operation 1b - Cancel the remaining data transfer in the same fashion as the CX bit. Stop the executing channel and force the minor loop to finish. The cancel takes effect after the last write of the current read/write sequence. The ECX bit clears itself after the cancel is honored. In addition to cancelling the transfer, ECX treats the cancel as an error condition, thus updating the Error Status register (DMAx_ES) and generating an optional error interrupt.
16-20 —	Reserved
21 GRP1PRI	Channel Group 1 Priority Group 1 priority level when fixed priority group arbitration is enabled.
22 —	Reserved

Table continues on the next page...

Field	Function
23 GRP0PRI	Channel Group 0 Priority Group 0 priority level when fixed priority group arbitration is enabled.
24 EMLM	Enable Minor Loop Mapping 0b - Disabled. TCDn.word2 is defined as a 32-bit NBYTES field. 1b - Enabled. TCDn.word2 is redefined to include individual enable fields, an offset field, and the NBYTES field. The individual enable fields allow the minor loop offset to be applied to the source address, the destination address, or both. The NBYTES field is reduced when either offset is enabled.
25 CLM	Continuous Link Mode **NOTE:** Do not use continuous link mode with a channel linking to itself if there is only one minor loop iteration per service request, for example, if the channel's NBYTES value is the same as either the source or destination size. The same data transfer profile can be achieved by simply increasing the NBYTES value, which provides more efficient, faster processing. 0b - A minor loop channel link made to itself goes through channel arbitration before being activated again. 1b - A minor loop channel link made to itself does not go through channel arbitration before being activated again. Upon minor loop completion, the channel activates again if that channel has a minor loop channel link enabled and the link channel is itself. This effectively applies the minor loop offsets and restarts the next minor loop.
26 HALT	Halt DMA Operations 0b - Normal operation 1b - Stall the start of any new channels. Executing channels are allowed to complete. Channel execution resumes when this bit is cleared.
27 HOE	Halt On Error 0b - Normal operation 1b - Any error causes the HALT bit to set. Subsequently, all service requests are ignored until the HALT bit is cleared.
28 ERGA	Enable Round Robin Group Arbitration 0b - Fixed priority arbitration is used for selection among the groups. 1b - Round robin arbitration is used for selection among the groups.
29 ERCA	Enable Round Robin Channel Arbitration 0b - Fixed priority arbitration is used for channel selection within each group. 1b - Round robin arbitration is used for channel selection within each group.
30 EDBG	Enable Debug 0b - When in debug mode, the DMA continues to operate. 1b - When in debug mode, the DMA stalls the start of a new channel. Executing channels are allowed to complete. Channel execution resumes when the system exits debug mode or the EDBG bit is cleared.
31 —	Reserved Reserved

17.4.6.3 Error Status Register (ES)

17.4.6.3.1 Offset

Register	Offset
ES	4h

17.4.6.3.2 Function

The ES provides information concerning the last recorded channel error. Channel errors can be caused by:

- A configuration error, that is:
  - An illegal setting in the transfer-control descriptor, or
  - An illegal priority register setting in fixed-arbitration
- An error termination to a bus master read or write cycle
- A cancel transfer with error bit that will be set when a transfer is canceled via the corresponding cancel transfer control bit

See Fault reporting and handling for more details.

17.4.6.3.3 Diagram

Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VLD	0														ECX
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	GPE	CPE	0	ERRCHN					SAE	SOE	DAE	DOE	NCE	SGE	SBE	DBE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

17.4.6.3.4 Fields

Field	Function
0 VLD	VLD Logical OR of all ERR status bits 0b - No ERR bits are set. 1b - At least one ERR bit is set indicating a valid error exists that has not been cleared.
1-14 —	Reserved

Table continues on the next page...

Field	Function
15 ECX	Transfer Canceled 0b - No canceled transfers 1b - The last recorded entry was a canceled transfer by the error cancel transfer input
16 GPE	Group Priority Error 0b - No group priority error 1b - The last recorded error was a configuration error among the group priorities. All group priorities are not unique.
17 CPE	Channel Priority Error 0b - No channel priority error 1b - The last recorded error was a configuration error in the channel priorities within a group. Channel priorities within a group are not unique.
18 —	Reserved
19-23 ERRCHN	Error Channel Number or Canceled Channel Number The channel number of the last recorded error, excluding GPE and CPE errors, or last recorded error canceled transfer.
24 SAE	Source Address Error 0b - No source address configuration error. 1b - The last recorded error was a configuration error detected in the TCDn_SADDR field. TCDn_SADDR is inconsistent with TCDn_ATTR[SSIZE].
25 SOE	Source Offset Error 0b - No source offset configuration error 1b - The last recorded error was a configuration error detected in the TCDn_SOFF field. TCDn_SOFF is inconsistent with TCDn_ATTR[SSIZE].
26 DAE	Destination Address Error 0b - No destination address configuration error 1b - The last recorded error was a configuration error detected in the TCDn_DADDR field. TCDn_DADDR is inconsistent with TCDn_ATTR[DSIZE].
27 DOE	Destination Offset Error 0b - No destination offset configuration error 1b - The last recorded error was a configuration error detected in the TCDn_DOFF field. TCDn_DOFF is inconsistent with TCDn_ATTR[DSIZE].
28 NCE	NBYTES/CITER Configuration Error 0b - No NBYTES/CITER configuration error 1b - The last recorded error was a configuration error detected in the TCDn_NBYTES or TCDn_CITER fields. TCDn_NBYTES is not a multiple of TCDn_ATTR[SSIZE] and TCDn_ATTR[DSIZE], orTCDn_CITER[CITER] is equal to zero, orTCDn_CITER[ELINK] is not equal to TCDn_BITER[ELINK]
29 SGE	Scatter/Gather Configuration Error 0b - No scatter/gather configuration error 1b - The last recorded error was a configuration error detected in the TCDn_DLASTSGA field. This field is checked at the beginning of a scatter/gather operation after major loop completion if TCDn_CSR[ESG] is enabled. TCDn_DLASTSGA is not on a 32 byte boundary.
30 SBE	Source Bus Error 0b - No source bus error 1b - The last recorded error was a bus error on a source read
31 DBE	Destination Bus Error 0b - No destination bus error 1b - The last recorded error was a bus error on a destination write

17.4.6.4 Enable Request Register (ERQ)

17.4.6.4.1 Offset

Register	Offset
ERQ	Ch

17.4.6.4.2 Function

The ERQ register provides a bit map for the 32 channels to enable the request signal for each channel. The state of any given channel enable is directly affected by writes to this register; it is also affected by writes to the SERQ and CERQ registers. These registers are provided so the request enable for a single channel can easily be modified without needing to perform a read-modify-write sequence to this register.

DMA request input signals and this enable request flag must be asserted before a channel's hardware service request is accepted. The state of the DMA enable request flag does not affect a channel service request made explicitly through software or a linked channel request.

NOTE

Disable a channel's hardware service request at the source before clearing the channel's ERQ bit.

17.4.6.4.3 Diagram

Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ERQ31	ERQ30	ERQ29	ERQ28	ERQ27	ERQ26	ERQ25	ERQ24	ERQ23	ERQ22	ERQ21	ERQ20	ERQ19	ERQ18	ERQ17	ERQ16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	ERQ15	ERQ14	ERQ13	ERQ12	ERQ11	ERQ10	ERQ9	ERQ8	ERQ7	ERQ6	ERQ5	ERQ4	ERQ3	ERQ2	ERQ1	ERQ0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

17.4.6.4.4 Fields

Field	Function
0 ERQ31	Enable DMA Request 31 0b - The DMA request signal for the corresponding channel is disabled 1b - The DMA request signal for the corresponding channel is enabled
1 ERQ30	Enable DMA Request 30 0b - The DMA request signal for the corresponding channel is disabled 1b - The DMA request signal for the corresponding channel is enabled
2 ERQ29	Enable DMA Request 29 0b - The DMA request signal for the corresponding channel is disabled 1b - The DMA request signal for the corresponding channel is enabled
3 ERQ28	Enable DMA Request 28 0b - The DMA request signal for the corresponding channel is disabled 1b - The DMA request signal for the corresponding channel is enabled
4 ERQ27	Enable DMA Request 27 0b - The DMA request signal for the corresponding channel is disabled 1b - The DMA request signal for the corresponding channel is enabled
5 ERQ26	Enable DMA Request 26 0b - The DMA request signal for the corresponding channel is disabled 1b - The DMA request signal for the corresponding channel is enabled
6 ERQ25	Enable DMA Request 25 0b - The DMA request signal for the corresponding channel is disabled 1b - The DMA request signal for the corresponding channel is enabled
7 ERQ24	Enable DMA Request 24 0b - The DMA request signal for the corresponding channel is disabled 1b - The DMA request signal for the corresponding channel is enabled
8 ERQ23	Enable DMA Request 23 0b - The DMA request signal for the corresponding channel is disabled 1b - The DMA request signal for the corresponding channel is enabled
9 ERQ22	Enable DMA Request 22 0b - The DMA request signal for the corresponding channel is disabled 1b - The DMA request signal for the corresponding channel is enabled
10 ERQ21	Enable DMA Request 21 0b - The DMA request signal for the corresponding channel is disabled 1b - The DMA request signal for the corresponding channel is enabled
11 ERQ20	Enable DMA Request 20 0b - The DMA request signal for the corresponding channel is disabled 1b - The DMA request signal for the corresponding channel is enabled
12 ERQ19	Enable DMA Request 19 0b - The DMA request signal for the corresponding channel is disabled 1b - The DMA request signal for the corresponding channel is enabled
13 ERQ18	Enable DMA Request 18 0b - The DMA request signal for the corresponding channel is disabled 1b - The DMA request signal for the corresponding channel is enabled
14 ERQ17	Enable DMA Request 17 0b - The DMA request signal for the corresponding channel is disabled 1b - The DMA request signal for the corresponding channel is enabled
15 ERQ16	Enable DMA Request 16 0b - The DMA request signal for the corresponding channel is disabled 1b - The DMA request signal for the corresponding channel is enabled

Table continues on the next page...

Field	Function
16 ERQ15	Enable DMA Request 15 0b - The DMA request signal for the corresponding channel is disabled 1b - The DMA request signal for the corresponding channel is enabled
17 ERQ14	Enable DMA Request 14 0b - The DMA request signal for the corresponding channel is disabled 1b - The DMA request signal for the corresponding channel is enabled
18 ERQ13	Enable DMA Request 13 0b - The DMA request signal for the corresponding channel is disabled 1b - The DMA request signal for the corresponding channel is enabled
19 ERQ12	Enable DMA Request 12 0b - The DMA request signal for the corresponding channel is disabled 1b - The DMA request signal for the corresponding channel is enabled
20 ERQ11	Enable DMA Request 11 0b - The DMA request signal for the corresponding channel is disabled 1b - The DMA request signal for the corresponding channel is enabled
21 ERQ10	Enable DMA Request 10 0b - The DMA request signal for the corresponding channel is disabled 1b - The DMA request signal for the corresponding channel is enabled
22 ERQ9	Enable DMA Request 9 0b - The DMA request signal for the corresponding channel is disabled 1b - The DMA request signal for the corresponding channel is enabled
23 ERQ8	Enable DMA Request 8 0b - The DMA request signal for the corresponding channel is disabled 1b - The DMA request signal for the corresponding channel is enabled
24 ERQ7	Enable DMA Request 7 0b - The DMA request signal for the corresponding channel is disabled 1b - The DMA request signal for the corresponding channel is enabled
25 ERQ6	Enable DMA Request 6 0b - The DMA request signal for the corresponding channel is disabled 1b - The DMA request signal for the corresponding channel is enabled
26 ERQ5	Enable DMA Request 5 0b - The DMA request signal for the corresponding channel is disabled 1b - The DMA request signal for the corresponding channel is enabled
27 ERQ4	Enable DMA Request 4 0b - The DMA request signal for the corresponding channel is disabled 1b - The DMA request signal for the corresponding channel is enabled
28 ERQ3	Enable DMA Request 3 0b - The DMA request signal for the corresponding channel is disabled 1b - The DMA request signal for the corresponding channel is enabled
29 ERQ2	Enable DMA Request 2 0b - The DMA request signal for the corresponding channel is disabled 1b - The DMA request signal for the corresponding channel is enabled
30 ERQ1	Enable DMA Request 1 0b - The DMA request signal for the corresponding channel is disabled 1b - The DMA request signal for the corresponding channel is enabled
31 ERQ0	Enable DMA Request 0 0b - The DMA request signal for the corresponding channel is disabled 1b - The DMA request signal for the corresponding channel is enabled

17.4.6.5 Enable Error Interrupt Register (EEI)

17.4.6.5.1 Offset

Register	Offset
EEI	14h

17.4.6.5.2 Function

The EEI register provides a bit map for the 32 channels to enable the error interrupt signal for each channel. The state of any given channel's error interrupt enable is directly affected by writes to this register; it is also affected by writes to the SEEI and CEEI. These registers are provided so that the error interrupt enable for a single channel can easily be modified without the need to perform a read-modify-write sequence to the EEI register.

The DMA error indicator and the error interrupt enable flag must be asserted before an error interrupt request for a given channel is asserted to the interrupt controller.

17.4.6.5.3 Diagram

Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	EEI31	EEI30	EEI29	EEI28	EEI27	EEI26	EEI25	EEI24	EEI23	EEI22	EEI21	EEI20	EEI19	EEI18	EEI17	EEI16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	EEI15	EEI14	EEI13	EEI12	EEI11	EEI10	EEI 9	EEI 8	EEI 7	EEI 6	EEI 5	EEI 4	EEI 3	EEI 2	EEI 1	EEI 0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

17.4.6.5.4 Fields

Field	Function
0 EEI31	Enable Error Interrupt 31 0b - The error signal for corresponding channel does not generate an error interrupt 1b - The assertion of the error signal for corresponding channel generates an error interrupt request
1	Enable Error Interrupt 30

Table continues on the next page...

Field	Function
EEI30	0b - The error signal for corresponding channel does not generate an error interrupt 1b - The assertion of the error signal for corresponding channel generates an error interrupt request
2 EEI29	Enable Error Interrupt 29 0b - The error signal for corresponding channel does not generate an error interrupt 1b - The assertion of the error signal for corresponding channel generates an error interrupt request
3 EEI28	Enable Error Interrupt 28 0b - The error signal for corresponding channel does not generate an error interrupt 1b - The assertion of the error signal for corresponding channel generates an error interrupt request
4 EEI27	Enable Error Interrupt 27 0b - The error signal for corresponding channel does not generate an error interrupt 1b - The assertion of the error signal for corresponding channel generates an error interrupt request
5 EEI26	Enable Error Interrupt 26 0b - The error signal for corresponding channel does not generate an error interrupt 1b - The assertion of the error signal for corresponding channel generates an error interrupt request
6 EEI25	Enable Error Interrupt 25 0b - The error signal for corresponding channel does not generate an error interrupt 1b - The assertion of the error signal for corresponding channel generates an error interrupt request
7 EEI24	Enable Error Interrupt 24 0b - The error signal for corresponding channel does not generate an error interrupt 1b - The assertion of the error signal for corresponding channel generates an error interrupt request
8 EEI23	Enable Error Interrupt 23 0b - The error signal for corresponding channel does not generate an error interrupt 1b - The assertion of the error signal for corresponding channel generates an error interrupt request
9 EEI22	Enable Error Interrupt 22 0b - The error signal for corresponding channel does not generate an error interrupt 1b - The assertion of the error signal for corresponding channel generates an error interrupt request
10 EEI21	Enable Error Interrupt 21 0b - The error signal for corresponding channel does not generate an error interrupt 1b - The assertion of the error signal for corresponding channel generates an error interrupt request
11 EEI20	Enable Error Interrupt 20 0b - The error signal for corresponding channel does not generate an error interrupt 1b - The assertion of the error signal for corresponding channel generates an error interrupt request
12 EEI19	Enable Error Interrupt 19 0b - The error signal for corresponding channel does not generate an error interrupt 1b - The assertion of the error signal for corresponding channel generates an error interrupt request
13 EEI18	Enable Error Interrupt 18 0b - The error signal for corresponding channel does not generate an error interrupt 1b - The assertion of the error signal for corresponding channel generates an error interrupt request
14 EEI17	Enable Error Interrupt 17 0b - The error signal for corresponding channel does not generate an error interrupt 1b - The assertion of the error signal for corresponding channel generates an error interrupt request
15 EEI16	Enable Error Interrupt 16 0b - The error signal for corresponding channel does not generate an error interrupt 1b - The assertion of the error signal for corresponding channel generates an error interrupt request
16 EEI15	Enable Error Interrupt 15 0b - The error signal for corresponding channel does not generate an error interrupt 1b - The assertion of the error signal for corresponding channel generates an error interrupt request
17 EEI14	Enable Error Interrupt 14 0b - The error signal for corresponding channel does not generate an error interrupt 1b - The assertion of the error signal for corresponding channel generates an error interrupt request

Table continues on the next page...

Field	Function
18 EEI13	Enable Error Interrupt 13 0b - The error signal for corresponding channel does not generate an error interrupt 1b - The assertion of the error signal for corresponding channel generates an error interrupt request
19 EEI12	Enable Error Interrupt 12 0b - The error signal for corresponding channel does not generate an error interrupt 1b - The assertion of the error signal for corresponding channel generates an error interrupt request
20 EEI11	Enable Error Interrupt 11 0b - The error signal for corresponding channel does not generate an error interrupt 1b - The assertion of the error signal for corresponding channel generates an error interrupt request
21 EEI10	Enable Error Interrupt 10 0b - The error signal for corresponding channel does not generate an error interrupt 1b - The assertion of the error signal for corresponding channel generates an error interrupt request
22 EEI9	Enable Error Interrupt 9 0b - The error signal for corresponding channel does not generate an error interrupt 1b - The assertion of the error signal for corresponding channel generates an error interrupt request
23 EEI8	Enable Error Interrupt 8 0b - The error signal for corresponding channel does not generate an error interrupt 1b - The assertion of the error signal for corresponding channel generates an error interrupt request
24 EEI7	Enable Error Interrupt 7 0b - The error signal for corresponding channel does not generate an error interrupt 1b - The assertion of the error signal for corresponding channel generates an error interrupt request
25 EEI6	Enable Error Interrupt 6 0b - The error signal for corresponding channel does not generate an error interrupt 1b - The assertion of the error signal for corresponding channel generates an error interrupt request
26 EEI5	Enable Error Interrupt 5 0b - The error signal for corresponding channel does not generate an error interrupt 1b - The assertion of the error signal for corresponding channel generates an error interrupt request
27 EEI4	Enable Error Interrupt 4 0b - The error signal for corresponding channel does not generate an error interrupt 1b - The assertion of the error signal for corresponding channel generates an error interrupt request
28 EEI3	Enable Error Interrupt 3 0b - The error signal for corresponding channel does not generate an error interrupt 1b - The assertion of the error signal for corresponding channel generates an error interrupt request
29 EEI2	Enable Error Interrupt 2 0b - The error signal for corresponding channel does not generate an error interrupt 1b - The assertion of the error signal for corresponding channel generates an error interrupt request
30 EEI1	Enable Error Interrupt 1 0b - The error signal for corresponding channel does not generate an error interrupt 1b - The assertion of the error signal for corresponding channel generates an error interrupt request
31 EEI0	Enable Error Interrupt 0 0b - The error signal for corresponding channel does not generate an error interrupt 1b - The assertion of the error signal for corresponding channel generates an error interrupt request

17.4.6.6 Set Enable Request Register (SERQ)

17.4.6.6.1 Offset

Register	Offset
SERQ	18h

17.4.6.6.2 Function

The SERQ provides a simple memory-mapped mechanism to set a given bit in the ERQ to enable the DMA request for a given channel. The data value on a register write causes the corresponding bit in the ERQ to be set. Setting the SAER bit provides a global set function, forcing the entire contents of ERQ to be set.

If the NOP bit is set, the command is ignored. This allows you to set a single, byte-wide register with a 32-bit write while not affecting the other registers addressed in the write. In such a case the other three bytes of the word would all have their NOP bit set so that that these register will not be affected by the write.

Reads of this register return all zeroes.

17.4.6.6.3 Diagram

Bits	0	1	2	3	4	5	6	7
R	0	0		0				
W	NOP	SAER	0	SERQ				
Reset	0	0	0	0	0	0	0	0

17.4.6.6.4 Fields

Field	Function
0 NOP	No Op enable 0b - Normal operation 1b - No operation, ignore the other bits in this register
1 SAER	Set All Enable Requests 0b - Set only the ERQ bit specified in the SERQ field 1b - Set all bits in ERQ
2 —	Reserved
3-7 SERQ	Set Enable Request Sets the corresponding bit in ERQ.

17.4.6.7 Clear Enable Request Register (CERQ)

17.4.6.7.1 Offset

Register	Offset
CERQ	19h

17.4.6.7.2 Function

The CERQ provides a simple memory-mapped mechanism to clear a given bit in the ERQ to disable the DMA request for a given channel. The data value on a register write causes the corresponding bit in the ERQ to be cleared. Setting the CAER bit provides a global clear function, forcing the entire contents of the ERQ to be cleared, disabling all DMA request inputs.

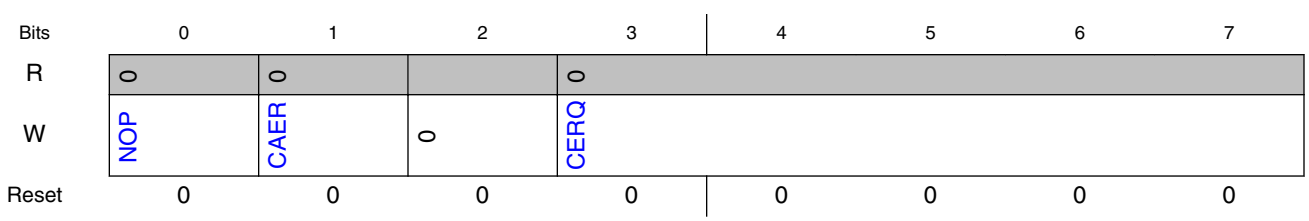
If the NOP bit is set, the command is ignored. This allows you to set a single, byte-wide register with a 32-bit write while not affecting the other registers addressed in the write. In such a case the other three bytes of the word would all have their NOP bit set so that that these register will not be affected by the write.

Reads of this register return all zeroes.

NOTE

Disable a channel's hardware service request at the source before clearing the channel's ERQ bit.

17.4.6.7.3 Diagram



17.4.6.7.4 Fields

Field	Function
0	No Op enable 0b - Normal operation

Table continues on the next page...

Field	Function
NOP	1b - No operation, ignore the other bits in this register
1 CAER	Clear All Enable Requests 0b - Clear only the ERQ bit specified in the CERQ field 1b - Clear all bits in ERQ
2 —	Reserved
3-7 CERQ	Clear Enable Request Clears the corresponding bit in ERQ.

17.4.6.8 Set Enable Error Interrupt Register (SEEI)

17.4.6.8.1 Offset

Register	Offset
SEEI	1Ah

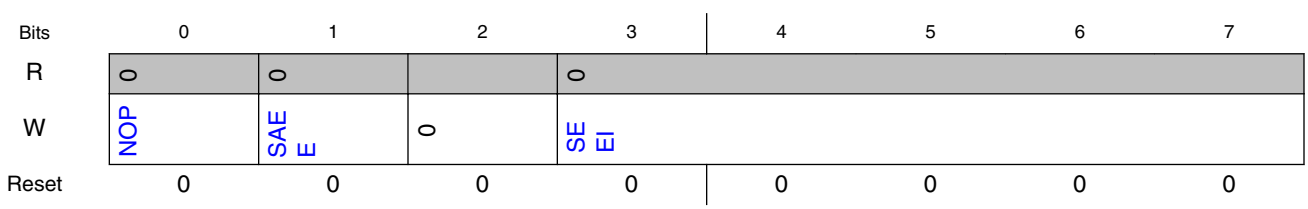
17.4.6.8.2 Function

The SEEI provides a simple memory-mapped mechanism to set a given bit in the EEI to enable the error interrupt for a given channel. The data value on a register write causes the corresponding bit in the EEI to be set. Setting the SAEE bit provides a global set function, forcing the entire EEI contents to be set.

If the NOP bit is set, the command is ignored. This allows you to set a single, byte-wide register with a 32-bit write while not affecting the other registers addressed in the write. In such a case the other three bytes of the word would all have their NOP bit set so that that these register will not be affected by the write.

Reads of this register return all zeroes.

17.4.6.8.3 Diagram



17.4.6.8.4 Fields

Field	Function
0 NOP	No Op enable 0b - Normal operation 1b - No operation, ignore the other bits in this register
1 SAEE	Sets All Enable Error Interrupts 0b - Set only the EEI bit specified in the SEEI field. 1b - Sets all bits in EEI
2 —	Reserved
3-7 SEEI	Set Enable Error Interrupt Sets the corresponding bit in EEI

17.4.6.9 Clear Enable Error Interrupt Register (CEEI)

17.4.6.9.1 Offset

Register	Offset
CEEI	1Bh

17.4.6.9.2 Function

The CEEI provides a simple memory-mapped mechanism to clear a given bit in the EEI to disable the error interrupt for a given channel. The data value on a register write causes the corresponding bit in the EEI to be cleared. Setting the CAEE bit provides a global clear function, forcing the EEI contents to be cleared, disabling all DMA request inputs.

If the NOP bit is set, the command is ignored. This allows you to set a single, byte-wide register with a 32-bit write while not affecting the other registers addressed in the write. In such a case the other three bytes of the word would all have their NOP bit set so that that these register will not be affected by the write.

Reads of this register return all zeroes.

17.4.6.9.3 Diagram

Bits	0	1	2	3	4	5	6	7
R	0	0		0				
W	NOP	CAEE	0	CEEI				
Reset	0	0	0	0	0	0	0	0

17.4.6.9.4 Fields

Field	Function
0 NOP	No Op enable 0b - Normal operation 1b - No operation, ignore the other bits in this register
1 CAEE	Clear All Enable Error Interrupts 0b - Clear only the EEI bit specified in the CEEI field 1b - Clear all bits in EEI
2 —	Reserved
3-7 CEEI	Clear Enable Error Interrupt Clears the corresponding bit in EEI

17.4.6.10 Clear Interrupt Request Register (CINT)

17.4.6.10.1 Offset

Register	Offset
CINT	1Ch

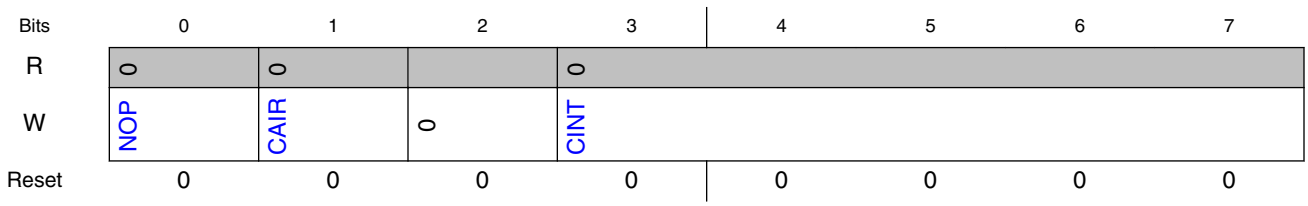
17.4.6.10.2 Function

The CINT provides a simple, memory-mapped mechanism to clear a given bit in the INT to disable the interrupt request for a given channel. The given value on a register write causes the corresponding bit in the INT to be cleared. Setting the CAIR bit provides a global clear function, forcing the entire contents of the INT to be cleared, disabling all DMA interrupt requests.

If the NOP bit is set, the command is ignored. This allows you to set a single, byte-wide register with a 32-bit write while not affecting the other registers addressed in the write. In such a case the other three bytes of the word would all have their NOP bit set so that that these register will not be affected by the write.

Reads of this register return all zeroes.

17.4.6.10.3 Diagram



17.4.6.10.4 Fields

Field	Function
0 NOP	No Op enable 0b - Normal operation 1b - No operation, ignore the other bits in this register
1 CAIR	Clear All Interrupt Requests 0b - Clear only the INT bit specified in the CINT field 1b - Clear all bits in INT
2 —	Reserved
3-7 CINT	Clear Interrupt Request Clears the corresponding bit in INT

17.4.6.11 Clear Error Register (CERR)

17.4.6.11.1 Offset

Register	Offset
CERR	1Dh

17.4.6.11.2 Function

The CERR provides a simple memory-mapped mechanism to clear a given bit in the ERR to disable the error condition flag for a given channel. The given value on a register write causes the corresponding bit in the ERR to be cleared. Setting the CAEI bit provides a global clear function, forcing the ERR contents to be cleared, clearing all channel error indicators. If the NOP bit is set, the command is ignored. This allows you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

17.4.6.11.3 Diagram

Bits	0	1	2	3	4	5	6	7
R	0	0		0				
W	NOP	CAEI	0	CERR				
Reset	0	0	0	0	0	0	0	0

17.4.6.11.4 Fields

Field	Function
0 NOP	No Op enable 0b - Normal operation 1b - No operation, ignore the other bits in this register
1 CAEI	Clear All Error Indicators 0b - Clear only the ERR bit specified in the CERR field 1b - Clear all bits in ERR
2 —	Reserved
3-7 CERR	Clear Error Indicator Clears the corresponding bit in ERR

17.4.6.12 Set START Bit Register (SSRT)

17.4.6.12.1 Offset

Register	Offset
SSRT	1Eh

17.4.6.12.2 Function

The SSRT provides a simple memory-mapped mechanism to set the START bit in the TCD of the given channel. The data value on a register write causes the START bit in the corresponding transfer control descriptor to be set. Setting the SAST bit provides a global set function, forcing all START bits to be set.

If the NOP bit is set, the command is ignored. This allows you to set a single, byte-wide register with a 32-bit write while not affecting the other registers addressed in the write. In such a case the other three bytes of the word would all have their NOP bit set so that that these register will not be affected by the write.

Reads of this register return all zeroes.

17.4.6.12.3 Diagram

Bits	0	1	2	3	4	5	6	7
R	0	0		0				
W	NOP	SAST	0	SSRT				
Reset	0	0	0	0	0	0	0	0

17.4.6.12.4 Fields

Field	Function
0 NOP	No Op enable 0b - Normal operation 1b - No operation, ignore the other bits in this register
1 SAST	Set All START Bits (activates all channels) 0b - Set only the TCDn_CSR[START] bit specified in the SSRT field 1b - Set all bits in TCDn_CSR[START]
2 —	Reserved
3-7 SSRT	Set START Bit Sets the corresponding bit in TCDn_CSR[START]

17.4.6.13 Clear DONE Status Bit Register (CDNE)

17.4.6.13.1 Offset

Register	Offset
CDNE	1Fh

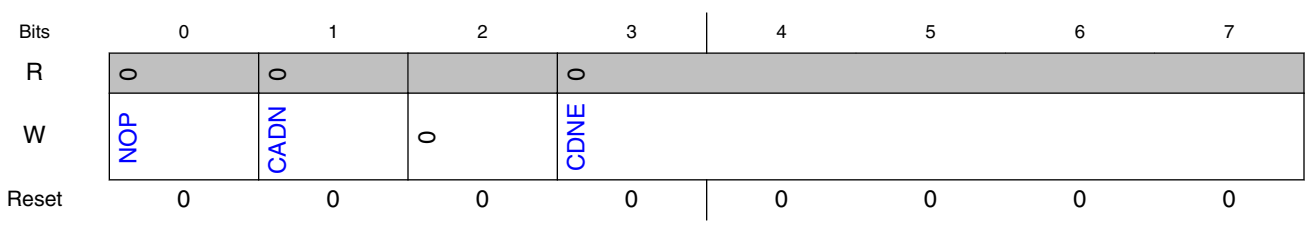
17.4.6.13.2 Function

The CDNE provides a simple memory-mapped mechanism to clear the DONE bit in the TCD of the given channel. The data value on a register write causes the DONE bit in the corresponding transfer control descriptor to be cleared. Setting the CADN bit provides a global clear function, forcing all DONE bits to be cleared.

If the NOP bit is set, the command is ignored. This allows you to set a single, byte-wide register with a 32-bit write while not affecting the other registers addressed in the write. In such a case the other three bytes of the word would all have their NOP bit set so that that these register will not be affected by the write.

Reads of this register return all zeroes.

17.4.6.13.3 Diagram



17.4.6.13.4 Fields

Field	Function
0 NOP	No Op enable 0b - Normal operation 1b - No operation, ignore the other bits in this register
1 CADN	Clears All DONE Bits 0b - Clears only the TCDn_CSR[DONE] bit specified in the CDNE field 1b - Clears all bits in TCDn_CSR[DONE]
2 —	Reserved
3-7 CDNE	Clear DONE Bit Clears the corresponding bit in TCDn_CSR[DONE]

17.4.6.14 Interrupt Request Register (INT)

17.4.6.14.1 Offset

Register	Offset
INT	24h

17.4.6.14.2 Function

The INT register provides a bit map for the 32 channels signaling the presence of an interrupt request for each channel. Depending on the appropriate bit setting in the transfer-control descriptors, the eDMA engine generates an interrupt on data transfer completion. The outputs of this register are directly routed to the interrupt controller. During the interrupt-service routine associated with any given channel, it is the software's responsibility to clear the appropriate bit, negating the interrupt request. Typically, a write to the CINT register in the interrupt service routine is used for this purpose.

The state of any given channel's interrupt request is directly affected by writes to this register; it is also affected by writes to the CINT register. On writes to INT, a 1 in any bit position clears the corresponding channel's interrupt request. A zero in any bit position has no affect on the corresponding channel's current interrupt status. The CINT register is provided so the interrupt request for a single channel can easily be cleared without the need to perform a read-modify-write sequence to the INT register.

17.4.6.14.3 Diagram

Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	INT31	INT30	INT29	INT28	INT27	INT26	INT25	INT24	INT23	INT22	INT21	INT20	INT19	INT18	INT17	INT16
W	W1C	W1C	W1C	W1C	W1C	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

Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	INT15	INT14	INT13	INT12	INT11	INT10	INT9	INT8	INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0
W	W1C	W1C	W1C	W1C	W1C	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

17.4.6.14.4 Fields

Field	Function
0 INT31	Interrupt Request 31 0b - The interrupt request for corresponding channel is cleared 1b - The interrupt request for corresponding channel is active
1 INT30	Interrupt Request 30 0b - The interrupt request for corresponding channel is cleared 1b - The interrupt request for corresponding channel is active
2 INT29	Interrupt Request 29 0b - The interrupt request for corresponding channel is cleared 1b - The interrupt request for corresponding channel is active
3 INT28	Interrupt Request 28 0b - The interrupt request for corresponding channel is cleared 1b - The interrupt request for corresponding channel is active
4 INT27	Interrupt Request 27 0b - The interrupt request for corresponding channel is cleared 1b - The interrupt request for corresponding channel is active
5 INT26	Interrupt Request 26 0b - The interrupt request for corresponding channel is cleared 1b - The interrupt request for corresponding channel is active
6 INT25	Interrupt Request 25 0b - The interrupt request for corresponding channel is cleared 1b - The interrupt request for corresponding channel is active
7 INT24	Interrupt Request 24 0b - The interrupt request for corresponding channel is cleared 1b - The interrupt request for corresponding channel is active
8 INT23	Interrupt Request 23 0b - The interrupt request for corresponding channel is cleared 1b - The interrupt request for corresponding channel is active
9 INT22	Interrupt Request 22 0b - The interrupt request for corresponding channel is cleared 1b - The interrupt request for corresponding channel is active
10 INT21	Interrupt Request 21 0b - The interrupt request for corresponding channel is cleared 1b - The interrupt request for corresponding channel is active
11 INT20	Interrupt Request 20 0b - The interrupt request for corresponding channel is cleared 1b - The interrupt request for corresponding channel is active
12 INT19	Interrupt Request 19 0b - The interrupt request for corresponding channel is cleared 1b - The interrupt request for corresponding channel is active
13 INT18	Interrupt Request 18 0b - The interrupt request for corresponding channel is cleared 1b - The interrupt request for corresponding channel is active
14 INT17	Interrupt Request 17 0b - The interrupt request for corresponding channel is cleared 1b - The interrupt request for corresponding channel is active
15	Interrupt Request 16 0b - The interrupt request for corresponding channel is cleared

Table continues on the next page...

Field	Function
INT16	1b - The interrupt request for corresponding channel is active
16 INT15	Interrupt Request 15 0b - The interrupt request for corresponding channel is cleared 1b - The interrupt request for corresponding channel is active
17 INT14	Interrupt Request 14 0b - The interrupt request for corresponding channel is cleared 1b - The interrupt request for corresponding channel is active
18 INT13	Interrupt Request 13 0b - The interrupt request for corresponding channel is cleared 1b - The interrupt request for corresponding channel is active
19 INT12	Interrupt Request 12 0b - The interrupt request for corresponding channel is cleared 1b - The interrupt request for corresponding channel is active
20 INT11	Interrupt Request 11 0b - The interrupt request for corresponding channel is cleared 1b - The interrupt request for corresponding channel is active
21 INT10	Interrupt Request 10 0b - The interrupt request for corresponding channel is cleared 1b - The interrupt request for corresponding channel is active
22 INT9	Interrupt Request 9 0b - The interrupt request for corresponding channel is cleared 1b - The interrupt request for corresponding channel is active
23 INT8	Interrupt Request 8 0b - The interrupt request for corresponding channel is cleared 1b - The interrupt request for corresponding channel is active
24 INT7	Interrupt Request 7 0b - The interrupt request for corresponding channel is cleared 1b - The interrupt request for corresponding channel is active
25 INT6	Interrupt Request 6 0b - The interrupt request for corresponding channel is cleared 1b - The interrupt request for corresponding channel is active
26 INT5	Interrupt Request 5 0b - The interrupt request for corresponding channel is cleared 1b - The interrupt request for corresponding channel is active
27 INT4	Interrupt Request 4 0b - The interrupt request for corresponding channel is cleared 1b - The interrupt request for corresponding channel is active
28 INT3	Interrupt Request 3 0b - The interrupt request for corresponding channel is cleared 1b - The interrupt request for corresponding channel is active
29 INT2	Interrupt Request 2 0b - The interrupt request for corresponding channel is cleared 1b - The interrupt request for corresponding channel is active
30 INT1	Interrupt Request 1 0b - The interrupt request for corresponding channel is cleared 1b - The interrupt request for corresponding channel is active
31 INT0	Interrupt Request 0 0b - The interrupt request for corresponding channel is cleared 1b - The interrupt request for corresponding channel is active

17.4.6.15 Error Register (ERR)

17.4.6.15.1 Offset

Register	Offset
ERR	2Ch

17.4.6.15.2 Function

The ERR register provides a bit map for the 32 channels, signaling the presence of an error for each channel. The eDMA engine signals the occurrence of an error condition by setting the appropriate bit in this register. The outputs of this register are enabled by the contents of the EEI register, then logically summed across groups of 16 and 32 channels to form several group error interrupt requests, which are then routed to the interrupt controller. During the execution of the interrupt-service routine associated with any DMA errors, it is software's responsibility to clear the appropriate bit, negating the error-interrupt request. Typically, a write to the CERR in the interrupt-service routine is used for this purpose. The normal DMA channel completion indicators (setting the transfer control descriptor DONE flag and the possible assertion of an interrupt request) are not affected when an error is detected.

The contents of this register can also be polled because a non-zero value indicates the presence of a channel error regardless of the state of the EEI fields. The state of any given channel's error indicators is affected by writes to this register; it is also affected by writes to the CERR. On writes to the ERR, a one in any bit position clears the corresponding channel's error status. A zero in any bit position has no affect on the corresponding channel's current error status. The CERR is provided so the error indicator for a single channel can easily be cleared.

17.4.6.15.3 Diagram

Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ERR31	ERR30	ERR29	ERR28	ERR27	ERR26	ERR25	ERR24	ERR23	ERR22	ERR21	ERR20	ERR19	ERR18	ERR17	ERR16
W	W1C	W1C	W1C	W1C	W1C	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

Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ERR15	ERR14	ERR13	ERR12	ERR11	ERR10	ERR9	ERR8	ERR7	ERR6	ERR5	ERR4	ERR3	ERR2	ERR1	ERR0
W	W1C	W1C	W1C	W1C	W1C	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

17.4.6.15.4 Fields

Field	Function
0 ERR31	Error In Channel 31 0b - An error in this channel has not occurred 1b - An error in this channel has occurred
1 ERR30	Error In Channel 30 0b - An error in this channel has not occurred 1b - An error in this channel has occurred
2 ERR29	Error In Channel 29 0b - An error in this channel has not occurred 1b - An error in this channel has occurred
3 ERR28	Error In Channel 28 0b - An error in this channel has not occurred 1b - An error in this channel has occurred
4 ERR27	Error In Channel 27 0b - An error in this channel has not occurred 1b - An error in this channel has occurred
5 ERR26	Error In Channel 26 0b - An error in this channel has not occurred 1b - An error in this channel has occurred
6 ERR25	Error In Channel 25 0b - An error in this channel has not occurred 1b - An error in this channel has occurred
7 ERR24	Error In Channel 24 0b - An error in this channel has not occurred 1b - An error in this channel has occurred
8 ERR23	Error In Channel 23 0b - An error in this channel has not occurred 1b - An error in this channel has occurred

Table continues on the next page...

Field	Function
9 ERR22	Error In Channel 22 0b - An error in this channel has not occurred 1b - An error in this channel has occurred
10 ERR21	Error In Channel 21 0b - An error in this channel has not occurred 1b - An error in this channel has occurred
11 ERR20	Error In Channel 20 0b - An error in this channel has not occurred 1b - An error in this channel has occurred
12 ERR19	Error In Channel 19 0b - An error in this channel has not occurred 1b - An error in this channel has occurred
13 ERR18	Error In Channel 18 0b - An error in this channel has not occurred 1b - An error in this channel has occurred
14 ERR17	Error In Channel 17 0b - An error in this channel has not occurred 1b - An error in this channel has occurred
15 ERR16	Error In Channel 16 0b - An error in this channel has not occurred 1b - An error in this channel has occurred
16 ERR15	Error In Channel 15 0b - An error in this channel has not occurred 1b - An error in this channel has occurred
17 ERR14	Error In Channel 14 0b - An error in this channel has not occurred 1b - An error in this channel has occurred
18 ERR13	Error In Channel 13 0b - An error in this channel has not occurred 1b - An error in this channel has occurred
19 ERR12	Error In Channel 12 0b - An error in this channel has not occurred 1b - An error in this channel has occurred
20 ERR11	Error In Channel 11 0b - An error in this channel has not occurred 1b - An error in this channel has occurred
21 ERR10	Error In Channel 10 0b - An error in this channel has not occurred 1b - An error in this channel has occurred
22 ERR9	Error In Channel 9 0b - An error in this channel has not occurred 1b - An error in this channel has occurred
23 ERR8	Error In Channel 8 0b - An error in this channel has not occurred 1b - An error in this channel has occurred
24 ERR7	Error In Channel 7 0b - An error in this channel has not occurred 1b - An error in this channel has occurred
25 ERR6	Error In Channel 6 0b - An error in this channel has not occurred

Table continues on the next page...

Field	Function
	1b - An error in this channel has occurred
26 ERR5	Error In Channel 5 0b - An error in this channel has not occurred 1b - An error in this channel has occurred
27 ERR4	Error In Channel 4 0b - An error in this channel has not occurred 1b - An error in this channel has occurred
28 ERR3	Error In Channel 3 0b - An error in this channel has not occurred 1b - An error in this channel has occurred
29 ERR2	Error In Channel 2 0b - An error in this channel has not occurred 1b - An error in this channel has occurred
30 ERR1	Error In Channel 1 0b - An error in this channel has not occurred 1b - An error in this channel has occurred
31 ERR0	Error In Channel 0 0b - An error in this channel has not occurred 1b - An error in this channel has occurred

17.4.6.16 Hardware Request Status Register (HRS)

17.4.6.16.1 Offset

Register	Offset
HRS	34h

17.4.6.16.2 Function

The HRS register provides a bit map for the DMA channels, signaling the presence of a hardware request for each channel. The hardware request status bits reflect the current state of the register and qualified (via the ERQ fields) DMA request signals as seen by the DMA's arbitration logic. This view into the hardware request signals may be used for debug purposes.

NOTE

These bits reflect the state of the request as seen by the arbitration logic. Therefore, this status is affected by the ERQ bits.

17.4.6.16.3 Diagram

Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	HRS31	HRS30	HRS29	HRS28	HRS27	HRS26	HRS25	HRS24	HRS23	HRS22	HRS21	HRS20	HRS19	HRS18	HRS17	HRS16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	HRS15	HRS14	HRS13	HRS12	HRS11	HRS10	HRS 9	HRS 8	HRS 7	HRS 6	HRS 5	HRS 4	HRS 3	HRS 2	HRS 1	HRS 0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

17.4.6.16.4 Fields

Field	Function
0 HRS31	Hardware Request Status Channel 31 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0b - A hardware service request for channel 31 is not present 1b - A hardware service request for channel 31 is present
1 HRS30	Hardware Request Status Channel 30 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0b - A hardware service request for channel 30 is not present 1b - A hardware service request for channel 30 is present
2 HRS29	Hardware Request Status Channel 29 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0b - A hardware service request for channel 29 is not preset 1b - A hardware service request for channel 29 is present
3 HRS28	Hardware Request Status Channel 28 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0b - A hardware service request for channel 28 is not present 1b - A hardware service request for channel 28 is present
4 HRS27	Hardware Request Status Channel 27 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0b - A hardware service request for channel 27 is not present

Table continues on the next page...

Field	Function
	1b - A hardware service request for channel 27 is present
5 HRS26	Hardware Request Status Channel 26 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0b - A hardware service request for channel 26 is not present 1b - A hardware service request for channel 26 is present
6 HRS25	Hardware Request Status Channel 25 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0b - A hardware service request for channel 25 is not present 1b - A hardware service request for channel 25 is present
7 HRS24	Hardware Request Status Channel 24 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0b - A hardware service request for channel 24 is not present 1b - A hardware service request for channel 24 is present
8 HRS23	Hardware Request Status Channel 23 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0b - A hardware service request for channel 23 is not present 1b - A hardware service request for channel 23 is present
9 HRS22	Hardware Request Status Channel 22 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0b - A hardware service request for channel 22 is not present 1b - A hardware service request for channel 22 is present
10 HRS21	Hardware Request Status Channel 21 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0b - A hardware service request for channel 21 is not present 1b - A hardware service request for channel 21 is present
11 HRS20	Hardware Request Status Channel 20 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0b - A hardware service request for channel 20 is not present 1b - A hardware service request for channel 20 is present
12 HRS19	Hardware Request Status Channel 19 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0b - A hardware service request for channel 19 is not present 1b - A hardware service request for channel 19 is present
13	Hardware Request Status Channel 18

Table continues on the next page...

Field	Function
HRS18	The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0b - A hardware service request for channel 18 is not present 1b - A hardware service request for channel 18 is present
14 HRS17	Hardware Request Status Channel 17 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0b - A hardware service request for channel 17 is not present 1b - A hardware service request for channel 17 is present
15 HRS16	Hardware Request Status Channel 16 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0b - A hardware service request for channel 16 is not present 1b - A hardware service request for channel 16 is present
16 HRS15	Hardware Request Status Channel 15 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0b - A hardware service request for channel 15 is not present 1b - A hardware service request for channel 15 is present
17 HRS14	Hardware Request Status Channel 14 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0b - A hardware service request for channel 14 is not present 1b - A hardware service request for channel 14 is present
18 HRS13	Hardware Request Status Channel 13 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0b - A hardware service request for channel 13 is not present 1b - A hardware service request for channel 13 is present
19 HRS12	Hardware Request Status Channel 12 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0b - A hardware service request for channel 12 is not present 1b - A hardware service request for channel 12 is present
20 HRS11	Hardware Request Status Channel 11 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0b - A hardware service request for channel 11 is not present 1b - A hardware service request for channel 11 is present
21 HRS10	Hardware Request Status Channel 10

Table continues on the next page...

Field	Function
	The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0b - A hardware service request for channel 10 is not present 1b - A hardware service request for channel 10 is present
22 HRS9	Hardware Request Status Channel 9 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0b - A hardware service request for channel 9 is not present 1b - A hardware service request for channel 9 is present
23 HRS8	Hardware Request Status Channel 8 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0b - A hardware service request for channel 8 is not present 1b - A hardware service request for channel 8 is present
24 HRS7	Hardware Request Status Channel 7 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0b - A hardware service request for channel 7 is not present 1b - A hardware service request for channel 7 is present
25 HRS6	Hardware Request Status Channel 6 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0b - A hardware service request for channel 6 is not present 1b - A hardware service request for channel 6 is present
26 HRS5	Hardware Request Status Channel 5 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0b - A hardware service request for channel 5 is not present 1b - A hardware service request for channel 5 is present
27 HRS4	Hardware Request Status Channel 4 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0b - A hardware service request for channel 4 is not present 1b - A hardware service request for channel 4 is present
28 HRS3	Hardware Request Status Channel 3 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0b - A hardware service request for channel 3 is not present 1b - A hardware service request for channel 3 is present
29 HRS2	Hardware Request Status Channel 2

Table continues on the next page...

Field	Function
	The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0b - A hardware service request for channel 2 is not present 1b - A hardware service request for channel 2 is present
30 HRS1	Hardware Request Status Channel 1 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0b - A hardware service request for channel 1 is not present 1b - A hardware service request for channel 1 is present
31 HRS0	Hardware Request Status Channel 0 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0b - A hardware service request for channel 0 is not present 1b - A hardware service request for channel 0 is present

17.4.6.17 Channel Priority Register (DCHPRI0 - DCHPRI31)

17.4.6.17.1 Offset

Register	Offset
DCHPRI3	100h
DCHPRI2	101h
DCHPRI1	102h
DCHPRI0	103h
DCHPRI7	104h
DCHPRI6	105h
DCHPRI5	106h
DCHPRI4	107h
DCHPRI11	108h
DCHPRI10	109h
DCHPRI9	10Ah
DCHPRI8	10Bh
DCHPRI15	10Ch
DCHPRI14	10Dh
DCHPRI13	10Eh
DCHPRI12	10Fh
DCHPRI19	110h
DCHPRI18	111h

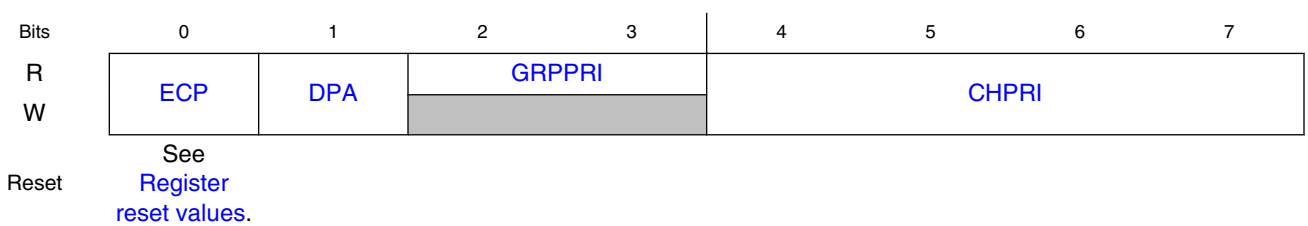
Table continues on the next page...

Register	Offset
DCHPRI17	112h
DCHPRI16	113h
DCHPRI23	114h
DCHPRI22	115h
DCHPRI21	116h
DCHPRI20	117h
DCHPRI27	118h
DCHPRI26	119h
DCHPRI25	11Ah
DCHPRI24	11Bh
DCHPRI31	11Ch
DCHPRI30	11Dh
DCHPRI29	11Eh
DCHPRI28	11Fh

17.4.6.17.2 Function

When fixed-priority channel arbitration is enabled (CR[ERCA] = 0), the contents of these registers define the unique priorities associated with each channel within a group. The channel priorities are evaluated by numeric value; for example, 0 is the lowest priority, 1 is the next higher priority, then 2, 3, etc. Software must program the channel priorities with unique values; otherwise, a configuration error is reported. The range of the priority value is limited to the values of 0 through 15. When read, the GRPPRI bits of the DCHPRIn register reflect the current priority level of the group of channels in which the corresponding channel resides. GRPPRI bits are not affected by writes to the DCHPRIn registers. The group priority is assigned in the DMA control register.

17.4.6.17.3 Diagram



17.4.6.17.4 Register reset values

Register	Reset value
DCHPRI0	00h
DCHPRI1	01h
DCHPRI2	02h
DCHPRI3	03h
DCHPRI4	04h
DCHPRI5	05h
DCHPRI6	06h
DCHPRI7	07h
DCHPRI8	08h
DCHPRI9	09h
DCHPRI10	0Ah
DCHPRI11	0Bh
DCHPRI12	0Ch
DCHPRI13	0Dh
DCHPRI14	0Eh
DCHPRI15	0Fh
DCHPRI16	10h
DCHPRI17	11h
DCHPRI18	12h
DCHPRI19	13h
DCHPRI20	14h
DCHPRI21	15h
DCHPRI22	16h
DCHPRI23	17h
DCHPRI24	18h
DCHPRI25	19h
DCHPRI26	1Ah
DCHPRI27	1Bh
DCHPRI28	1Ch
DCHPRI29	1Dh
DCHPRI30	1Eh
DCHPRI31	1Fh

17.4.6.17.5 Fields

Field	Function
0	Enable Channel Preemption. This field resets to 0. 0b - Channel n cannot be suspended by a higher priority channel's service request.

Table continues on the next page...

Field	Function
ECP	1b - Channel n can be temporarily suspended by the service request of a higher priority channel.
1 DPA	Disable Preempt Ability. This field resets to 0. 0b - Channel n can suspend a lower priority channel. 1b - Channel n cannot suspend any channel, regardless of channel priority.
2-3 GRPPRI	Channel n Current Group Priority Group priority assigned to this channel group when fixed-priority arbitration is enabled. This field is read-only; writes are ignored.
4-7 CHPRI	Channel n Arbitration Priority Channel priority when fixed-priority arbitration is enabled

17.4.6.18 Channel n Master ID Register (DCHMID0 - DCHMID31)

17.4.6.18.1 Offset

For n = 0 to 31:

Register	Offset
DCHMIDn	140h + (n × 1h)

17.4.6.18.2 Function

The DMA master ID replication registers allow the DMA to use the same protection level and system bus ID of the master programming the DMA's TCD. When enabled, the DMA uses the master ID and protection level stored in the DCHMID register instead of the DMA's default values. When a master, for example, a core, programs a TCD, its master ID and protection level are captured when the TCDn.CSR control attributes are written. Although the scatter/gather operation can change the contents of TCDn_CSR, that operation does not affect the DCHMID *n* registers.

17.4.6.18.3 Diagram

Bits	0	1	2	3	4	5	6	7
R	EMI	PAL	0		MID			
W								
Reset	0	0	0	0	0	0	0	0

17.4.6.18.4 Fields

Field	Function
0 EMI	Enable Master ID replication **NOTE:** If Master ID replication is disabled, the privileged protection level (supervisor mode) for DMA transfers is used. 0b - Master ID replication is disabled 1b - Master ID replication is enabled
1 PAL	Privileged Access Level **NOTE:** The protection level captured in this register reflects the level used when writing the channel's control attributes; lower byte of TCDn.CSR. 0b - User protection level for DMA transfers 1b - Privileged protection level for DMA transfers
2-3 —	Reserved
4-7 MID	Master ID DMA's master ID when channel n is active and master ID replication is enabled. **NOTE:** The master ID captured in this register reflects the ID used when writing the channel's control attributes; lower byte of TCDn.CSR.

17.4.6.19 TCD Source Address (TCD0_SADDR - TCD31_SADDR)

17.4.6.19.1 Offset

For n = 0 to 31:

Register	Offset
TCDn_SADDR	1000h + (n × 20h)

17.4.6.19.2 Function

This register contains the source address of the transfer.

17.4.6.19.3 Diagram

Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	SADDR															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	SADDR															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

17.4.6.19.4 Fields

Field	Function
0-31 SADDR	Source Address Memory address pointing to the source data.

17.4.6.20 TCD Transfer Attributes (TCD0_ATTR - TCD31_ATTR)

17.4.6.20.1 Offset

For n = 0 to 31:

Register	Offset
TCDn_ATTR	1004h + (n × 20h)

17.4.6.20.2 Diagram

Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	SMOD					SSIZE			DMOD					DSIZE		
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

17.4.6.20.3 Fields

Field	Function
0-4 SMOD	Source Address Modulo 00000b - Source address modulo feature is disabled 00001-11111b - This value defines a specific address range specified to be the value after SADDR + SOFF calculation is performed on the original register value. Setting this field provides the ability to implement a circular data queue easily. For data queues requiring power-of-2 size bytes, the queue should start at a 0-modulo-size address and the SMOD field should be set to the appropriate value for the queue, freezing the desired number of upper address bits. The value programmed into this field specifies the number of lower address bits allowed to change. For a circular queue application, the SOFF is typically set to the transfer size to implement post-increment addressing with the SMOD function constraining the addresses to a 0-modulo-size range.
5-7 SSIZE	Source data transfer size **NOTE:** Using a Reserved value causes a configuration error. **NOTE:** The eDMA defaults to privileged data access for all transactions. 000b - 8-bit 001b - 16-bit 010b - 32-bit 011b - Reserved 100b - 16-byte burst 101b - 32-byte burst 110b - Reserved 111b - Reserved
8-12 DMOD	Destination Address Modulo See the SMOD definition
13-15 DSIZE	Destination data transfer size See the SSIZE definition

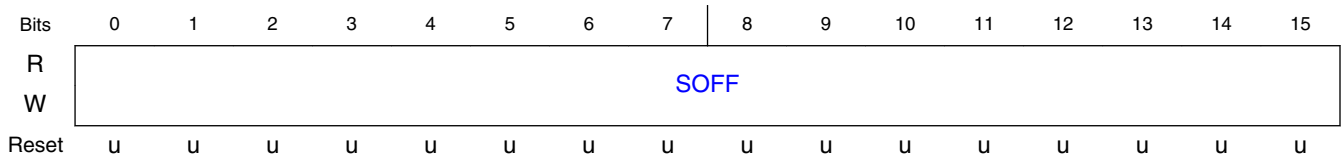
17.4.6.21 TCD Signed Source Address Offset (TCD0_SOFF - TCD31_SOFF)

17.4.6.21.1 Offset

For n = 0 to 31:

Register	Offset
TCDn_SOFF	1006h + (n × 20h)

17.4.6.21.2 Diagram



17.4.6.21.3 Fields

Field	Function
0-15 SOFF	Source address signed offset Sign-extended offset applied to the current source address to form the next-state value as each source read is completed.

17.4.6.22 TCD Minor Byte Count (Minor Loop Mapping Disabled) (TCD0_NBYTES_MLNO - TCD31_NBYTES_MLNO)

17.4.6.22.1 Offset

For n = 0 to 31:

Register	Offset
TCDn_NBYTES_MLNO	1008h + (n × 20h)

17.4.6.22.2 Function

This register, or one of the next two registers (TCD_NBYTES_MLOFFNO, TCD_NBYTES_MLOFFYES), defines the number of bytes to transfer per request. Which register to use depends on whether minor loop mapping is disabled, enabled but not used for this channel, or enabled and used.

TCD word 2 is defined as follows if:

- Minor loop mapping is disabled (CR[EMLM] = 0)

If minor loop mapping is enabled, see the TCD_NBYTES_MLOFFNO and TCD_NBYTES_MLOFFYES register descriptions for the definition of TCD word 2.

17.4.6.22.3 Diagram

Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	NBYTES															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	NBYTES															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

17.4.6.22.4 Fields

Field	Function
0-31 NBYTES	Minor Byte Transfer Count Number of bytes to be transferred in each service request of the channel. As a channel activates, the appropriate TCD contents load into the eDMA engine, and the appropriate reads and writes perform until the minor byte transfer count has transferred. This is an indivisible operation and cannot be halted. It can, however, be stalled by using the bandwidth control field, or via preemption. After the minor count is exhausted, the SADDR and DADDR values are written back into the TCD memory, the major iteration count is decremented and restored to the TCD memory. If the major iteration count is completed, additional processing is performed. **NOTE:** An NBYTES value of 0x0000_0000 is interpreted as a 4 GB transfer.

17.4.6.23 TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (TCD0_NBYTES_MLOFFNO - TCD31_NBYTES_MLOFFNO)

17.4.6.23.1 Offset

For n = 0 to 31:

Register	Offset
TCDn_NBYTES_MLOFFNO	1008h + (n × 20h)

17.4.6.23.2 Function

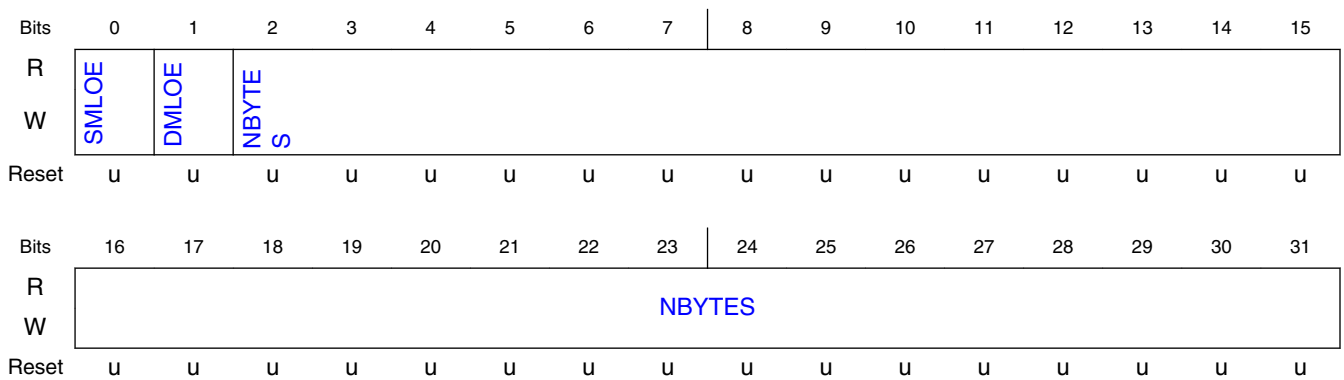
One of three registers (this register, TCD_NBYTES_MLNO, or TCD_NBYTES_MLOFFYES), defines the number of bytes to transfer per request. Which register to use depends on whether minor loop mapping is disabled, enabled but not used for this channel, or enabled and used.

TCD word 2 is defined as follows if:

- Minor loop mapping is enabled (CR[EMLM] = 1) and
- SMLOE = 0 and DMLOE = 0

If minor loop mapping is enabled and SMLOE or DMLOE is set, then refer to the TCD_NBYTES_MLOFFYES register description. If minor loop mapping is disabled, then refer to the TCD_NBYTES_MLNO register description.

17.4.6.23.3 Diagram



17.4.6.23.4 Fields

Field	Function
0 SMLOE	Source Minor Loop Offset Enable Selects whether the minor loop offset is applied to the source address upon minor loop completion. 0b - The minor loop offset is not applied to the SADDR 1b - The minor loop offset is applied to the SADDR
1 DMLOE	Destination Minor Loop Offset enable Selects whether the minor loop offset is applied to the destination address upon minor loop completion. 0b - The minor loop offset is not applied to the DADDR 1b - The minor loop offset is applied to the DADDR
2-31 NBYTES	Minor Byte Transfer Count Number of bytes to be transferred in each service request of the channel.

Field	Function
	As a channel activates, the appropriate TCD contents load into the eDMA engine, and the appropriate reads and writes perform until the minor byte transfer count has transferred. This is an indivisible operation and cannot be halted. It can, however, be stalled by using the bandwidth control field, or via preemption. After the minor count is exhausted, the SADDR and DADDR values are written back into the TCD memory, the major iteration count is decremented and restored to the TCD memory. If the major iteration count is completed, additional processing is performed.

17.4.6.24 TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (TCD0_NBYTES_MLOFFYES - TCD31_NBYTES_MLOFFYES)

17.4.6.24.1 Offset

For n = 0 to 31:

Register	Offset
TCDn_NBYTES_MLOFFYES	1008h + (n × 20h)

17.4.6.24.2 Function

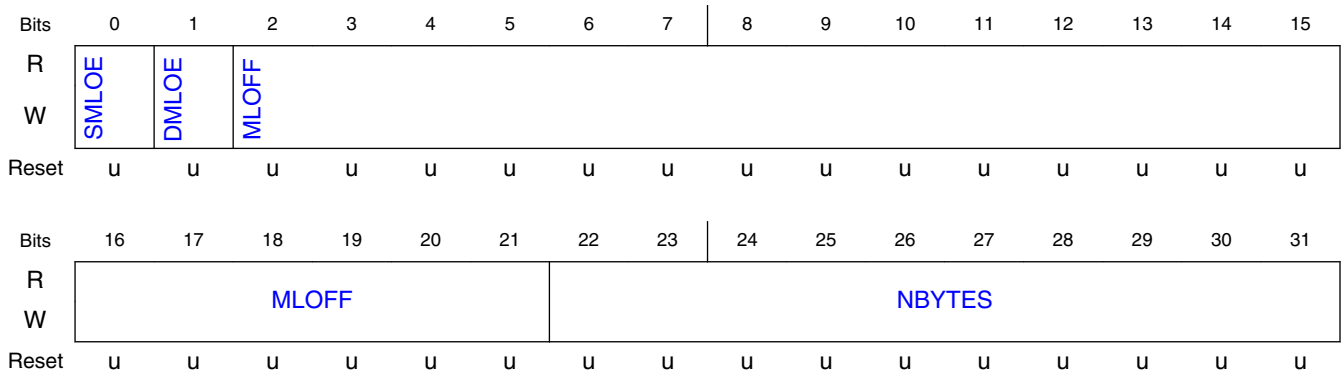
One of three registers (this register, TCD_NBYTES_MLNO, or TCD_NBYTES_MLOFFNO), defines the number of bytes to transfer per request. Which register to use depends on whether minor loop mapping is disabled, enabled but not used for this channel, or enabled and used.

TCD word 2 is defined as follows if:

- Minor loop mapping is enabled (CR[EMLM] = 1) and
- Minor loop offset is enabled (SMLOE or DMLOE = 1)

If minor loop mapping is enabled and SMLOE and DMLOE are cleared, then refer to the TCD_NBYTES_MLOFFNO register description. If minor loop mapping is disabled, then refer to the TCD_NBYTES_MLNO register description.

17.4.6.24.3 Diagram



17.4.6.24.4 Fields

Field	Function
0 SMLOE	Source Minor Loop Offset Enable Selects whether the minor loop offset is applied to the source address upon minor loop completion. 0b - The minor loop offset is not applied to the SADDR 1b - The minor loop offset is applied to the SADDR
1 DMLOE	Destination Minor Loop Offset enable Selects whether the minor loop offset is applied to the destination address upon minor loop completion. 0b - The minor loop offset is not applied to the DADDR 1b - The minor loop offset is applied to the DADDR
2-21 MLOFF	If SMLOE or DMLOE is set, this field represents a sign-extended offset applied to the source or destination address to form the next-state value after the minor loop completes.
22-31 NBYTES	Minor Byte Transfer Count Number of bytes to be transferred in each service request of the channel. As a channel activates, the appropriate TCD contents load into the eDMA engine, and the appropriate reads and writes perform until the minor byte transfer count has transferred. This is an indivisible operation and cannot be halted. It can, however, be stalled by using the bandwidth control field, or via preemption. After the minor count is exhausted, the SADDR and DADDR values are written back into the TCD memory, the major iteration count is decremented and restored to the TCD memory. If the major iteration count is completed, additional processing is performed.

17.4.6.25 TCD Last Source Address Adjustment (TCD0_SLAST - TCD31_SLAST)

17.4.6.25.1 Offset

For n = 0 to 31:

Register	Offset
TCDn_SLAST	100Ch + (n × 20h)

17.4.6.25.2 Diagram

Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	SLAST															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	SLAST															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

17.4.6.25.3 Fields

Field	Function
0-31 SLAST	Last Source Address Adjustment Adjustment value added to the source address at the completion of the major iteration count. This value can be applied to restore the source address to the initial value, or adjust the address to reference the next data structure. This register uses two's complement notation; the overflow bit is discarded.

17.4.6.26 TCD Destination Address (TCD0_DADDR - TCD31_DADDR)

17.4.6.26.1 Offset

For n = 0 to 31:

Register	Offset
TCDn_DADDR	1010h + (n × 20h)

17.4.6.26.2 Function

This register contains the destination address of the transfer.

17.4.6.26.3 Diagram

Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DADDR															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	DADDR															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

17.4.6.26.4 Fields

Field	Function
0-31 DADDR	Destination Address Memory address pointing to the destination data.

17.4.6.27 TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD0_CITER_ELINKNO - TCD31_CITER_ELINKNO)

17.4.6.27.1 Offset

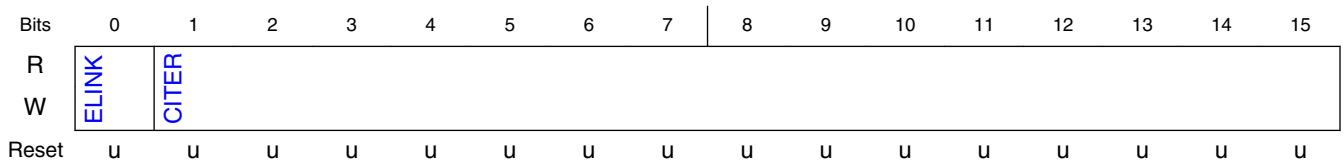
For n = 0 to 31:

Register	Offset
TCDn_CITER_ELINKNO	1014h + (n × 20h)

17.4.6.27.2 Function

This register contains the minor-loop channel-linking configuration and the channel's current iteration count. It is the same register as TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD0_CITER_ELINKYES - TCD31_CITER_ELINKYES), but its fields are defined differently based on the state of the ELINK field. If the ELINK field is cleared, this register is defined as follows.

17.4.6.27.3 Diagram



17.4.6.27.4 Fields

Field	Function
0 ELINK	Enable channel-to-channel linking on minor-loop complete As the channel completes the minor loop, this flag enables linking to another channel, defined by the LINKCH field. The link target channel initiates a channel service request via an internal mechanism that sets the TCDn_CSR[START] bit of the specified channel. If channel linking is disabled, the CITER value is extended to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking. **NOTE:** This bit must be equal to the BITER[ELINK] bit; otherwise, a configuration error is reported. 0b - The channel-to-channel linking is disabled 1b - The channel-to-channel linking is enabled
1-15 CITER	Current Major Iteration Count This field is the current major loop count for the channel. It is decremented each time the minor loop is completed and updated in the transfer control descriptor memory. After the major iteration count is exhausted, the channel performs a number of operations, for example, final source and destination address calculations, optionally generating an interrupt to signal channel completion before reloading the CITER field from the Beginning Iteration Count (BITER) field. **NOTE:** When the CITER field is initially loaded by software, it must be set to the same value as that contained in the BITER field. **NOTE:** If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.

17.4.6.28 TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD0_CITER_ELINKYES - TCD31_CITER_ELINKYES)

17.4.6.28.1 Offset

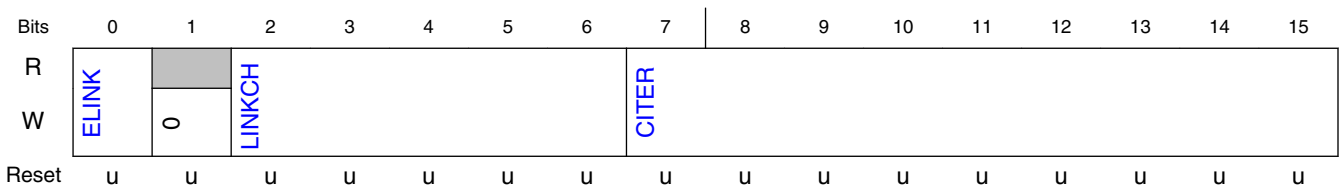
For n = 0 to 31:

Register	Offset
TCDn_CITER_ELINKYES	1014h + (n × 20h)

17.4.6.28.2 Function

This register contains the minor-loop channel-linking configuration and the channel's current iteration count. It is the same register as TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD0_CITER_ELINKNO - TCD31_CITER_ELINKNO), but its fields are defined differently based on the state of the ELINK field. If the ELINK field is set, this register is defined as follows.

17.4.6.28.3 Diagram



17.4.6.28.4 Fields

Field	Function
0 ELINK	Enable channel-to-channel linking on minor-loop complete As the channel completes the minor loop, this flag enables linking to another channel, defined by the LINKCH field. The link target channel initiates a channel service request via an internal mechanism that sets the TCDn_CSR[START] bit of the specified channel. If channel linking is disabled, the CITER value is extended to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking. **NOTE:** This bit must be equal to the BITER[ELINK] bit; otherwise, a configuration error is reported. 0b - The channel-to-channel linking is disabled 1b - The channel-to-channel linking is enabled
1 —	Reserved
2-6 LINKCH	Minor Loop Link Channel Number If channel-to-channel linking is enabled (ELINK = 1), then after the minor loop is exhausted, the eDMA engine initiates a channel service request to the channel defined by this field by setting that channel's TCDn_CSR[START] bit.
7-15 CITER	Current Major Iteration Count This field is the current major loop count for the channel. It is decremented each time the minor loop is completed and updated in the transfer control descriptor memory. After the major iteration count is exhausted, the channel performs a number of operations, for example, final source and destination address calculations, optionally generating an interrupt to signal channel completion before reloading the CITER field from the Beginning Iteration Count (BITER) field.

Field	Function
	NOTE: When the CITER field is initially loaded by software, it must be set to the same value as that contained in the BITER field. **NOTE:** If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.

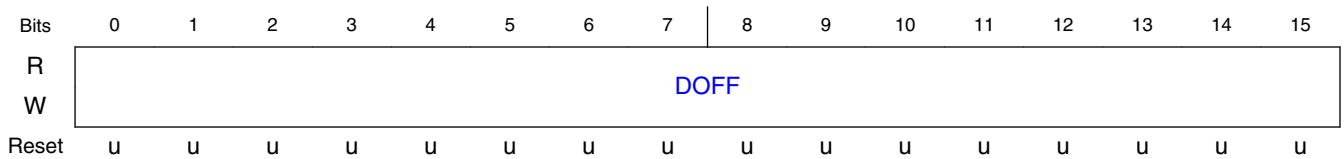
17.4.6.29 TCD Signed Destination Address Offset (TCD0_DOFF - TCD31_DOFF)

17.4.6.29.1 Offset

For n = 0 to 31:

Register	Offset
TCDn_DOFF	1016h + (n × 20h)

17.4.6.29.2 Diagram



17.4.6.29.3 Fields

Field	Function
0-15 DOFF	Destination Address Signed Offset Sign-extended offset applied to the current destination address to form the next-state value as each destination write is completed.

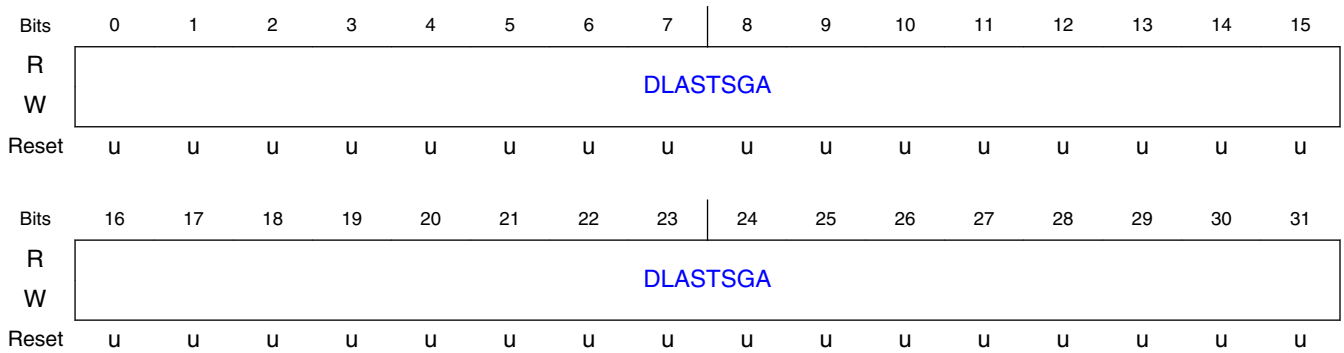
17.4.6.30 TCD Last Destination Address Adjustment/Scatter Gather Address (TCD0_DLASTSGA - TCD31_DLASTSGA)

17.4.6.30.1 Offset

For n = 0 to 31:

Register	Offset
TCDn_DLASTSGA	1018h + (n × 20h)

17.4.6.30.2 Diagram



17.4.6.30.3 Fields

Field	Function
0-31 DLASTSGA	DLASTSGA Destination last address adjustment or the memory address for the next transfer control descriptor to be loaded into this channel (scatter/gather). If (TCDn_CSR[ESG] = 0) then: • Adjustment value added to the destination address at the completion of the major iteration count. This value can apply to restore the destination address to the initial value or adjust the address to reference the next data structure. • This field uses two's complement notation for the final destination address adjustment. Otherwise: • This address points to the beginning of a 0-modulo-32-byte region containing the next transfer control descriptor to be loaded into this channel. This channel reload is performed as the major iteration count completes. The scatter/gather address must be 0-modulo-32-byte, otherwise a configuration error is reported.

17.4.6.31 TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD0_BITER_ELINKNO - TCD31_BITER_ELINKNO)

17.4.6.31.1 Offset

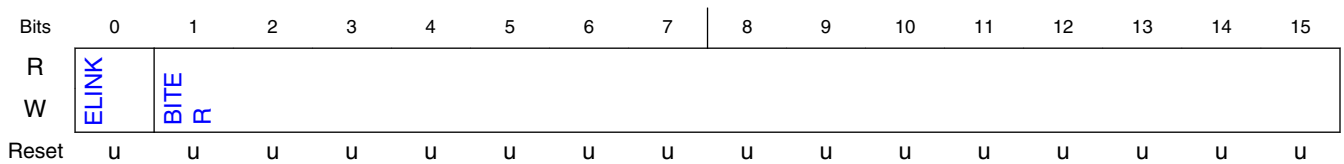
For n = 0 to 31:

Register	Offset
TCDn_BITER_ELINKNO	101Ch + (n × 20h)

17.4.6.31.2 Function

If the TCDn_BITER[ELINK] bit is cleared, the TCDn_BITER register is defined as follows.

17.4.6.31.3 Diagram



17.4.6.31.4 Fields

Field	Function
0 ELINK	Enables channel-to-channel linking on minor loop complete As the channel completes the minor loop, this flag enables the linking to another channel, defined by BITER[LINKCH]. The link target channel initiates a channel service request via an internal mechanism that sets the TCDn_CSR[START] bit of the specified channel. If channel linking is disabled, the BITER value extends to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking. **NOTE:** When the software loads the TCD, this field must be set equal to the corresponding CITER field; otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field. 0b - The channel-to-channel linking is disabled 1b - The channel-to-channel linking is enabled
1-15 BITER	Starting Major Iteration Count As the transfer control descriptor is first loaded by software, this 9-bit (ELINK = 1) or 15-bit (ELINK = 0) field must be equal to the value in the CITER field. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field. **NOTE:** When the software loads the TCD, this field must be set equal to the corresponding CITER field; otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field is reloaded into the CITER field. If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.

17.4.6.32 TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD0_BITER_ELINKYES - TCD31_BITER_ELINKYES)

17.4.6.32.1 Offset

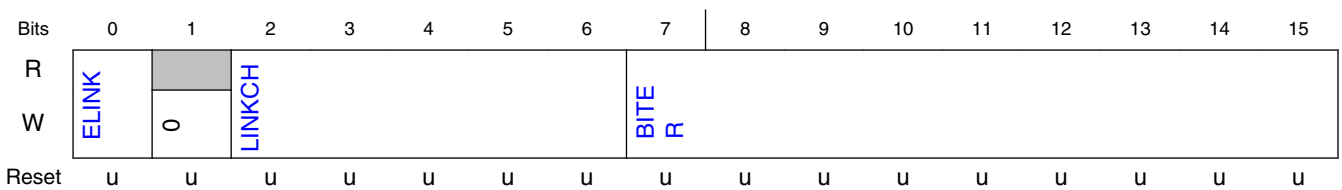
For n = 0 to 31:

Register	Offset
TCDn_BITER_ELINKYES	101Ch + (n × 20h)

17.4.6.32.2 Function

If the TCDn_BITER[ELINK] bit is set, the TCDn_BITER register is defined as follows.

17.4.6.32.3 Diagram



17.4.6.32.4 Fields

Field	Function
0 ELINK	Enables channel-to-channel linking on minor loop complete As the channel completes the minor loop, this flag enables the linking to another channel, defined by BITER[LINKCH]. The link target channel initiates a channel service request via an internal mechanism that sets the TCDn_CSR[START] bit of the specified channel. If channel linking disables, the BITER value extends to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking. **NOTE:** When the software loads the TCD, this field must be set equal to the corresponding CITER field; otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field. 0b - The channel-to-channel linking is disabled 1b - The channel-to-channel linking is enabled
1 —	Reserved
2-6 LINKCH	Link Channel Number

Table continues on the next page...

Field	Function
	If channel-to-channel linking is enabled (ELINK = 1), then after the minor loop is exhausted, the eDMA engine initiates a channel service request at the channel defined by this field by setting that channel's TCDn_CSR[START] bit. **NOTE:** When the software loads the TCD, this field must be set equal to the corresponding CITER field; otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field.
7-15 BITER	Starting major iteration count As the transfer control descriptor is first loaded by software, this 9-bit (ELINK = 1) or 15-bit (ELINK = 0) field must be equal to the value in the CITER field. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field. **NOTE:** When the software loads the TCD, this field must be set equal to the corresponding CITER field; otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field. If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.

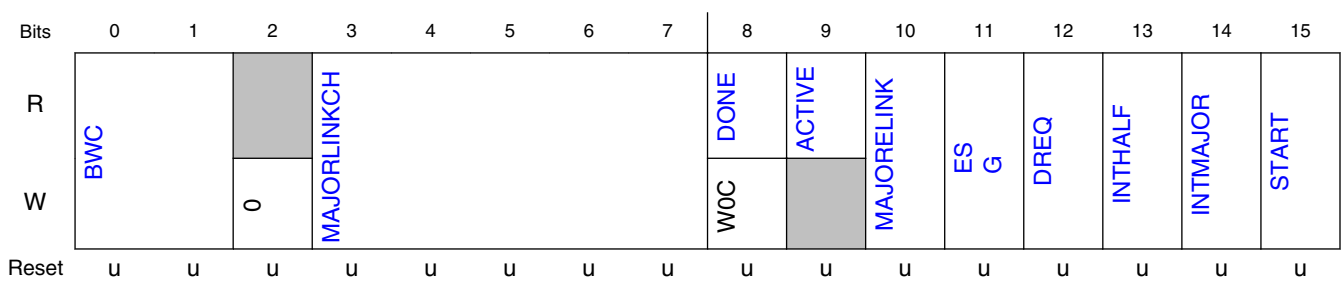
17.4.6.33 TCD Control and Status (TCD0_CSR - TCD31_CSR)

17.4.6.33.1 Offset

For n = 0 to 31:

Register	Offset
TCDn_CSR	101Eh + (n × 20h)

17.4.6.33.2 Diagram



17.4.6.33.3 Fields

Field	Function
0-1 BWC	Bandwidth Control

Table continues on the next page...

Field	Function
	Throttles the amount of bus bandwidth consumed by the eDMA. Generally, as the eDMA processes the minor loop, it continuously generates read/write sequences until the minor count is exhausted. This field forces the eDMA to stall after the completion of each read/write access to control the bus request bandwidth seen by the crossbar switch. **NOTE:** If the source and destination sizes are equal, this field is ignored between the first and second transfers and after the last write of each minor loop. This behavior is a side effect of reducing start-up latency. 00b - No eDMA engine stalls. 01b - Enable eDMA master high-priority elevation (HPE) mode. No eDMA engine stalls. 10b - eDMA engine stalls for 4 cycles after each R/W. 11b - eDMA engine stalls for 8 cycles after each R/W.
2 —	Reserved
3-7 MAJORLINKCH	Major Loop Link Channel Number If (MAJORELINK = 0) then: • No channel-to-channel linking, or chaining, is performed after the major loop counter is exhausted. Otherwise: • After the major loop counter is exhausted, the eDMA engine initiates a channel service request at the channel defined by this field by setting that channel's TCDn_CSR[START] bit.
8 DONE	Channel Done This flag indicates the eDMA has completed the major loop. The eDMA engine sets it as the CITER count reaches zero. The software clears it, or the hardware when the channel is activated. **NOTE:** This bit must be cleared to write the MAJORELINK or ESG bits.
9 ACTIVE	Channel Active This flag signals the channel is currently in execution. It is set when channel service begins, and is cleared by the eDMA as the minor loop completes or when any error condition is detected.
10 MAJORELINK	Enable channel-to-channel linking on major loop complete As the channel completes the major loop, this flag enables the linking to another channel, defined by MAJORLINKCH. The link target channel initiates a channel service request via an internal mechanism that sets the TCDn_CSR[START] bit of the specified channel. **NOTE:** To support the dynamic linking coherency model, this field is forced to zero when written to while the TCDn_CSR[DONE] bit is set. 0b - The channel-to-channel linking is disabled. 1b - The channel-to-channel linking is enabled.
11 ESG	Enable Scatter/Gather Processing As the channel completes the major loop, this flag enables scatter/gather processing in the current channel. If enabled, the eDMA engine uses DLASTSGA as a memory pointer to a 0-modulo-32 address containing a 32-byte data structure loaded as the transfer control descriptor into the local memory. **NOTE:** To support the dynamic scatter/gather coherency model, this field is forced to zero when written to while the TCDn_CSR[DONE] bit is set. 0b - The current channel's TCD is normal format. 1b - The current channel's TCD specifies a scatter gather format. The DLASTSGA field provides a memory pointer to the next TCD to be loaded into this channel after the major loop completes its execution.
12 DREQ	Disable Request If this flag is set, the eDMA hardware automatically clears the corresponding ERQ bit when the current major iteration count reaches zero.

Table continues on the next page...

Field	Function
	0b - The channel's ERQ bit is not affected. 1b - The channel's ERQ bit is cleared when the major loop is complete.
13 INTHALF	Enable an interrupt when major counter is half complete. If this flag is set, the channel generates an interrupt request by setting the appropriate bit in the INT register when the current major iteration count reaches the halfway point. Specifically, the comparison performed by the eDMA engine is (CITER == (BITER >> 1)). This halfway point interrupt request is provided to support double-buffered, also known as ping-pong, schemes or other types of data movement where the processor needs an early indication of the transfer's progress. **NOTE:** If BITER = 1, do not use INTHALF. Use INTMAJOR instead. 0b - The half-point interrupt is disabled. 1b - The half-point interrupt is enabled.
14 INTMAJOR	Enable an interrupt when major iteration count completes. If this flag is set, the channel generates an interrupt request by setting the appropriate bit in the INT when the current major iteration count reaches zero. 0b - The end-of-major loop interrupt is disabled. 1b - The end-of-major loop interrupt is enabled.
15 START	Channel Start If this flag is set, the channel is requesting service. The eDMA hardware automatically clears this flag after the channel begins execution. 0b - The channel is not explicitly started. 1b - The channel is explicitly started via a software initiated service request.

17.5 Functional description

The operation of the eDMA is described in the following subsections.

17.5.1 eDMA basic data flow

The basic flow of a data transfer can be partitioned into three segments.

As shown in the following diagram, the first segment involves the channel activation:

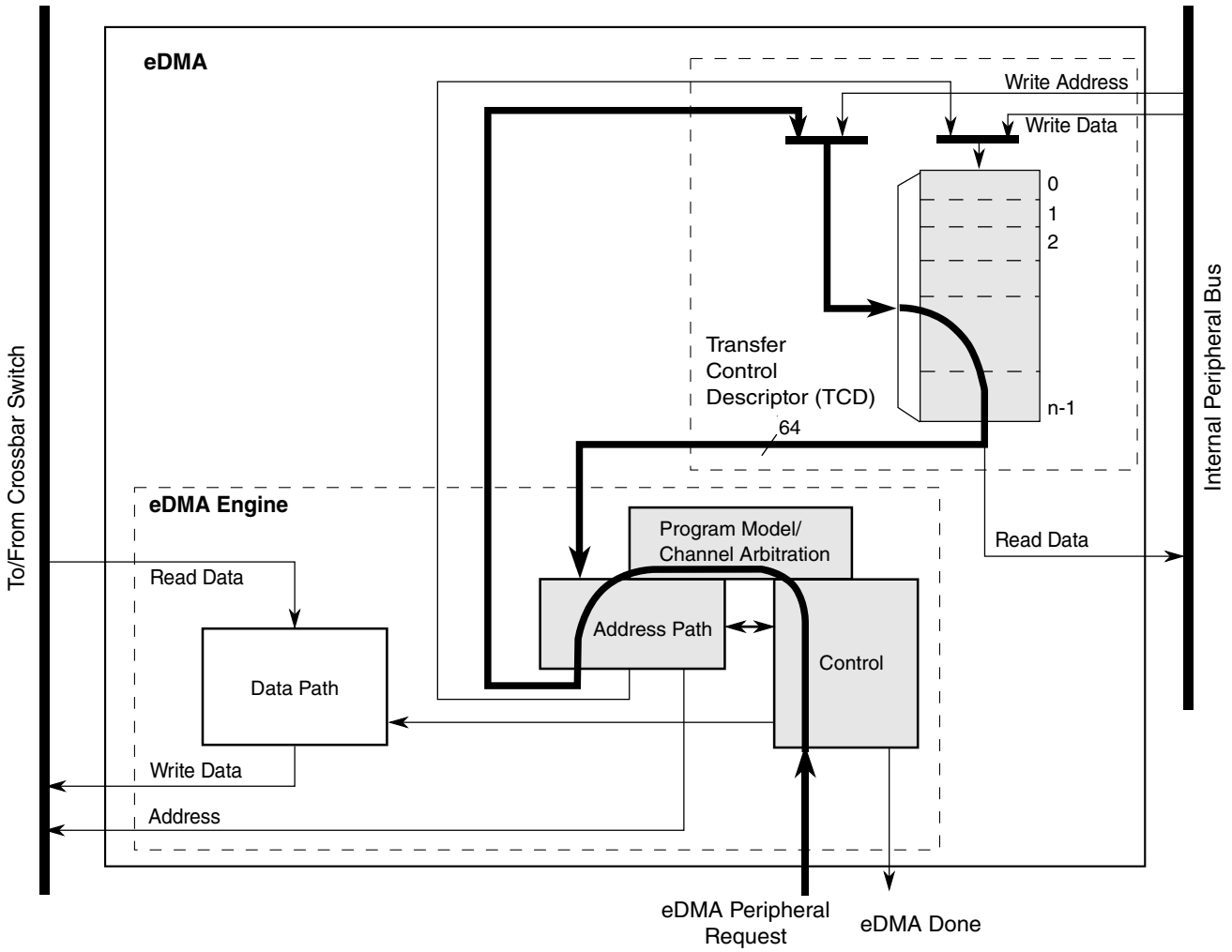


Figure 17-3. eDMA operation, part 1

This example uses the assertion of the eDMA peripheral request signal to request service for channel *n*. Channel activation via software and the TCD*n*_CSR[START] bit follows the same basic flow as peripheral requests. The eDMA request input signal is registered internally and then routed through the eDMA engine: first through the control module, then into the program model and channel arbitration. In the next cycle, the channel arbitration performs, using the fixed-priority or round-robin algorithm. After arbitration is complete, the activated channel number is sent through the address path and converted into the required address to access the local memory for TCD*n*. Next, the TCD memory is accessed and the required descriptor read from the local memory and loaded into the eDMA engine address path channel x or y registers. The TCD memory is 64 bits wide to minimize the time needed to fetch the activated channel descriptor and load it into the address path channel x or y registers.

The following diagram illustrates the second part of the basic data flow:

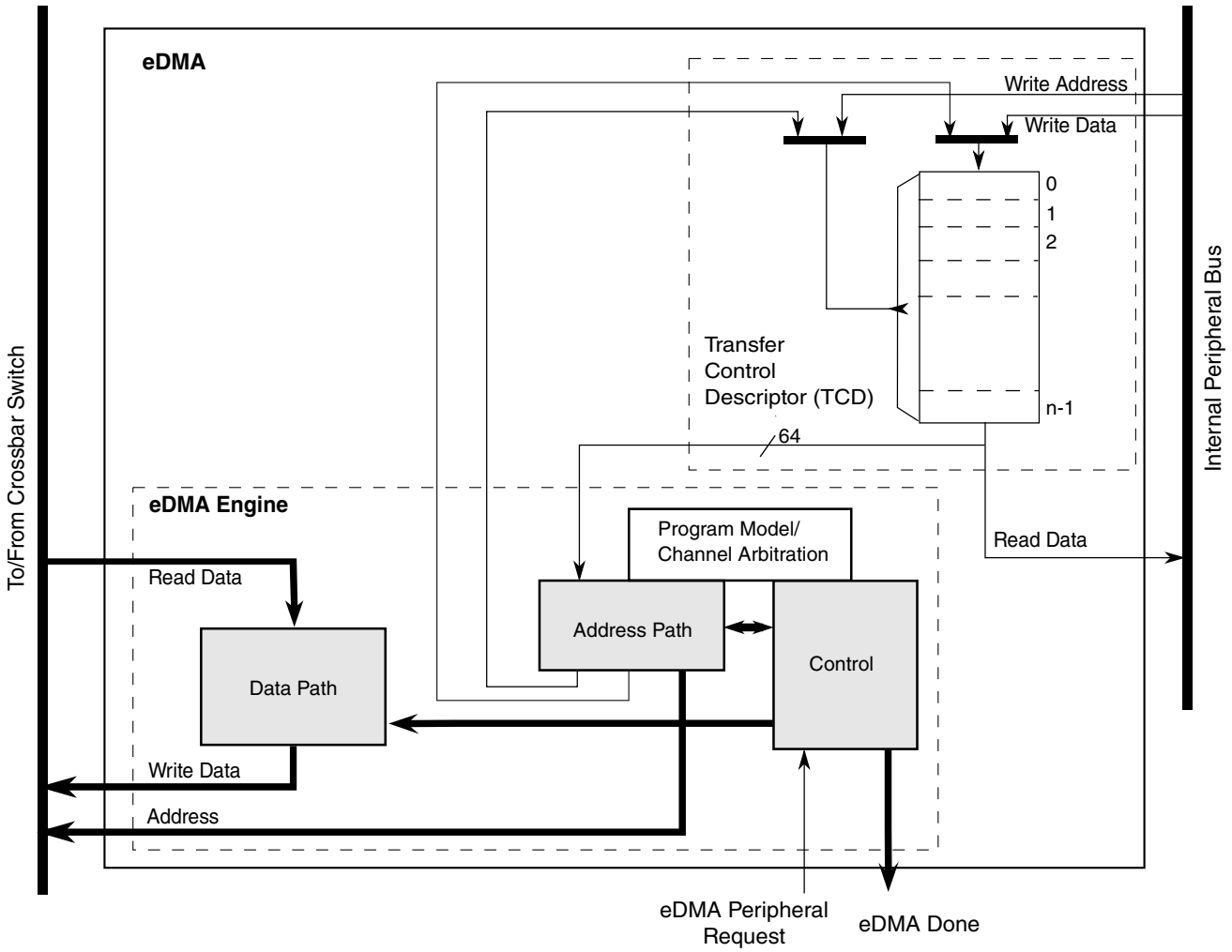


Figure 17-4. eDMA operation, part 2

The modules associated with the data transfer (address path, data path, and control) sequence through the required source reads and destination writes to perform the actual data movement. The source reads are initiated and the fetched data is temporarily stored in the data path block until it is gated onto the internal bus during the destination write. This source read/destination write processing continues until the minor byte count has transferred.

After the minor byte count has moved, the final phase of the basic data flow is performed. In this segment, the address path logic performs the required updates to certain fields in the appropriate TCD, for example, SADDR, DADDR, CITER. If the major iteration count is exhausted, additional operations are performed. These include the final address adjustments and reloading of the BITER field into the CITER. Assertion of an optional interrupt request also occurs at this time, as does a possible fetch of a new TCD from memory using the scatter/gather address pointer included in the descriptor (if scatter/gather is enabled). The updates to the TCD memory and the assertion of an interrupt request are shown in the following diagram.

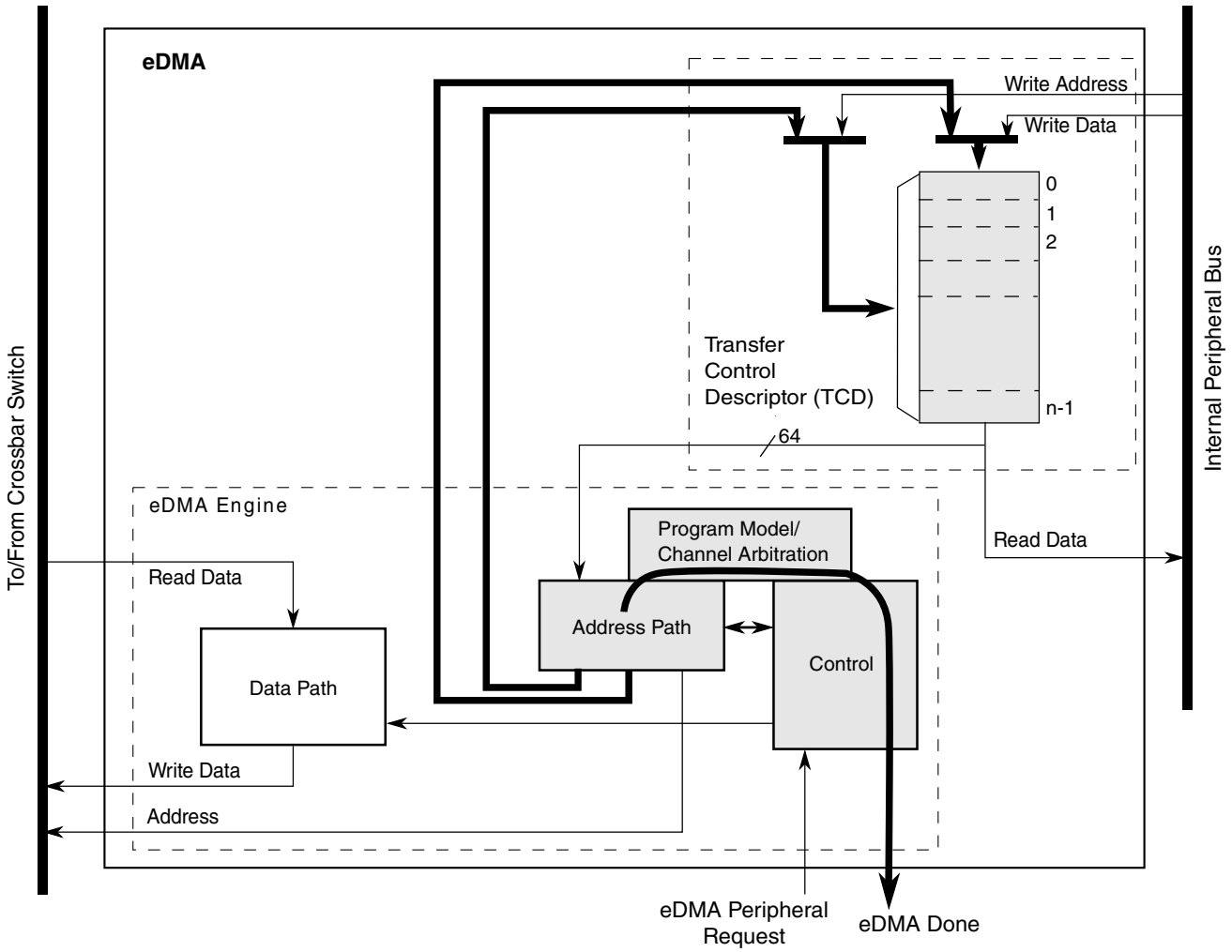


Figure 17-5. eDMA operation, part 3

17.5.2 Fault reporting and handling

Channel errors are reported in the Error Status register (DMAx_ES) and can be caused by:

- A configuration error, which is an illegal setting in the transfer-control descriptor or an illegal priority register setting in Fixed-Arbitration mode, or
- An error termination to a bus master read or write cycle

A configuration error is reported when the starting source or destination address, source or destination offsets, minor loop byte count, or the transfer size represent an inconsistent state. Each of these possible causes are detailed below:

- The addresses and offsets must be aligned on 0-modulo-transfer-size boundaries.
- The minor loop byte count must be a multiple of the source and destination transfer sizes.

- All source reads and destination writes must be configured to the natural boundary of the programmed transfer size respectively.
- In fixed arbitration mode, a configuration error is caused by any two channel priorities being equal. All channel priority levels must be unique when fixed arbitration mode is enabled.

NOTE

When two channels have the same priority, a channel priority error exists and will be reported in the Error Status register. However, the channel number will not be reported in the Error Status register. When all of the channel priorities within a group are not unique, the channel number selected by arbitration in undetermined.

To aid in Channel Priority Error (CPE) debug, set the Halt On Error bit in the DMA's Control Register. If all of the channel priorities within a group are not unique, the DMA will be halted after the CPE error is recorded. The DMA will remain halted and will not process any channel service requests. Once all of the channel priorities are set to unique numbers, the DMA may be enabled again by clearing the Halt bit.

- If a scatter/gather operation is enabled upon channel completion, a configuration error is reported if the scatter/gather address (DLAST_SGA) is not aligned on a 32-byte boundary.
- If minor loop channel linking is enabled upon channel completion, a configuration error is reported when the link is attempted if the TCDn_CITER[E_LINK] bit does not equal the TCDn_BITER[E_LINK] bit.

If enabled, all configuration error conditions, except the scatter/gather and minor-loop link errors, report as the channel activates and asserts an error interrupt request. A scatter/gather configuration error is reported when the scatter/gather operation begins at major loop completion when properly enabled. A minor loop channel link configuration error is reported when the link operation is serviced at minor loop completion.

If a system bus read or write is terminated with an error, the data transfer is stopped and the appropriate bus error flag set. In this case, the state of the channel's transfer control descriptor is updated by the eDMA engine with the current source address, destination address, and current iteration count at the point of the fault. When a system bus error occurs, the channel terminates after the next transfer. Due to pipeline effect, the next transfer is already in progress when the bus error is received by the eDMA. If a bus error

occurs on the last read prior to beginning the write sequence, the write executes using the data captured during the bus error. If a bus error occurs on the last write prior to switching to the next read sequence, the read sequence executes before the channel terminates due to the destination bus error.

A transfer may be cancelled by software with the CR[CX] bit. When a cancel transfer request is recognized, the DMA engine stops processing the channel. The current read-write sequence is allowed to finish. If the cancel occurs on the last read-write sequence of a major or minor loop, the cancel request is discarded and the channel retires normally.

The error cancel transfer is the same as a cancel transfer except the Error Status register (DMAx_ES) is updated with the cancelled channel number and ECX is set. The TCD of a cancelled channel contains the source and destination addresses of the last transfer saved in the TCD. If the channel needs to be restarted, you must re-initialize the TCD because the aforementioned fields no longer represent the original parameters. When a transfer is cancelled by the error cancel transfer mechanism, the channel number is loaded into DMA_ES[ERRCHN] and ECX and VLD are set. In addition, an error interrupt may be generated if enabled.

NOTE

The cancel transfer request allows the user to stop a large data transfer in the event the full data transfer is no longer needed. The cancel transfer bit does not abort the channel. It simply stops the transferring of data and then retires the channel through its normal shutdown sequence. The application software must handle the context of the cancel. If an interrupt is desired (or not), then the interrupt should be enabled (or disabled) before the cancel request. The application software must clean up the transfer control descriptor since the full transfer did not occur.

The occurrence of any error causes the eDMA engine to stop normal processing of the active channel immediately (it goes to its error processing states and the transaction to the system bus still has pipeline effect), and the appropriate channel bit in the eDMA error register is asserted. At the same time, the details of the error condition are loaded into the Error Status register (DMAx_ES). The major loop complete indicators, setting the transfer control descriptor DONE flag and the possible assertion of an interrupt request, are not affected when an error is detected. After the error status has been updated, the eDMA engine continues operating by servicing the next appropriate channel. A channel that experiences an error condition is not automatically disabled. If a channel is terminated by an error and then issues another service request before the error is fixed, that channel executes and terminates with the same error condition.

17.5.3 Channel preemption

Channel preemption is enabled on a per-channel basis by setting the DCHPRIn[ECP] bit. Channel preemption allows the executing channel's data transfers to temporarily suspend in favor of starting a higher priority channel. After the preempting channel has completed all its minor loop data transfers, the preempted channel is restored and resumes execution. After the restored channel completes one read/write sequence, it is again eligible for preemption. If any higher priority channel is requesting service, the restored channel is suspended and the higher priority channel is serviced. Nested preemption, that is, attempting to preempt a preempting channel, is not supported. After a preempting channel begins execution, it cannot be preempted. Preemption is available only when fixed arbitration is selected.

A channel's ability to preempt another channel can be disabled by setting DCHPRIn[DPA]. When a channel's preempt ability is disabled, that channel cannot suspend a lower priority channel's data transfer, regardless of the lower priority channel's ECP setting. This allows for a pool of low priority, large data-moving channels to be defined. These low priority channels can be configured to not preempt each other, thus preventing a low priority channel from consuming the preempt slot normally available to a true, high priority channel.

17.6 Initialization/application information

The following sections discuss initialization of the eDMA and programming considerations.

17.6.1 eDMA initialization

To initialize the eDMA:

1. Write to the CR if a configuration other than the default is desired.
2. Write the channel priority levels to the DCHPRI*n* registers if a configuration other than the default is desired.
3. Enable error interrupts in the EEI register if so desired.
4. Write the 32-byte TCD for each channel that may request service.
5. Enable any hardware service requests via the ERQ register.

6. Request channel service via either:
    - Software: setting the TCD*n*_CSR[START]
    - Hardware: slave device asserting its eDMA peripheral request signal

After any channel requests service, a channel is selected for execution based on the arbitration and priority levels written into the programmer's model. The eDMA engine reads the entire TCD, including the TCD control and status fields, as shown in the following table, for the selected channel into its internal address path module.

As the TCD is read, the first transfer is initiated on the internal bus, unless a configuration error is detected. Transfers from the source, as defined by TCD*n*_SADDR, to the destination, as defined by TCD*n*_DADDR, continue until the number of bytes specified by TCD*n*_NBYTES are transferred.

When the transfer is complete, the eDMA engine's local TCD*n*_SADDR, TCD*n*_DADDR, and TCD*n*_CITER are written back to the main TCD memory and any minor loop channel linking is performed, if enabled. If the major loop is exhausted, further post processing executes, such as interrupts, major loop channel linking, and scatter/gather operations, if enabled.

Table 17-5. TCD Control and Status fields

TCD*n*_CSR field name	Description
START	Control bit to start channel explicitly when using a software initiated DMA service (Automatically cleared by hardware)
ACTIVE	Status bit indicating the channel is currently in execution
DONE	Status bit indicating major loop completion (cleared by software when using a software initiated DMA service)
D_REQ	Control bit to disable DMA request at end of major loop completion when using a hardware initiated DMA service
BWC	Control bits for throttling bandwidth control of a channel
E_SG	Control bit to enable scatter-gather feature
INT_HALF	Control bit to enable interrupt when major loop is half complete
INT_MAJ	Control bit to enable interrupt when major loop completes

The following figure shows how each DMA request initiates one minor-loop transfer, or iteration, without CPU intervention. DMA arbitration can occur after each minor loop, and one level of minor loop DMA preemption is allowed. The number of minor loops in a major loop is specified by the beginning iteration count (BITER).

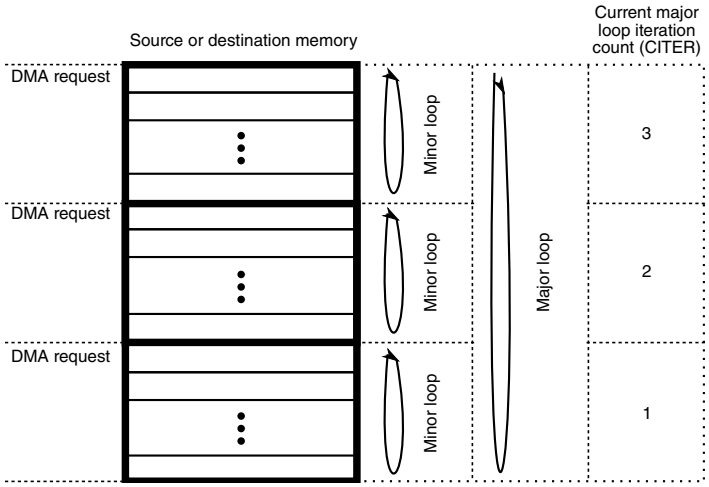


Figure 17-6. Example of multiple loop iterations

The following figure lists the memory array terms and how the TCD settings interrelate.

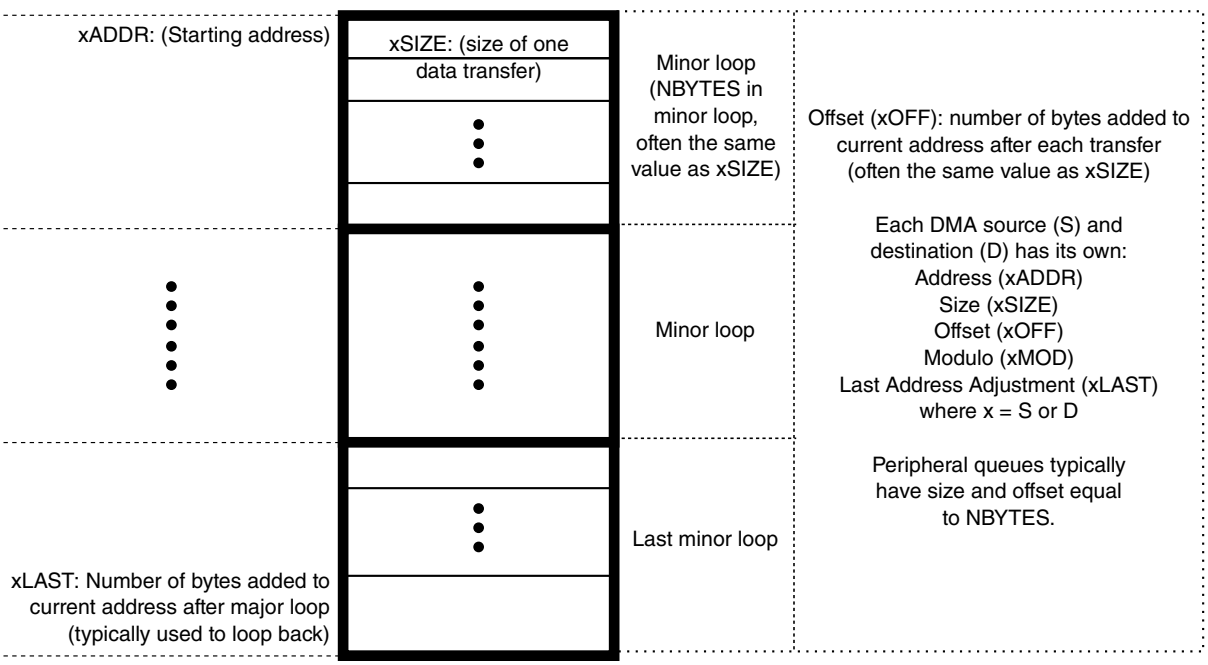


Figure 17-7. Memory array terms

17.6.2 Programming errors

The eDMA performs various tests on the transfer control descriptor to verify consistency in the descriptor data. Most programming errors are reported on a per channel basis with the exception of channel priority error (ES[CPE]).

For all error types other than group or channel priority errors, the channel number causing the error is recorded in the Error Status register (DMAx_ES). If the error source is not removed before the next activation of the problem channel, the error is detected and recorded again.

Channel priority errors are identified within a group once that group has been selected as the active group. For example:

1. The eDMA is configured for fixed group and fixed channel arbitration modes.
2. Group 1 is the highest priority and all channels are unique in that group.
3. Group 0 is the next highest priority and has two channels with the same priority level.
4. If Group 1 has any service requests, those requests will be executed.
5. After all of Group 1 requests have completed, Group 0 will be the next active group.
6. If Group 0 has a service request, then an undefined channel in Group 0 will be selected and a channel priority error will occur.
7. This repeats until the all of Group 0 requests have been removed or a higher priority Group 1 request comes in.

In this sequence, for item 2, the eDMA acknowledge lines will assert only if the selected channel is requesting service via the eDMA peripheral request signal. If interrupts are enabled for all channels, the user will get an error interrupt, but the channel number for the ERR register and the error interrupt request line may be wrong because they reflect the selected channel. A group priority error is global and any request in any group will cause a group priority error.

If priority levels are not unique, when any channel requests service, a channel priority error is reported. The highest channel/group priority with an active request is selected, but the lowest numbered channel with that priority is selected by arbitration and executed by the eDMA engine. The hardware service request handshake signals, error interrupts, and error reporting is associated with the selected channel.

17.6.3 Arbitration mode considerations

This section discusses arbitration considerations for the eDMA.

17.6.3.1 Fixed group arbitration, Fixed channel arbitration

In this mode, the channel service request from the highest priority channel in the highest priority group is selected to execute. If the eDMA is programmed so that the channels within one group use "fixed" priorities, and that group is assigned the highest "fixed" priority of all groups, that group can take all the bandwidth of the eDMA controller. That is, no other groups will be serviced if there is always at least one DMA request pending on a channel in the highest priority group when the controller arbitrates the next DMA request. The advantage of this scenario is that latency can be small for channels that need to be serviced quickly. Preemption is available in this scenario only.

17.6.3.2 Fixed group arbitration, Round-robin channel arbitration

The highest priority group with a request will be serviced. Lower priority groups will be serviced if no pending requests exist in the higher priority groups.

Within each group, c hannels are serviced starting with the highest channel number and rotating through to the lowest channel number without regard to the channel priority levels assigned within the group.

This scenario could cause the same bandwidth consumption problem as indicated in Fixed group arbitration, Fixed channel arbitration, but all the channels in the highest priority group will be serviced. Service latency will be short on the highest priority group, but could potentially be very much longer as the group priority decreases.

17.6.4 Performing DMA transfers

This section presents examples on how to perform DMA transfers with the eDMA.

17.6.4.1 Single request

To perform a simple transfer of n bytes of data with one activation, set the major loop to one (TCD*n*_CITER = TCD*n*_BITER = 1). The data transfer begins after the channel service request is acknowledged and the channel is selected to execute. After the transfer is complete, the TCD*n*_CSR[DONE] bit is set and an interrupt generates if properly enabled.

For example, the following TCD entry is configured to transfer 16 bytes of data. The eDMA is programmed for one iteration of the major loop transferring 16 bytes per iteration. The source memory has a byte wide memory port located at 0x1000. The destination memory has a 32-bit port located at 0x2000. The address offsets are

programmed in increments to match the transfer size: one byte for the source and four bytes for the destination. The final source and destination addresses are adjusted to return to their beginning values.

```
TCDn_CITER = TCDn_BITER = 1
TCDn_NBYTES = 16
TCDn_SADDR = 0x1000
TCDn_SOFF = 1
TCDn_ATTR[SSIZE] = 0
TCDn_SLAST = -16
TCDn_DADDR = 0x2000
TCDn_DOFF = 4
TCDn_ATTR[DSIZE] = 2
TCDn_DLAST_SGA= -16
TCDn_CSR[INT_MAJ] = 1
TCDn_CSR[START] = 1 (Should be written last after all other fields have been initialized)
All other TCDn fields = 0
```

This generates the following event sequence:

1. User write to the TCD*n*_CSR[START] bit requests channel service.
2. The channel is selected by arbitration for servicing.
3. eDMA engine writes: TCD*n*_CSR[DONE] = 0, TCD*n*_CSR[START] = 0, TCD*n*_CSR[ACTIVE] = 1.
4. eDMA engine reads: channel TCD data from local memory to internal register file.
5. The source-to-destination transfers are executed as follows:
    a. Read byte from location 0x1000, read byte from location 0x1001, read byte from 0x1002, read byte from 0x1003.
    b. Write 32-bits to location 0x2000 → first iteration of the minor loop.
    c. Read byte from location 0x1004, read byte from location 0x1005, read byte from 0x1006, read byte from 0x1007.
    d. Write 32-bits to location 0x2004 → second iteration of the minor loop.
    e. Read byte from location 0x1008, read byte from location 0x1009, read byte from 0x100A, read byte from 0x100B.
    f. Write 32-bits to location 0x2008 → third iteration of the minor loop.
    g. Read byte from location 0x100C, read byte from location 0x100D, read byte from 0x100E, read byte from 0x100F.
    h. Write 32-bits to location 0x200C → last iteration of the minor loop → major loop complete.

6. The eDMA engine writes: TCD*n*_SADDR = 0x1000, TCD*n*_DADDR = 0x2000, TCD*n*_CITER = 1 (TCD*n*_BITER).
7. The eDMA engine writes: TCD*n*_CSR[ACTIVE] = 0, TCD*n*_CSR[DONE] = 1, INT[*n*] = 1.
8. The channel retires and the eDMA goes idle or services the next channel.

17.6.4.2 Multiple requests

The following example transfers 32 bytes via two hardware requests, but is otherwise the same as the previous example. The only fields that change are the major loop iteration count and the final address offsets. The eDMA is programmed for two iterations of the major loop transferring 16 bytes per iteration. After the channel's hardware requests are enabled in the ERQ register, the slave device initiates channel service requests.

```
TCDn_CITER = TCDn_BITER = 2
TCDn_SLAST = -32
TCDn_DLAST_SGA = -32
```

This would generate the following sequence of events:

1. First hardware, that is, eDMA peripheral, request for channel service.
2. The channel is selected by arbitration for servicing.
3. eDMA engine writes: TCD*n*_CSR[DONE] = 0, TCD*n*_CSR[START] = 0, TCD*n*_CSR[ACTIVE] = 1.
4. eDMA engine reads: channel TCD*n* data from local memory to internal register file.
5. The source to destination transfers are executed as follows:
   a. Read byte from location 0x1000, read byte from location 0x1001, read byte from 0x1002, read byte from 0x1003.
   b. Write 32-bits to location 0x2000 → first iteration of the minor loop.
   c. Read byte from location 0x1004, read byte from location 0x1005, read byte from 0x1006, read byte from 0x1007.
   d. Write 32-bits to location 0x2004 → second iteration of the minor loop.
   e. Read byte from location 0x1008, read byte from location 0x1009, read byte from 0x100A, read byte from 0x100B.
   f. Write 32-bits to location 0x2008 → third iteration of the minor loop.

g. Read byte from location 0x100C, read byte from location 0x100D, read byte from 0x100E, read byte from 0x100F.

h. Write 32-bits to location 0x200C → last iteration of the minor loop.

6. eDMA engine writes: TCD*n*_SADDR = 0x1010, TCD*n*_DADDR = 0x2010, TCD*n*_CITER = 1.
7. eDMA engine writes: TCD*n*_CSR[ACTIVE] = 0.
8. The channel retires → one iteration of the major loop. The eDMA goes idle or services the next channel.
9. Second hardware, that is, eDMA peripheral, requests channel service.
10. The channel is selected by arbitration for servicing.
11. eDMA engine writes: TCD*n*_CSR[DONE] = 0, TCD*n*_CSR[START] = 0, TCD*n*_CSR[ACTIVE] = 1.
12. eDMA engine reads: channel TCD data from local memory to internal register file.
13. The source to destination transfers are executed as follows:

    a. Read byte from location 0x1010, read byte from location 0x1011, read byte from 0x1012, read byte from 0x1013.

    b. Write 32-bits to location 0x2010 → first iteration of the minor loop.

    c. Read byte from location 0x1014, read byte from location 0x1015, read byte from 0x1016, read byte from 0x1017.

    d. Write 32-bits to location 0x2014 → second iteration of the minor loop.

    e. Read byte from location 0x1018, read byte from location 0x1019, read byte from 0x101A, read byte from 0x101B.

    f. Write 32-bits to location 0x2018 → third iteration of the minor loop.

    g. Read byte from location 0x101C, read byte from location 0x101D, read byte from 0x101E, read byte from 0x101F.

    h. Write 32-bits to location 0x201C → last iteration of the minor loop → major loop complete.

14. eDMA engine writes: TCD*n*_SADDR = 0x1000, TCD*n*_DADDR = 0x2000, TCD*n*_CITER = 2 (TCD*n*_BITER).
15. eDMA engine writes: TCD*n*_CSR[ACTIVE] = 0, TCD*n*_CSR[DONE] = 1, INT[n] = 1.

16. The channel retires → major loop complete. The eDMA goes idle or services the next channel.

17.6.4.3 Using the modulo feature

The modulo feature of the eDMA provides the ability to implement a circular data queue in which the size of the queue is a power of 2. MOD is a 5-bit field for the source and destination in the TCD, and it specifies which lower address bits increment from their original value after the address+offset calculation. All upper address bits remain the same as in the original value. A setting of 0 for this field disables the modulo feature.

The following table shows how the transfer addresses are specified based on the setting of the MOD field. Here a circular buffer is created where the address wraps to the original value while the 28 upper address bits (0x1234567*x*) retain their original value. In this example the source address is set to 0x12345670, the offset is set to 4 bytes and the MOD field is set to 4, allowing for a 2^4 byte (16-byte) size queue.

Table 17-6. Modulo example

Transfer Number	Address
1	0x12345670
2	0x12345674
3	0x12345678
4	0x1234567C
5	0x12345670
6	0x12345674

17.6.5 Monitoring transfer descriptor status

This section discusses how to monitor eDMA status.

17.6.5.1 Testing for minor loop completion

There are two methods to test for minor loop completion when using software initiated service requests. The first is to read the TCD*n*_CITER field and test for a change. Another method may be extracted from the sequence shown below. The second method is to test the TCD*n*_CSR[START] bit and the TCD*n*_CSR[ACTIVE] bit. The minor-loop-

complete condition is indicated by both bits reading zero after the TCD*n*_CSR[START] was set. Polling the TCD*n*_CSR[ACTIVE] bit may be inconclusive, because the active status may be missed if the channel execution is short in duration.

The TCD status bits execute the following sequence for a software activated channel:

Stage	TCD*n*_CSR bits			State
	START	ACTIVE	DONE	
1	1	0	0	Channel service request via software
2	0	1	0	Channel is executing
3a	0	0	0	Channel has completed the minor loop and is idle
3b	0	0	1	Channel has completed the major loop and is idle

The best method to test for minor-loop completion when using hardware, that is, peripheral, initiated service requests is to read the TCD*n*_CITER field and test for a change. The hardware request and acknowledge handshake signals are not visible in the programmer's model.

The TCD status bits execute the following sequence for a hardware-activated channel:

Stage	TCD*n*_CSR bits			State
	START	ACTIVE	DONE	
1	0	0	0	Channel service request via hardware (peripheral request asserted)
2	0	1	0	Channel is executing
3a	0	0	0	Channel has completed the minor loop and is idle
3b	0	0	1	Channel has completed the major loop and is idle

For both activation types, the major-loop-complete status is explicitly indicated via the TCD*n*_CSR[DONE] bit.

The TCD*n*_CSR[START] bit is cleared automatically when the channel begins execution regardless of how the channel activates.

17.6.5.2 Reading the transfer descriptors of active channels

The eDMA reads back the true TCD*n*_SADDR, TCD*n*_DADDR, and TCD*n*_NBYTES values if read while a channel executes. The true values of the SADDR, DADDR, and NBYTES are the values the eDMA engine currently uses in its internal register file and not the values in the TCD local memory for that channel. The addresses, SADDR and

DADDR, and NBYTES, which decrement to zero as the transfer progresses, can give an indication of the progress of the transfer. All other values are read back from the TCD local memory.

17.6.5.3 Checking channel preemption status

Preemption is available only when fixed arbitration is selected for both group and channel arbitration modes. A preemptive situation is one in which a preempt-enabled channel runs and a higher priority request becomes active. When the eDMA engine is not operating in fixed group, fixed channel arbitration mode, the determination of the actively running relative priority outstanding requests become undefined. Channel and/or group priorities are treated as equal, that is, constantly rotating, when Round-Robin Arbitration mode is selected.

The TCD*n*_CSR[ACTIVE] bit for the preempted channel remains asserted throughout the preemption. The preempted channel is temporarily suspended while the preempting channel executes one major loop iteration. If two TCD*n*_CSR[ACTIVE] bits are set simultaneously in the global TCD map, a higher priority channel is actively preempting a lower priority channel.

17.6.6 Channel Linking

Channel linking (or chaining) is a mechanism where one channel sets the TCD*n*_CSR[START] bit of another channel (or itself), therefore initiating a service request for that channel. When properly enabled, the EDMA engine automatically performs this operation at the major or minor loop completion.

The minor loop channel linking occurs at the completion of the minor loop (or one iteration of the major loop). The TCD*n*_CITER[E_LINK] field determines whether a minor loop link is requested. When enabled, the channel link is made after each iteration of the major loop except for the last. When the major loop is exhausted, only the major loop channel link fields are used to determine if a channel link should be made. For example, the initial fields of:

```
TCDn_CITER[E_LINK] = 1
TCDn_CITER[LINKCH] = 0xC
TCDn_CITER[CITER] value = 0x4
TCDn_CSR[MAJOR_E_LINK] = 1
TCDn_CSR[MAJOR_LINKCH] = 0x7
```

executes as:

1. Minor loop done → set TCD12_CSR[START] bit

2. Minor loop done → set TCD12_CSR[START] bit
3. Minor loop done → set TCD12_CSR[START] bit
4. Minor loop done, major loop done→ set TCD7_CSR[START] bit

When minor loop linking is enabled (TCD*n*_CITER[E_LINK] = 1), the TCD*n*_CITER[CITER] field uses a nine bit vector to form the current iteration count. When minor loop linking is disabled (TCD*n*_CITER[E_LINK] = 0), the TCD*n*_CITER[CITER] field uses a 15-bit vector to form the current iteration count. The bits associated with the TCD*n*_CITER[LINKCH] field are concatenated onto the CITER value to increase the range of the CITER.

Note

The TCD*n*_CITER[E_LINK] bit and the TCD*n*_BITER[E_LINK] bit must equal or a configuration error is reported. The CITER and BITER vector widths must be equal to calculate the major loop, half-way done interrupt point.

The following table summarizes how a DMA channel can link to another DMA channel, i.e, use another channel's TCD, at the end of a loop.

Table 17-7. Channel Linking Parameters

Desired Link Behavior	TCD Control Field Name	Description
Link at end of Minor Loop	CITER[E_LINK]	Enable channel-to-channel linking on minor loop completion (current iteration)
	CITER[LINKCH]	Link channel number when linking at end of minor loop (current iteration)
Link at end of Major Loop	CSR[MAJOR_E_LINK]	Enable channel-to-channel linking on major loop completion
	CSR[MAJOR_LINKCH]	Link channel number when linking at end of major loop

17.6.7 Dynamic programming

This section provides recommended methods to change the programming model during channel execution.

17.6.7.1 Dynamically changing the channel priority

The following two options are recommended for dynamically changing channel priority levels:

1. Switch to Round-Robin Channel Arbitration mode, change the channel priorities, then switch back to Fixed Arbitration mode,
2. Disable all the channels, change the channel priorities, then enable the appropriate channels.

17.6.7.2 Dynamic channel linking

Dynamic channel linking is the process of setting the TCDn_CSR[MAJORELINK] bit during channel execution (see the diagram in TCD structure). This bit is read from the TCD local memory at the end of channel execution, thus allowing the user to enable the feature during channel execution.

Because the user is allowed to change the configuration during execution, a coherency model is needed. Consider the scenario where the user attempts to execute a dynamic channel link by enabling the TCDn_CSR[MAJORELINK] bit at the same time the eDMA engine is retiring the channel. The TCDn_CSR[MAJORELINK] would be set in the programmer's model, but it would be unclear whether the actual link was made before the channel retired.

The following coherency model is recommended when executing a dynamic channel link request.

1. Write 1 to the TCDn_CSR[MAJORELINK] bit.
2. Read back the TCDn_CSR[MAJORELINK] bit.
3. Test the TCDn_CSR[MAJORELINK] request status:
   - If TCDn_CSR[MAJORELINK] = 1, the dynamic link attempt was successful.
   - If TCDn_CSR[MAJORELINK] = 0, the attempted dynamic link did not succeed (the channel was already retiring).

For this request, the TCD local memory controller forces the TCDn_CSR[MAJORELINK] bit to zero on any writes to a channel's TCD.word7 after that channel's TCD.done bit is set, indicating the major loop is complete.

NOTE

The user must clear the The TCDn_CSR[DONE] bit before writing the TCDn_CSR[MAJORELINK] bit. The TCDn_CSR[DONE] bit is cleared automatically by the eDMA engine after a channel begins execution.

17.6.7.3 Dynamic scatter/gather

Scatter/gather is the process of automatically loading a new TCD into a channel. It allows a DMA channel to use multiple TCDs; this enables a DMA channel to scatter the DMA data to multiple destinations or gather it from multiple sources. When scatter/gather is enabled and the channel has finished its major loop, a new TCD is fetched from system memory and loaded into that channel's descriptor location in eDMA programmer's model, thus replacing the current descriptor.

Because the user is allowed to change the configuration during execution, a coherency model is needed. Consider the scenario where the user attempts to execute a dynamic scatter/gather operation by enabling the TCDn_CSR[ESG] bit at the same time the eDMA engine is retiring the channel. The ESG bit would be set in the programmer's model, but it would be unclear whether the actual scatter/gather request was honored before the channel retired.

Two methods for this coherency model are shown in the following subsections. Method 1 has the advantage of reading the MAJORLINKCH field and the ESG bit with a single read. For both dynamic channel linking and scatter/gather requests, the TCD local memory controller forces the TCD MAJOR.E_LINK and E_SG bits to zero on any writes to a channel's TCD word 7 if that channel's TCD.DONE bit is set indicating the major loop is complete.

NOTE

The user must clear the TCDn_CSR[DONE] bit before writing the MAJORELINK or ESG bits. The TCDn_CSR[DONE] bit is cleared automatically by the eDMA engine after a channel begins execution.

17.6.7.3.1 Method 1 (channel not using major loop channel linking)

For a channel not using major loop channel linking, the coherency model described here may be used for a dynamic scatter/gather request.

When the TCDn_CSR[MAJORELINK] bit is zero, the TCDn_CSR[MAJORLINKCH] field is not used by the eDMA. In this case, the MAJORLINKCH field may be used for other purposes. This method uses the MAJORLINKCH field as a TCD identification (ID).

1. When the descriptors are built, write a unique TCD ID in the TCDn_CSR[MAJORLINKCH] field for each TCD associated with a channel using dynamic scatter/gather.
2. Write 1b to the TCDn_CSR[DREQ] bit.

Should a dynamic scatter/gather attempt fail, setting the DREQ bit will prevent a future hardware activation of this channel. This stops the channel from executing with a destination address (DADDR) that was calculated using a scatter/gather address (written in the next step) instead of a dlast final offset value.

3. Write the TCDn_DLASTSGA register with the scatter/gather address.
4. Write 1b to the TCDn_CSR[ESG] bit.
5. Read back the 16 bit TCD control/status field.
6. Test the ESG request status and MAJORLINKCH value in the TCDn_CSR register:

   If ESG = 1b, the dynamic link attempt was successful.

   If ESG = 0b and the MAJORLINKCH (ID) did not change, the attempted dynamic link did not succeed (the channel was already retiring).

   If ESG = 0b and the MAJORLINKCH (ID) changed, the dynamic link attempt was successful (the new TCD's E_SG value cleared the ESG bit).

17.6.7.3.2 Method 2 (channel using major loop channel linking)

For a channel using major loop channel linking, the coherency model described here may be used for a dynamic scatter/gather request. This method uses the TCD.DLAST_SGA field as a TCD identification (ID).

1. Write 1b to the TCDn_CSR[DREQ] bit.

   Should a dynamic scatter/gather attempt fail, setting the DREQ bit will prevent a future hardware activation of this channel. This stops the channel from executing with a destination address (DADDR) that was calculated using a scatter/gather address (written in the next step) instead of a dlast final offset value.

2. Write the TCDn_DLASTSGA register with the scatter/gather address.
3. Write 1b to the TCDn_CSR[ESG] bit.
4. Read back the ESG bit.
5. Test the ESG request status:

   If ESG = 1b, the dynamic link attempt was successful.

   If ESG = 0b, read the 32 bit TCDn_DLASTSGA field.

   If ESG = 0b and the TCDn_DLASTSGA did not change, the attempted dynamic link did not succeed (the channel was already retiring).

If ESG = 0b and the TCDn_DLASTSGA changed, the dynamic link attempt was successful (the new TCD's E_SG value cleared the ESG bit).

17.6.8 Suspend/resume a DMA channel with active hardware service requests

The DMA allows the user to move data from memory or peripheral registers to another location in memory or peripheral registers without CPU interaction. Once the DMA and peripherals have been configured and are active, it is rare to suspend a peripheral's service request dynamically. In this scenario, there are certain restrictions to disabling a DMA hardware service request. For coherency, a specific procedure must be followed. This section provides guidance on how to coherently suspend and resume a Direct Memory Access (DMA) channel when the DMA is triggered by a slave module such as the Serial Peripheral Interface (SPI), Sigma Delta Analog to Digital Convertor (SDADC), or other module.

17.6.8.1 Suspend an active DMA channel

To suspend an active DMA channel:

1. Stop the DMA service request at the peripheral first. Confirm it has been disabled by reading back the appropriate register in the peripheral.
2. Check the DMA's Hardware Request Status Register (DMA_HRSn) to ensure there is no service request to the DMA channel being suspended. Then disable the hardware service request by clearing the ERQ bit on appropriate DMA channel.

17.6.8.2 Resume a DMA channel

To resume a DMA channel:

1. Enable the DMA service request on the appropriate channel by setting the its ERQ bit.
2. Enable the DMA service request at the peripheral.

For example, assume the SPI is set as a master for transmitting data via a DMA service request when the SPI_TXFIFO has an empty slot. The DMA will transfer the next command and data to the TXFIFO upon the request. If the user needs to suspend the DMA/SPI transfer loop, perform the following steps:

1. Disable the DMA service request at the source by writing 0 to SPI_RSER[TFFF_RE]. Confirm that SPI_RSER[TFFF_RE] is 0.

2. Ensure there is no DMA service request from the SPI by verifying that DMA_HRS[HRS*n*] is 0 for the appropriate channel. If no service request is present, disable the DMA channel by clearing the channel's ERQ bit. If a service request is present, wait until the request has been processed and the HRS bit reads zero.

Chapter 18
Enhanced Secured Digital Host Controller (eSDHC)

18.1 The eSDHC module as implemented on the chip

This section provides details about how the eSDHC module is implemented on the chip.

18.1.1 LS1012A eSDHC module integration

The following table describes the eSDHC module integration into this chip:

Table 18-1. eSDHC module integration

Module	Module Base address
eSDHC controller 1 (eSDHC1)	156_0000
eSDHC controller 2 (eSDHC2)	158_0000

The remainder of this chapter refers to a single eSDHC module.

18.1.2 LS1012A eSDHC signals

The following table lists the SoC signal names and their corresponding eSDHC module signal names used in this chapter:

Table 18-2. LS1012A eSDHC signals

LS1012A signal name	eSDHC module signal
SDHC1_CLK / SDHC2_CLK	SDHC_CLK
SDHC1_CMD / SDHC2_CMD	SDHC_CMD
SDHC1_DAT[3:0] / SDHC2_DAT[3:0]	SDHC_DAT[3:0]
SDHC1_CD_B	SDHC_CD_B
SDHC1_WP	SDHC_WP

Table continues on the next page...

Table 18-2. LS1012A eSDHC signals (continued)

LS1012A signal name	eSDHC module signal
SDHC1_VSEL	SDHC_VS

NOTE

The eSDHC controller 2 does not have the SDHC_CD_B, SDHC_WP, and SDHC_VS signals; these signals are only available in eSDHC controller 1.

18.1.3 LS1012A eSDHC module special consideration

The chip supports two eSDHC controllers with different capabilities. The eSDHC module implements the following parameter settings in the chip:

Table 18-3. LS1012A eSDHC parameter settings

eSDHC parameters	LS1012A parameter value	
	eSDHC1	eSDHC2
SD card support	Yes	No
Voltage requirement	3.3V, 1.8V (for eMMC only)	1.8V
Card detect	Yes	No
Write protect	Yes	No
SD memory card	Yes	No
SDIO card	Yes	No
eSDIO	No	Yes
MMC card	No	No
eMMC DDR	Yes	Yes
eMMC FS/HS/HS200	Yes	Yes

NOTE

- Though eSDHC1 boots at 3.3V, but it can dynamically switch to 1.8V controlled by the SDHC_VSEL signal. However, for eMMC 1.8V, the IO voltage can be set to 1.8V by default.
- There is no card detect pin for SDHC2 interface since only embedded devices are supported (eSDIO, eMMC). Software needs to assume card is always present.

18.2 Overview

The enhanced secured digital host controller (eSDHC) provides interface between the host system and the SD/SDIO cards and eMMC devices, as depicted in the figure below.

The eSDHC acts as a bridge, passing host bus transactions to the SD/SDIO cards and eMMC devices by sending commands and performing data accesses to/from the cards.

It handles the SD/SDIO/eMMC protocols at the transmission level.

Different types of cards supported by the eSDHC are described below:

- Secure digital (SD) card

  The secure digital (SD) card is an evolution over the old MMC technology. It is specifically designed to meet the security, capacity, performance, and environmental requirements inherent in the emerging audio and video consumer electronic devices. The physical form factor, pin assignments, and data transfer protocol are forward-compatible with the old MMC. The chip supports SDXC card.

- SDIO

  Under the SD protocol, the SD cards can be categorized as a memory card, I/O card, or combo card. The memory card invokes a copyright protection mechanism that complies with the security of the SDMI standard. The I/O card provides high-speed data I/O with low power consumption for mobile electronic devices. The combo card has both memory and I/O functions. For the sake of simplicity, the following figure does not show cards with reduced sizes.

In addition, the chip also supports the embedded devices, such as eMMC and eSDIO. Note that all references to SDIO are also applicable to eSDIO.

The eSDHC acts as a bridge, passing host bus transactions to SD/SDIO cards by sending commands and performing data accesses to or from the cards. It handles the SD/SDIO protocol at the transmission level. The figure below shows connection of the eSDHC.

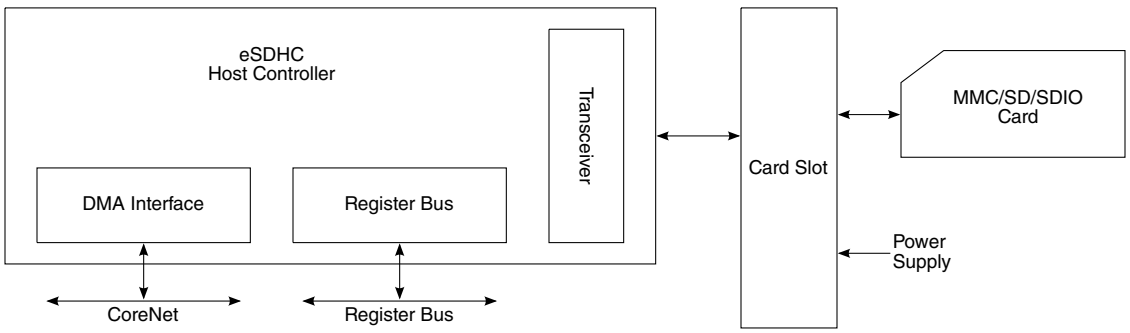


Figure 18-1. System connection of the eSDHC

NOTE

The card slot is not valid for embedded devices.

The figure below is a high-level block diagram of eSDHC.

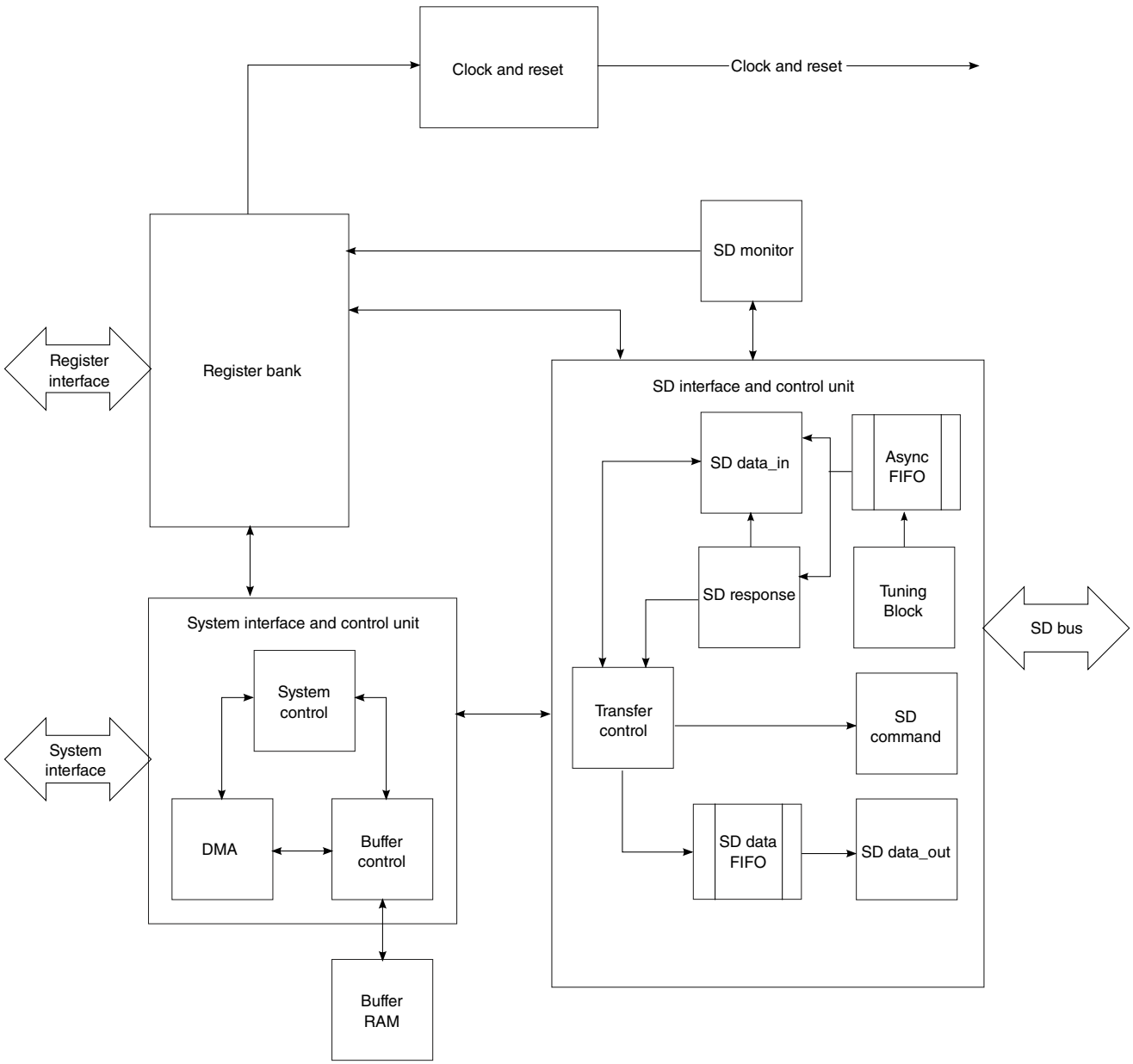


Figure 18-2. Enhanced secure digital host controller block diagram

18.2.1 eSDHC features summary

The features of the eSDHC module include the following:

- Conforms to the SD Host controller standard specification version 3.0, including test event register support
- Compatible with the eMMC system specification version 4.5
- Compatible with the SD memory card specification version 3.01, and supports the high capacity SD memory card

- Compatible with the SDIO card specification version 3.0
- Designed to work with SD memory, SDIO, SD combo, and their variants, such as mini and micro.
- eMMC high capacity support
- Supports 1- or 4-bit SD and SDIO modes, 1- or 4-bit eMMC modes
- Supports single-block and multi-block read, and write data transfer
- Supports block sizes of 1-2048 bytes
- Supports the mechanical write protect detection. In the case where write protect is enabled, the host will not initiate any write data command to the card
- Supports card detection from SDHC_CD_B
- 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 and Auto CMD23 for multi-block transfer
- Host can initiate command that does not use data lines, while data transfer is in progress
- Allows card to interrupt the host in 1 bit and 4 bit SDIO modes, also supports interrupt period
- Embodies a configurable 128 x 32 bit FIFO for read/write data
- Supports SDMA, ADMA1, and ADMA2 capabilities
- Host will send 80 idle SD clock cycles to card, which are needed during card power-up, if bit INITA in the system control register (SYSCTL) is set
- Supports SDIO asynchronous interrupt
- Supports clock divider with finer granularity, that is, with values 1,2,3....1024 or 1,2,4,8...2048
- Supports standard, high and extended capacity card types
- Supports SD UHS-1 speed modes: SDR12, SDR25, SDR50, SDR104, DDR50
- Supports eMMC high speed, HS200 and DDR mode

18.2.2 Modes and operations

18.2.2.1 Data transfer modes

The eSDHC can select the following modes for data transfer:

- SD 1 bit
- SD 4 bit
- eMMC 1 bit
- eMMC 4 bit
- Identification mode

- eMMC full speed mode
- eMMC high speed mode
- SD/SDIO full speed mode
- SD/SDIO high speed mode
- SD/SDIO UHS-1 SDR12 mode
- SD/SDIO UHS-1 SDR25 mode
- SD/SDIO UHS-1 SDR50 mode
- SD/SDIO UHS-1 SDR104 mode
- SD/SDIO UHS-1 DDR50 mode
- eMMC HS200 mode
- eMMC DDR mode

NOTE

For the maximum supported speed of the modes, refer chip-specific datasheet.

18.3 External signals

18.3.1 External signals overview

The eSDHC has the following associated I/O signals:

- SDHC_CLK is an internally generated clock signal that drives the SD/SDIO card or eMMC device.
- SDHC_CLK_SYNC_OUT has same frequency as SDHC_CLK. It should be sent out on board for loop back from close to external card.
- SDHC_CLK_SYNC_IN has same frequency as SDHC_CLK. It is loopback input from close to card on board.
- SDHC_CMD I/O sends commands to and receives responses from the card.
- SDHC_DAT[3:0] performs data transfers between the eSDHC and the card.
- SDHC_CD_B and SDHC_WP are card detection and write protection signals from the socket.
- SDHC_VS is an output signal used to control the voltage of external SD bus supply.

18.3.2 eSDHC signal descriptions

The following table describes eSDHC signals.

Table 18-4. Signal properties

Name	Port	Function	Reset state	Pull up
SDHC_CLK	O	Clock for SD/SDIO card or eMMC device	0	N/A
SDHC_CLK_SYNC_OUT	O	Clock to loopback from close to card	0	N/A
SDHC_CLK_SYNC_IN	I	Clock looped back from card	0	N/A
SDHC_CMD	I/O	CMD line connect to card	1	Pull up
SDHC_DAT3	I/O	DAT3 line in 4-bit mode DAT3 line in 1-bit mode	0	Pull up. Do not use DAT3 pin as a CD pin.
SDHC_DAT2	I/O	DAT2 line or read wait in 4-bit mode Read wait in 1-bit mode	1	Pull up
SDHC_DAT1	I/O	DAT1 line in 4-bit mode Also used to detect interrupt in 1/4-bit mode	1	Pull up
SDHC_DAT0	I/O	DAT0 line in all modes Also used to detect busy state	1	Pull up
SDHC_CD_B	I	Card detection pin If not used tie high	N/A	N/A
SDHC_WP	I	Card write protect detect If not used tie low	N/A	N/A
SDHC_VS	O	Control the voltage on SD pads to be high voltage (around 3.0V) or low voltage (around 1.8V). '0' stands for high voltage range Optional output	0	N/A

18.4 eSDHC register descriptions

The table below shows the memory-mapped registers of the eSDHC module and lists the offset, name, and a cross-reference to the complete description of each register. These register only support 32-bit accesses.

18.4.1 eSDHC Memory map

ESDHC1 base address: 156_0000h

ESDHC2 base address: 158_0000h

Offset	Register	Width (In bits)	Access	Reset value
0h	SDMA system address register/Block attributes 2 (DSADDR_BLKATTR2)	32	RW	0000_0000h
4h	Block attributes register (BLKATTR)	32	RW	0000_0000h
8h	Command argument register (CMDARG)	32	RW	0000_0000h
Ch	Transfer type register (XFERTYP)	32	RW	0000_0000h
10h	Command response 0 register (CMDRSP0)	32	RO	0000_0000h
14h	Command response 1 register (CMDRSP1)	32	RO	0000_0000h
18h	Command response 2 register (CMDRSP2)	32	RO	0000_0000h
1Ch	Command Response 3 register (CMDRSP3)	32	RO	0000_0000h
20h	Buffer data port register (DATPORT)	32	RW	0000_0000h
24h	Present state register (PRSSTAT)	32	RO	See description.
28h	Protocol control register (PROCTL)	32	RW	0000_0020h
2Ch	System Control Register when ESDHCCTL[CRS=0] (SYSCTL_ESDHCCTL_CRS_0)	32	RW	0000_8038h
2Ch	System Control Register when ESDHCCTL[CRS=1] (SYSCTL_ESDHCCTL_CRS_1)	32	RW	See description.
30h	Interrupt status register (IRQSTAT)	32	W1C	0000_0000h
34h	Interrupt status enable register (IRQSTATEN)	32	RW	377F_11FFh
38h	Interrupt signal enable register (IRQSIGEN)	32	RW	0400_1000h
3Ch	Auto CMD Error Status Register / System Control 2 Register (AUTOCERR_SYSCTL2)	32	RW	0000_0000h
40h	Host controller capabilities register (HOSTCAPBLT)	32	RO	See description.
44h	Watermark level register (WML)	32	RW	0010_0010h
50h	Force event register (FEVT)	32	WO	0000_0000h
54h	ADMA error status register (ADMAES)	32	RO	0000_0000h
58h	ADMA system address register (ADSADDR)	32	RW	0000_0000h
FCh	Host controller version register (HOSTVER)	32	RO	0000_2102h
104h	DMA error address register (DMAERRADDR)	32	RO	0000_0000h
10Ch	DMA error attribute register (DMAERRATTR)	32	RO	0000_0000h
114h	Host controller capabilities register 2 (HOSTCAPBLT2)	32	RO	See description.
120h	Tuning block control register (TBCTL)	32	RW	See description.
124h	Tuning block status register (TBSTAT)	32	RW	0000_0000h
128h	Tuning block pointer register (TBPTR)	32	RW	0000_0000h
140h	SD direction control register (SDDIRCTL)	32	RW	0000_0001h
144h	SD Clock Control Register (SDCLKCTL)	32	RW	0000_0000h
40Ch	eSDHC control register (ESDHCCTL)	32	RW	0000_0000h

18.4.2 SDMA system address register/Block attributes 2 (DSADDR_BLKATTR2)

18.4.2.1 Offset

Register	Offset
DSADDR_BLKATTR2	0h

18.4.2.2 Function

The DSADDR contains the physical system memory address used for single DMA transfers or second argument for Auto CMD23.

18.4.2.3 Diagram

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

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

18.4.2.4 Fields

Field	Function
0-31 DS_ADDR	DMA system address / Block attributes 2 This register contains the physical system memory address used for DMA transfers or second argument for Auto CMD23. SDMA system address:

Field	Function
	This register contains the 32-bit system memory address for a single DMA (SDMA) transfer. When the eSDHC 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 (that is, after a transaction has stopped). The host driver should initialize this register before starting a every DMA transaction. After DMA has stopped, the system address of the next contiguous data position can be read from this register. The eSDHC DMA does not support a virtual memory system. It only supports continuous physical memory access. Block attributes 2: This register is used with Auto CMD23 to set 32-bit block count value to the argument of CMD23 while executing Auto CMD23. If Auto CMD23 is used with ADMA, the full 32-bit block count value can be used. If Auto CMD23 is used without ADMA, the available block count value is limited by 16-bit Block Count field in Block Attributes register. 65536 blocks is the maximum value available in this case.

18.4.3 Block attributes register (BLKATTR)

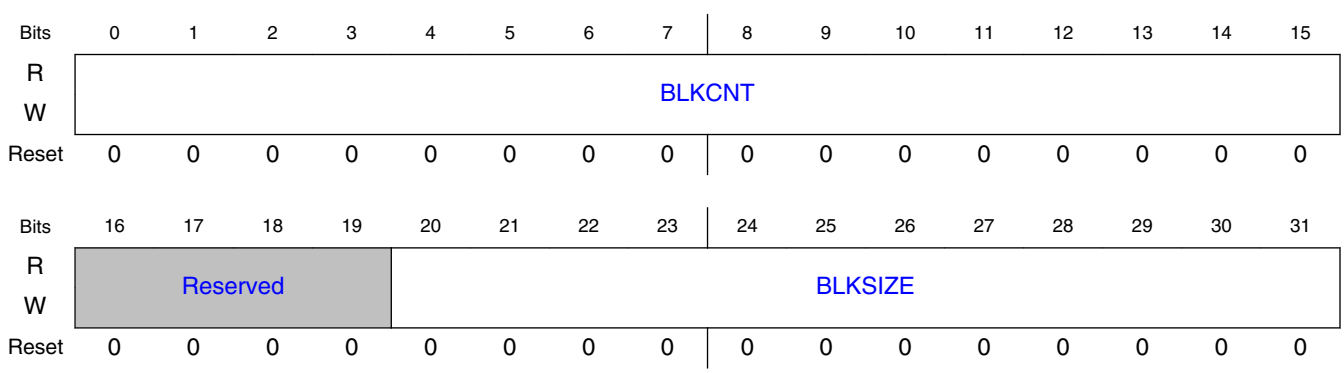
18.4.3.1 Offset

Register	Offset
BLKATTR	4h

18.4.3.2 Function

The BLKATTR is used to configure the number of data blocks and the number of bytes in each block.

18.4.3.3 Diagram



18.4.3.4 Fields

Field	Function
0-15 BLKCNT	Blocks count for current transfer. • This register is enabled when XFERTYP[BCEN] is set to 1 and is valid only for multiple block transfers. The host driver should set this register to a value between 1 and the maximum block count. The eSDHC 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 (that is, 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, eSDHC will regard the current transfer is aborted and change BLKCNT 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 should restore the previously saved block count. 0000000000000000b - Stop count 0000000000000001b - 1 block 0000000000000010b - 2 blocks 1111111111111111b - 65535 blocks
16-19 —	Reserved.
20-31 BLKSIZE	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 (that is, after a transaction has stopped). Read operations during transfers may return an invalid value, and write operations will be ignored. 000000000000b - No data transfer 000000000001b - 1 byte 000000000010b - 2 bytes 000000000011b - 3 bytes 000000000100b - 4 bytes 000111111111b - 511 bytes 001000000000b - 512 bytes 100000000000b - 2048 bytes

18.4.4 Command argument register (CMDARG)

18.4.4.1 Offset

Register	Offset
CMDARG	8h

18.4.4.2 Function

The CMDARG contains the SD/eMMC command argument.

18.4.4.3 Diagram

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

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

18.4.4.4 Fields

Field	Function
0-31 CMDARG	Command argument. The SD/eMMC command argument is specified as bits 39-8 of the command format in the SD or eMMC Specification. This register is write protected when PRSSTAT[CIHB] is set.

18.4.5 Transfer type register (XFERTYP)

18.4.5.1 Offset

Register	Offset
XFERTYP	Ch

18.4.5.2 Function

The host driver should set the XFERTYP to issue any new command. To prevent data loss, the eSDHC prevents writing to the bits, that are involved in the data transfer of this register, when data transfer is active: MBSEL, DTDSEL, AC12EN, BCEN, and DMAEN.

The host driver should check PRSSTAT[CDIHB] and PRSSTAT[CIHB] before writing to this register. When PRSSTAT[CDIHB] is set, any attempt to send a command with data by writing to this register is ignored; when PRSSTAT[CIHB] is set, any write to this register is ignored.

On sending commands with data transfer, it is mandatory that the block size is non-zero. Besides, block count must also be non-zero, or indicated as single block transfer (XFERTYP[MSBSEL] is '0' when written), or block count is disabled (XFERTYP[BCEN] is '0' when written), otherwise eSDHC will ignore the sending of this command and do nothing. For write commands, with all above restrictions, it is also mandatory that the write protect switch is not active (PRSSTAT[WPS] is '1), otherwise eSDHC will also ignore the command.

If the commands with write data transfer does not receive the response in 64 clock cycles, that is, response time-out, eSDHC will regard the external device does not accept the command and will not initiate the data transfer on SD bus. In this scenario, the driver should perform error recovery and issue the command again to re-try the transfer.

It is also possible that for some reason the card responds to the read data command but eSDHC does not receive the response, and if it is a DMA (SDMA or ADMA) read operation, the external system memory is over-written by the DMA with data sent back from the card.

Table 18-5. 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

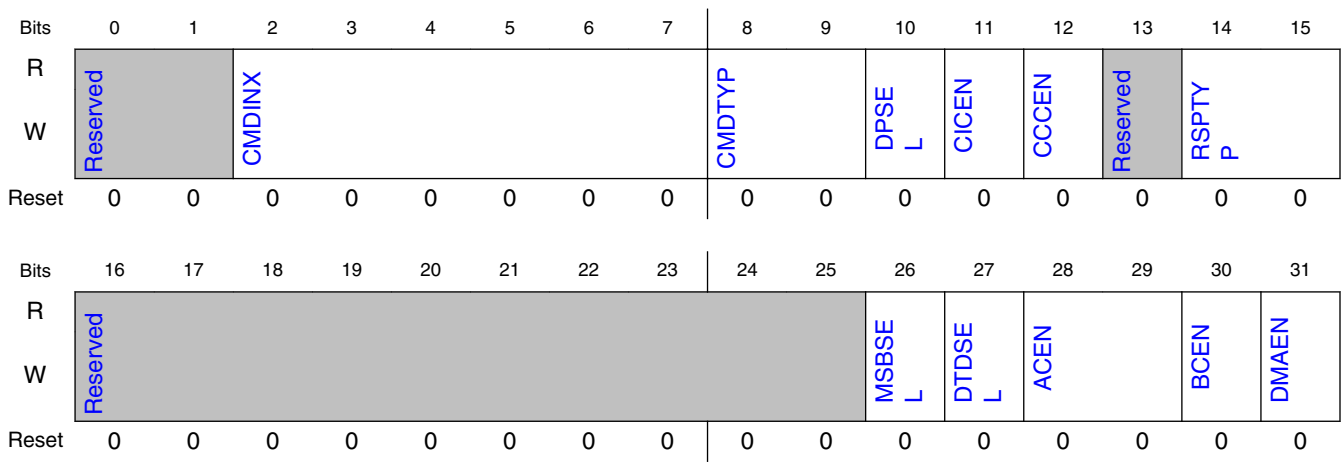
The table below shows the relationship between the command index check enable and the command CRC check enable, in regards to the response type bits as well as the name of the response type.

Table 18-6. 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, R7
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 eSDHC checks the busy status after receiving a response. For example, usually CMD52 is used with R5, but the I/O abort command should be used with R5b.
- The CRC field for R3 and R4 is expected to be all 1 bits. The CRC check should be disabled for these response types.

18.4.5.3 Diagram



18.4.5.4 Fields

Field	Function
0-1 —	Reserved.
2-7 CMDINX	Command index

Table continues on the next page...

Field	Function
	Command index. These bits should 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 .
8-9 CMDTYP	Command Type Command Type. There are three types of special commands: suspend, resume, and abort. These bits should be set to 00b for all other commands. • Suspend command: If the suspend command succeeds, the eSDHC should assume that the card bus has been released and that it is possible to issue the next command which uses the DAT line. Since eSDHC 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 eSDHC that a suspend command was successfully issued. Refer to Suspend resume for more details. After the start bit of command is sent, the eSDHC 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 eSDHC will maintain its current state, and the host driver should restart the transfer by setting PROCTL[CREQ]. • 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. • Abort command: If this command is set when executing a read transfer, the eSDHC will stop write to the buffer after end bit of abort command is sent. If this command is set when executing a write transfer, the eSDHC will stop driving the DAT line. After issuing the abort command, the host driver should issue a software reset (abort transaction). 00b - Normal other commands 01b - Suspend CMD52 for writing bus suspend in CCCR 10b - Resume CMD52 for writing function select in CCCR 11b - Abort CMD12, CMD52 for writing I/O abort in CCCR
10 DPSEL	Data present select. This bit is set to 1 to indicate that data is present and should be transferred using the DAT line. It is set to 0 for the following: • Commands using only the CMD line (for example, CMD52) • Commands with no data transfer, but using the busy signal on DAT[0] line (R1b or R5b, for example, CMD38) **NOTE:** In resume command, this bit should be set, and other bits in this register should be set the same as when the transfer was initially launched. When the write protect switch is on (that is, PRSSTAT[WPS] is active as '0'), any command with a write operation will be ignored. That is to say, when this bit is set, while XFRTYP[DTDSEL] is 0, writes to the transfer type register (XFRTYP) are ignored. 0b - No data present 1b - Data present
11 CICEN	Command index check enable. If this bit is set to 1, the eSDHC checks 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. 0b - Disable 1b - Enable
12 CCCEN	Command CRC check enable Command CRC check enable. If this bit is set to 1, the eSDHC should 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 Table 18-6.) 0b - Disable 1b - Enable
13	Reserved.

Table continues on the next page...

Field	Function
—	
14-15 RSPTYP	Response type select. 00b - No response 01b - Response length 136 10b - Response length 48 11b - Response length 48, check busy after response
16-25 —	Reserved.
26 MSBSEL	Multi-/single-block select Multi-/single-block select. This bit enables multiple block DAT line data transfers. For any other commands, this bit should be set to 0. If this bit is 0, it is not necessary to set the block count register. (Refer to Table 18-5.) 0b - Single block 1b - Multiple blocks
27 DTDSEL	Data transfer direction select. This bit defines the direction of DAT line data transfers. The bit is set to 1 by the host driver to transfer data from the SD card to the eSDHC and is set to 0 for all other commands. 0b - Write (host to card) 1b - Read (card to host)
28-29 ACEN	Auto CMD12 enable Auto CMD12 enable. Multiple block transfers for memory require a CMD12 to stop the transaction. When this bit is set to 1, the eSDHC will issue a CMD12 automatically when the last block transfer has completed. The host driver should 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 do not require CMD12. In single block transfer, the eSDHC will ignore this bit whether if it is set or not. Auto CMD23 Enable. When this bit is set to 10b, the eSDHC will issue a CMD23 automatically before issuing a command specified in the transfer type register. The following conditions are required to use Auto CMD23: • A memory card that supports CMD23 (For SD card, SCR[33]=1) • If DMA is used, it shall be ADMA • Only when CMD18 or CMD25 is issued **NOTE:** eSDHC does not check command index Auto CMD23 can be issued with or without ADMA, by writing Transfer Type register, eSDHC issues a CMD23 first and then issues a command specified by Transfer Type register. If response errors of CMD23 are detected, the second command is not issued. A CMD23 error is indicated in the Auto CMD Error Status register. 32-bit block count value for CMD23 is set to DMA System Address / Argument 2 register. 00b - Auto CMD Disable 01b - Auto CMD12 Enable 10b - Auto CMD23 Enable 11b - Reserved
30 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. If ADMA data transfer is more than 65535 blocks, this bit shall be set to 0. In this case, data transfer length is designated by Descriptor Table. 0b - Disable 1b - Enable

Table continues on the next page...

Field	Function
31 DMAEN	DMA enable DMA enable. This bit enables DMA functionality. If this bit is set to 1, a DMA operation should begin when the host driver sets the DPSEL bit of this register. Whether the single DMA or ADMA is active depends on PROCTL[DMAS]. 0b - Disable 1b - Enable

18.4.6 Command response 0 register (CMDRSP0)

18.4.6.1 Offset

Register	Offset
CMDRSP0	10h

18.4.6.2 Function

The CMDRSP0 is used to store part 0 of the response bits from the card.

18.4.6.3 Diagram

Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CMDRSP0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CMDRSP0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

18.4.6.4 Fields

Field	Function
0-31 CMDRSP0	Command response 0 Command response 0. Refer to Command Response 3 register (CMDRSP3) for the mapping of command responses from the SD bus to this register for each response type.

18.4.7 Command response 1 register (CMDRSP1)

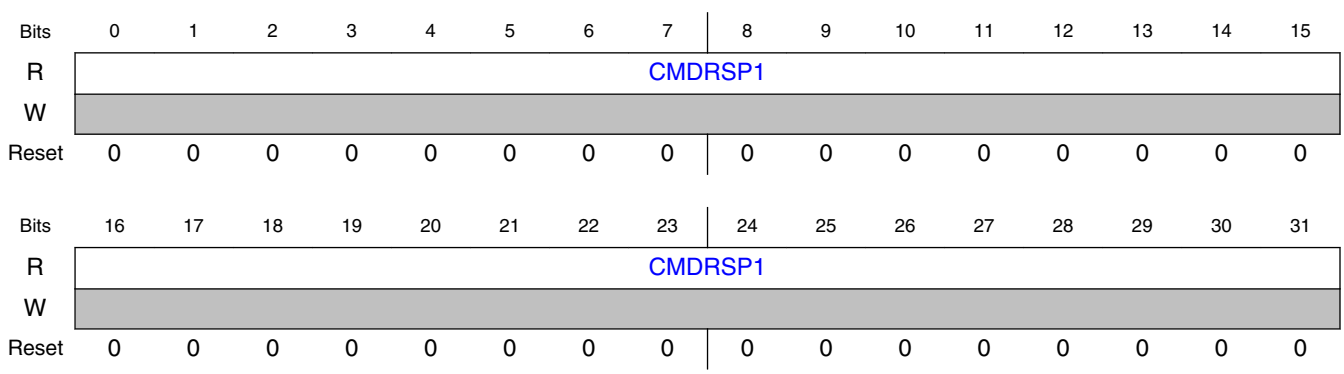
18.4.7.1 Offset

Register	Offset
CMDRSP1	14h

18.4.7.2 Function

The CMDRSP1 is used to store part 1 of the response bits from the card.

18.4.7.3 Diagram



18.4.7.4 Fields

Field	Function
0-31 CMDRSP1	Command response 1 Command response 1. Refer to Command Response 3 register (CMDRSP3) for the mapping of command responses from the SD bus to this register for each response type.

18.4.8 Command response 2 register (CMDRSP2)

18.4.8.1 Offset

Register	Offset
CMDRSP2	18h

18.4.8.2 Function

The CMDRSP2 is used to store part 2 of the response bits from the card.

18.4.8.3 Diagram

Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CMDRSP2															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CMDRSP2															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

18.4.8.4 Fields

Field	Function
0-31 CMDRSP2	Command response 2 Command response 2. Refer to Command Response 3 register (CMDRSP3) for the mapping of command responses from the SD bus to this register for each response type.

18.4.9 Command Response 3 register (CMDRSP3)

18.4.9.1 Offset

Register	Offset
CMDRSP3	1Ch

18.4.9.2 Function

The CMDRSP3 is used to store part 3 of the response bits from the card.

Table below describes the mapping of command responses from the SD bus to command response registers for each response type. In the table, R *n* refers to a bit range within the response data as transmitted on the SD bus.

Table 18-7. 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[8:31], CMDRSP2, CMDRSP1, CMDRSP0}
R3 (OCR register)	OCR register for memory	R[39:8]	CMDRSP0
R4 (OCR register)	OCR register for I/O	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 the following:

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 eSDHC only stores part of the response data in the command response registers (CMDRSP *n*). 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 eSDHC (as specified by the command index check enable and the command CRC check enable bits in the transfer type register, XFERTYP) 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 eSDHC checks R[47:1], and if the response length is 136 the eSDHC checks R[119:1].

Since eSDHC may have a multiple block data transfer executing concurrently with a CMD_wo_DAT command, it stores the Auto CMD12 response in the CMDRSP3 register. The CMD_wo_DAT response is stored in CMDRSP0. This allows the eSDHC to avoid overwriting the Auto CMD12 response with the CMD_wo_DAT and vice versa. When the eSDHC modifies part of the command response registers (CMDRSP *n*), as shown in the table above, it preserves the unmodified bits.

18.4.9.3 Diagram

Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CMDRSP3															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CMDRSP3															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

18.4.9.4 Fields

Field	Function
0-31 CMDRSP3	Command response 3 Command response 3. Refer to Command Response 3 register (CMDRSP3) for the mapping of command responses from the SD bus to this register for each response type.

18.4.10 Buffer data port register (DATPORT)

18.4.10.1 Offset

Register	Offset
DATPORT	20h

18.4.10.2 Function

The DATPORT is a 32-bit data port register used to access the internal buffer. Byte access is not allowed.

18.4.10.3 Diagram

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

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

18.4.10.4 Fields

Field	Function
0-31 DATCONT	Data content. The buffer data port register is for 32-bit data access by the CPU. When the DMA is enabled, any write to this register is ignored, and any read from this register always yields zeros.

18.4.11 Present state register (PRSSTAT)

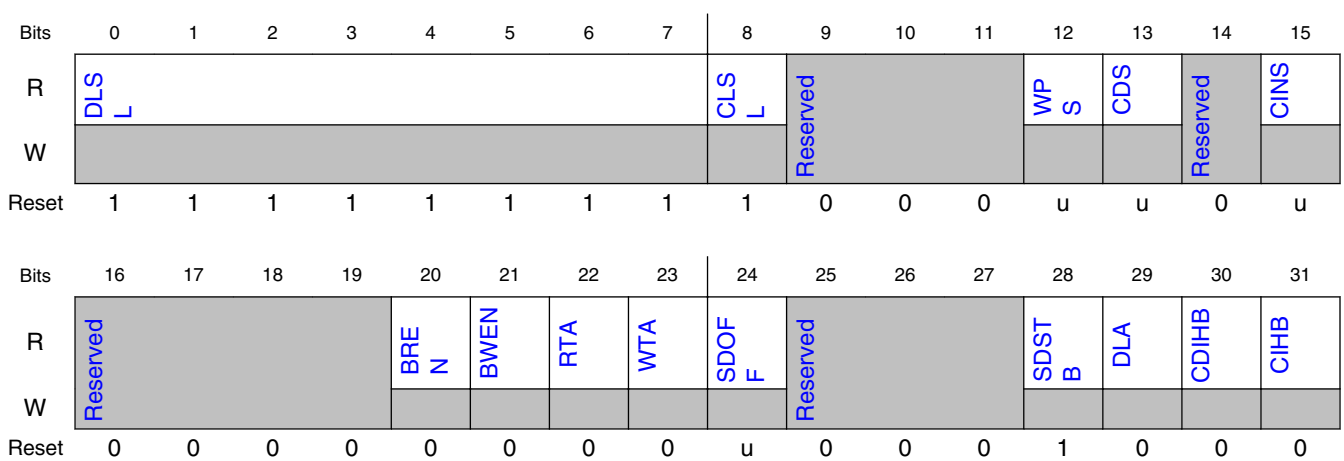
18.4.11.1 Offset

Register	Offset
PRSSTAT	24h

18.4.11.2 Function

The host driver can get the status of the eSDHC from the PRSSTAT, which is a 32-bit, read-only register.

18.4.11.3 Diagram



18.4.11.4 Fields

Field	Function
0-7 DLSL	DAT[7:0] line signal level DAT[7:0] line signal level. This status is used to check the DAT line level to recover from errors, and for debugging.This is especially useful in detecting the busy signal level from DAT[0]. The reset value is effected by the external pull-up/pull-down resistors. By default, the read value of this bit field after reset is 8'b11111111 DLSL[0]: Data 0 line signal level DLSL[1]: Data 1 line signal level DLSL[2]: Data 2 line signal level DLSL[3]: Data 3 line signal level DLSL[4]: Reserved DLSL[5]:Reserved DLSL[6]: Reserved DLSL[7]: Reserved
8 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 effected by the external pull-up/pull-down resistor, by default, the read value of this bit after reset is 1, when the command line is pulled up.
9-11 —	Reserved.
12 WPS	Write protect state Write protect state. The write protect switch is supported for memory and combo cards. This bit reflects the write protect state depending on value of SDHC_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 SDHC_WP pin is not used, it should be tied low, so that the reset value of this bit is high and write is enabled. 0b - Write protected (SDHC_WP =1) 1b - Write enabled (SDHC_WP =0)
13 CDS	Card detect state Card detect state. This bit reflects the card inserted/removed state depending on value of the SDHC_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 (FEVT) does not affect this bit. The reset value is effected by the external card detection pin. This bit shows the value on the SDHC_CD_B pin (that is, when a card is inserted in the socket, it is 0 on the SDHC_CD_B input, and consequently the CDS reads 1.) 0b - No card present (SDHC_CD_B =1) 1b - Card present SDHC_CD_B =0)
14 —	Reserved.
15 CINS	Card inserted. This bit indicates whether a card has been inserted. The eSDHC 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 (IRQSTAT). Changing from a 1 to 0 generates a card removal interrupt in the interrupt status register (IRQSTAT). A write to the force event register (FEVT) does not effect this bit.

Table continues on the next page...

Field	Function
	The software reset for all in the system control register (SYSCTL) does not effect this bit. A software reset does not effect this bit. 0b - Power on reset or no card 1b - Card inserted
16-19 —	Reserved.
20 BREN	Buffer read enable. This status bit is used for non-DMA read transfers. The eSDHC may implement multiple buffers 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 any read from the buffer is made. 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. 0b - Read disable 1b - Read enable
21 BWEN	Buffer write enable. This status bit is used for non-DMA write transfers. The eSDHC can implement multiple buffers 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 any write to the buffer is made. 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. 0b - Write disable 1b - Write enable

22
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 read operation is restarted by writing a 1 to PROCTL[CREQ].
	• 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.
	  • In the case of ADMA2, end of read operation is designated by descriptor table.
	  • When all valid data blocks in the host controller have been transferred 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.
	A transfer complete interrupt is generated when this bit changes to 0.
	0b - No valid data
	1b - Transferring data

23
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 eSDHC.
	• This bit is set in either of the following cases:
	  • After the end bit of the write command.
	  • When write operation is restarted by writing a 1 to PROCTL[CREQ].
	• 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). In case of ADMA2, transfer count is designated by descriptor table.
	  • 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.
	0b - No valid data

Table continues on the next page...

Field	Function
	1b - Transferring data
24 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. 0b - SD clock is active 1b - SD clock is gated off
25-27 —	Reserved.
28 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 SYSCTL[SDCLKEN] to remove glitch on the card clock when the frequency is changing. 0b - Clock is changing frequency and not stable 1b - Clock is stable

29
DLA
	Data line active. This status bit indicates whether one of the DAT 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 (IRQSTAT).
	• This bit will be set in either of the following cases:
	  • After the end bit of the read command.
	  • When writing a 1 to PROCTL[CREQ] to restart a read transfer.
	• This bit should be cleared in either of the following cases:
	  • When the end bit of the last data block is sent from the SD bus to the host controller. In case of ADMA2, the last block is designated by the last transfer of descriptor table.
	  • When a read transfer is stopped at the block gap initiated by a stop at block gap request.
	• The eSDHC 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 eSDHC 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 (IRQSTAT).
	• This bit will be set in either of the following cases:
	  • After the end bit of the write command.
	  • When writing to 1 to PROCTL[CREQ] 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. If SD card does not drive busy signal for 8 SD Clocks, the host controller should consider the card drive "Not Busy". In case of ADMA2, the last block is designated by the last transfer of descriptor table.
	  • 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:
	• Command with busy. This status indicates whether a command indicates busy (for example, erase command for memory) is executing on the SD bus. This bit is set after the end bit of the command with busy and cleared when busy is de-asserted.
	Changing this bit from 1 to 0 generate a transfer complete interrupt in the interrupt status register (IRQSTAT).
	0b - DAT line inactive
	1b - DAT line active

Table continues on the next page...

Field	Function
30 CDIHB	Command inhibit (DAT). 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 the eSDHC can issue the next SD/eMMC Command. Commands with busy signal belong to command inhibit (DAT) (for example, R1b, R5b type). Changing from 1 to 0 generates a transfer complete interrupt in the interrupt status register (IRQSTAT). **NOTE:** The SD host driver can save registers for a suspend transaction after this bit has changed from 1 to 0. 0b - Can issue command which uses the DAT line 1b - Cannot issue command which uses the DAT line
31 CIHB	Command inhibit (CMD). If this status bit is 0, it indicates that the CMD line is not in use and the eSDHC can issue a SD/eMMC Command using the CMD line. This bit is set also immediately after the transfer type register (XFERTYP) is written. This bit is cleared when the command response is received. Even if the command inhibit (DAT) 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 (IRQSTAT). If the eSDHC 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. 0b - Can issue command using only CMD line 1b - Cannot issue command

18.4.12 Protocol control register (PROCTL)

18.4.12.1 Offset

Register	Offset
PROCTL	28h

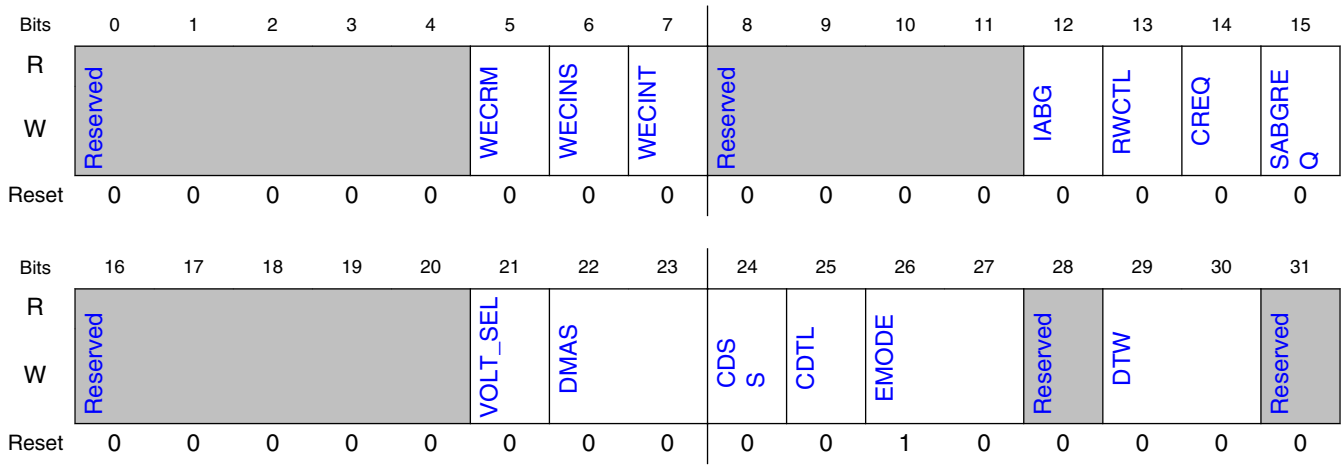
18.4.12.2 Function

There are three cases to restart the transfer after stop at the block gap. Which case is appropriate depends on whether the eSDHC 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 should be used to restart the transfer.
2. If the host driver issues a suspend command and the SD card accepts it, a resume command should 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 should be used to restart the transfer.

Any time a stop at block gap request stops the data transfer, the host driver should wait for a transfer complete (in the interrupt status register, IRQSTAT), before attempting to restart the transfer. When restarting the data transfer by continue request, the host driver should clear the stop at block gap request before or simultaneously.

18.4.12.3 Diagram



18.4.12.4 Fields

Field	Function
0-4 —	Reserved.
5 WECRM	Wakeup event enable on SD card removal Wakeup event enable on SD card removal. This bit enables a wakeup event, via a card removal, in the interrupt status register (IRQSTAT). FN_WUS (wake up support) in CIS of SDIO card does not effect this bit. When this bit is set, the card removal status and the eSDHC interrupt can be asserted without SDHC_CLK toggling. When the wakeup feature is not enabled, the SDHC_CLK must be active in order to assert the card removal status and the eSDHC interrupt. 0b - Disable 1b - Enable
6 WECINS	Wakeup event enable on SD card insertion Wakeup event enable on SD card insertion. This bit enables a wakeup event, via a card insertion, in the interrupt status register (IRQSTAT). FN_WUS (wake up support) in CIS of the SDIO card does not affect this bit. When this bit is set, the card insertion status and the eSDHC interrupt can be asserted without SDHC_CLK toggling. When the wakeup feature is not enabled, the SDHC_CLK must be active in order to assert the card insertion status and the eSDHC interrupt. 0b - Disable 1b - Enable
7	Wakeup event enable on card interrupt

Table continues on the next page...

Field	Function
WECINT	Wakeup event enable on card interrupt. This bit enables a wakeup event, via a card interrupt, in the interrupt status register (IRQSTAT). This bit can be set to 1 if FN_WUS (wake up support) in CIS of SDIO card is set to 1. When this bit is set, the card interrupt status and the eSDHC interrupt can be asserted without SDHC_CLK toggling. When the wakeup feature is not enabled, the SDHC_CLK must be active in order to assert the card interrupt status and the eSDHC interrupt. 0b - Disable 1b - Enable
8-11 —	Reserved.
12 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 cannot 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 should set this bit according to the CCCR of the card. 0b - Disable 1b - Enable
13 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 DAT[2] line. Otherwise the eSDHC has to stop the SD Clock to hold read data, which restricts commands generation. When the host driver detects an SDIO card insertion, it should set this bit according to the CCCR of the card. If the card does not support read wait, this bit should never be set to 1, otherwise DAT line conflicts may occur. If this bit is set to 0, stop at block gap during read operation is also supported, but the eSDHC will stop the SD clock to pause reading operation. 0b - Disable read wait control, and stop SD clock at block gap when SABGREQ bit is set 1b - Enable read wait control, and assert read wait without stopping SD clock at block gap when SABGREQ bit is set
14 CREQ	Continue request. This bit is used to restart a transaction, which was stopped using the stop at block gap request. To cancel stop at the block gap, set stop at block gap request to 0 and set this bit 1 to restart the transfer. The eSDHC automatically clears this bit in either of the following cases: • In the case of a read transaction, the DAT line active changes from 0 to 1 as a read transaction restarts. • In the case of a write transaction, the write transfer active changes from 0 to 1 as the write transaction restarts. Therefore, it is not necessary for host driver to set this bit to 0. If stop at block gap request is set to 1, any write to this bit is ignored. 0b - No effect 1b - Restart
15 SABGREQ	Stop at block gap request Stop at block gap request. 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 should 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 eSDHC 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 should 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 eSDHC 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 should set this bit

Table continues on the next page...

Field	Function
	after all block data is written. If this bit is set to 1, the host driver should not write data to the data port register after a block is sent. Once this bit is set, the host driver should not clear this bit before the transfer complete bit in (IRQSTAT)is set, otherwise the eSDHCs behavior is undefined. This bit effects read transfer active, write transfer active, PRSSTAT[DLA] and PRSSTAT[CDIHB]. 0b - Transfer 1b - Stop
16-20 —	Reserved.
21 VOLT_SEL	Voltage selection Voltage selection. Change the value of output signal SDHC_VS , to control the SD bus supply voltage for external card. There must be a control circuit out of eSDHC to change the voltage. 0b - Change the SD Bus Supply voltage to high voltage range, around 3.0V 1b - Change the SD bus supply voltage to low voltage range, around 1.8V
22-23 DMAS	DMA select DMA select. This field is valid while DMA (SDMA or ADMA) is enabled and selects the DMA operation. 00b - Single DMA is selected 01b - ADMA1 is selected 10b - 32-bit address ADMA2 is selected 11b - Reserved
24 CDSS	Card detect signal selection. This bit selects the source for the card detection. 0b - SD CD pin is selected (for normal purposes) 1b - Card detection test level is selected (for test purposes)
25 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. 0b - Card detect test level is 0, no card inserted 1b - Card detect test level is 1, card inserted
26-27 EMODE	Endian mode. The eSDHC supports little and big endian mode for transferring between buffer port register and data buffer. 00b - Big endian mode 01b - Reserved 10b - Little endian mode 11b - Reserved
28 —	Reserved.
29-30 DTW	Data transfer width. This bit selects the data width of the SD bus for a data transfer. The host driver should set it to match the data width of the card. Possible Data transfer widths are 1-bit, 4-bits, and 8-bits. 00b - 1-bit mode 01b - 4-bit mode 10b - 8-bit mode 11b - Reserved
31 —	Reserved.

18.4.13 System Control Register when ESDHCCTL[CRS=0] (SYSCTL_ESDHCCTL_CRS_0)

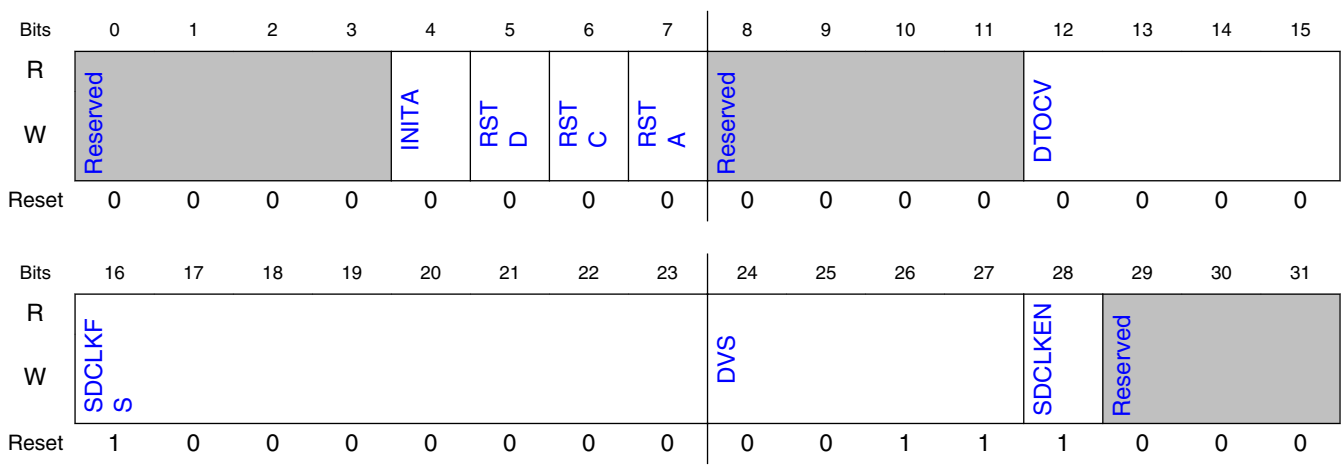
18.4.13.1 Offset

Register	Offset
SYSCTL_ESDHCCTL_CRS_0	2Ch

18.4.13.2 Function

The clock divider mode(16-27 bits in this register) is selected according to Clock Register Select field in eSDHC Control register. Other bits of the register remain unaffected by clock register select value.

18.4.13.3 Diagram



18.4.13.4 Fields

Field	Function
0-3 —	Reserved.

Table continues on the next page...

Field	Function
4 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. 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 PRSSTAT[CIHB] or PRSSTAT[CDIHB] are set, writing 1 to this bit is ignored (that is, when command line or data lines are active, write to this bit is not allowed). On the other hand, when this bit is set, that is, 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.

5
RSTD
	Software reset for DAT line. Only part of the data circuit is reset. DMA circuit is also reset.
	The following registers and bits are cleared by this bit:
	• Data port register
	  • Buffer is cleared and initialized
	• Present state register (PRSSTAT)
	  • Buffer read enable
	  • Buffer write enable
	  • Read transfer active
	  • Write transfer active
	  • DAT line active
	  • Command inhibit (DAT)
	• Protocol control register (PROCTL)
	  • Continue request
	  • Stop at block gap request
	• Interrupt status register (IRQSTAT)
	  • Buffer read ready
	  • Buffer write ready
	  • DMA interrupt
	  • Block gap event
	  • Transfer complete
	NOTE: This bit will be self cleared by eSDHC when Software Reset for Data is complete, so Software should poll for it to be cleared after setting it.
	0b - No reset
	1b - Reset

6 RSTC	Software reset for CMD line. Only part of the command circuit is reset. The following registers and bits are cleared by this bit: • PRSSTAT[CIHB] • IRQSTAT[CC] **NOTE:** This bit will be self cleared by eSDHC when Software Reset for Command is complete, so software should poll for it to be cleared after setting it. 0b - No reset 1b - Reset
7 RSTA	Software reset for all. This reset effects the entire host controller except for the card detection circuit. Register bits of type R, RW, RW1C are cleared. During its initialization, the host driver should set this bit to 1 to reset the eSDHC. The eSDHC should 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. **NOTE:** The Software Reset For All in the System Control register clears Card Insertion and Card Removal bits in Interrupt Status Register. Software should issue partial reset(Reset for Command and Reset for Data) instead of Reset for All, if it wants to reset eSDHC without clearing these Interrupt Status Register bits.

Table continues on the next page...

Field	Function
	This bit will be self cleared by eSDHC when Software Reset for All is complete, so Software should poll for it to be cleared after setting it. 0b - No reset 1b - Reset
8-11 —	Reserved.
12-15 DTOCV	Data timeout counter value 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 enable register (IRQSTATEN) for information on factors that dictate time-out generation. Time-out clock frequency will be generated by dividing the SDCLK value by this value. The host driver can clear IRQSTATEN[DTOESEN] to prevent inadvertent time-out events. 0000b - SDCLK x 2^{13} 0001b - SDCLK x 2^{14} 1110b - SDCLK x 2^{27} 1111b - Reserved
16-23 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. Base clock can be selected by programming ESDHCCTL[PCS]. It selects between platform clock and peripheral clock / 2 . Setting 0x00 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 the base clock and the following divisor bits. The frequency of SDCLK is set by the following formula: Clock Frequency = (Base Clock) / (prescaler x divisor) For example, if the base clock frequency is 96 MHz and the target frequency is 25 MHz, then choosing the prescaler value of 0x01 and divisor value of 0x1 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 0x08 and divisor value of 0xE yields the exact clock value of 400 KHz. The reset value of this bit field is 0x80, so if the input base clock is about 96 MHz, the default SD clock after reset is 375 KHz. The programmed SD Clock frequency shall never exceed maximum SD clock supported by the card. **NOTE:** Both DVS and SDCLKFS fields should not be programmed 0 simultaneously. Only the following settings are allowed: 00000000b - Base clock 00000001b - Base clock divided by 2 00000010b - Base clock divided by 4 00000100b - Base clock divided by 8 00001000b - Base clock divided by 16 00010000b - Base clock divided by 32 00100000b - Base clock divided by 64 01000000b - Base clock divided by 128 10000000b - Base clock divided by 256
24-27 DVS	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 divisor without deterioration of duty cycle. The settings are as following: 0000b - Divisor by 1

Table continues on the next page...

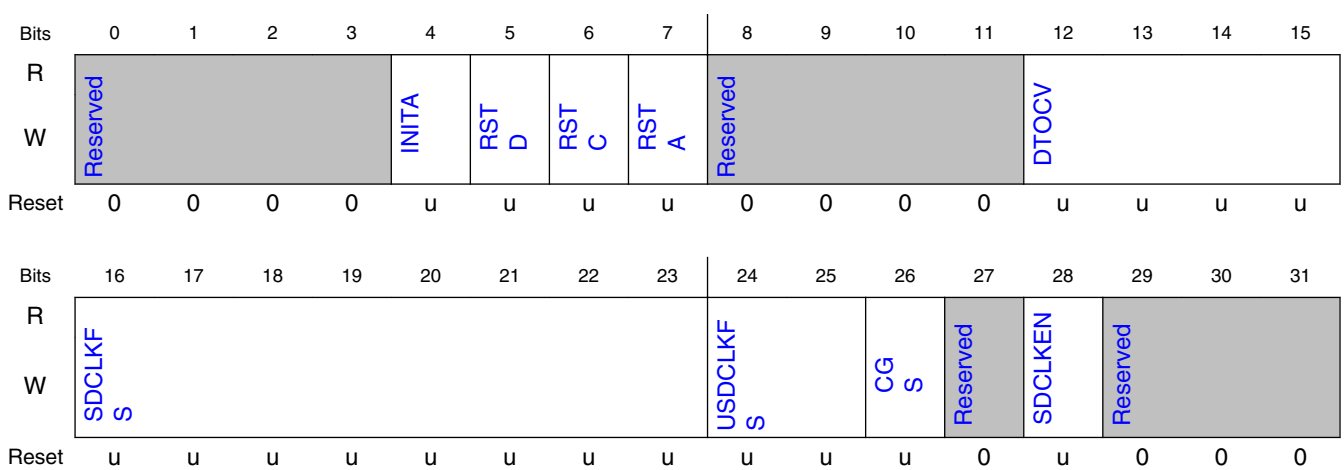
Field	Function
	0001b - Divisor by 2 ... 1110b - Divisor by 15 1111b - Divisor by 16
28 SDCLKEN	SD clock enable. the host controller should stop SDCLK when writing this bit to 0. SDCLK frequency can be changed when this bit is 0. Then, the host controller should maintain the same clock frequency until SDCLK is stopped (stop at SDCLK=0). If PRSSTAT[CINS] is cleared, this bit should be cleared by the host driver to save power.
29-31 —	Reserved.

18.4.14 System Control Register when ESDHCCTL[CRS=1] (SYSCTL_ESDHCCTL_CRS_1)

18.4.14.1 Offset

Register	Offset
SYSCTL_ESDHCCTL_CRS_1	2Ch

18.4.14.2 Diagram



18.4.14.3 Fields

Field	Function
0-3 —	Reserved.
4 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. 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 PRSSTAT[CIHB] or PRSSTAT[CDIHB] are set, writing 1 to this bit is ignored (that is, when command line or data lines are active, write to this bit is not allowed). On the other hand, when this bit is set, that is, 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.

5
RSTD
	Software reset for DAT line. Only part of the data circuit is reset. DMA circuit is also reset.
	The following registers and bits are cleared by this bit:
	• Data port register
	  • Buffer is cleared and initialized
	• Present state register (PRSSTAT)
	  • Buffer read enable
	  • Buffer write enable
	  • Read transfer active
	  • Write transfer active
	  • DAT line active
	  • Command inhibit (DAT)
	• Protocol control register (PROCTL)
	  • Continue request
	  • Stop at block gap request
	• Interrupt status register (IRQSTAT)
	  • Buffer read ready
	  • Buffer write ready
	  • DMA interrupt
	  • Block gap event
	  • Transfer complete
	NOTE: This bit will be self cleared by eSDHC when Software Reset for Data is complete, so Software should poll for it to be cleared after setting it.
	0b - No reset
	1b - Reset

6 RSTC	Software reset for CMD line. Only part of the command circuit is reset. The following registers and bits are cleared by this bit: • PRSSTAT[CIHB] • IRQSTAT[CC] **NOTE:** This bit will be self cleared by eSDHC when Software Reset for Command is complete, so software should poll for it to be cleared after setting it. 0b - No reset 1b - Reset
7 RSTA	Software reset for all. This reset effects the entire host controller except for the card detection circuit. Register bits of type R, RW, RW1C are cleared. During its initialization, the host driver should set this bit to 1 to reset the eSDHC. The eSDHC should reset this bit to 0 when the capabilities registers are valid

Table continues on the next page...

Field	Function
	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. **NOTE:** The software reset for all in the system control register clears card insertion and card removal bits in interrupt status register. Software should issue partial reset (reset for command and reset for data) instead of reset for all, if it wants to reset eSDHC without clearing these interrupt status register bits. This bit will be self cleared by eSDHC when software reset for all is complete, so software should poll for it to be cleared after setting it. 0b - No reset 1b - Reset
8-11 —	Reserved.
12-15 DTOCV	Data timeout counter value 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 enable register (IRQSTATEN) for information on factors that dictate time-out generation. Time-out clock frequency will be generated by dividing the SDCLK value by this value. The host driver can clear IRQSTATEN[DTOESEN] to prevent inadvertent time-out events. 0000b - SDCLK x 2^{13} 0001b - SDCLK x 2^{14} 1110b - SDCLK x 2^{27} 1111b - Reserved
16-23 SDCLKFS	SDCLK Frequency Select: This register is used to select the frequency of the SDCLK pin. 10-bit divisor value is formed by concatenating USDCLKFS(2-bit) and SDCLKFS(8-bit), that is, Divisor = {USDCLKFS[0:1], SDCLKFS[0:7]} Following Divisor definition is selected depending on Clock Generation Select value: **10-bit Divided Clock Mode** 0x3FF Base clock divided by 2048 0x04 Base clock divided by 8 0x03 Base clock divided by 6 0x02 Base clock divided by 4 0x01 Base clock divided by 2 0x00 Reserved **10-bit Programmable Clock Mode** 0x3FF Base clock divided by 1024 0x04 Base clock divided by 5 0x03 Base clock divided by 4 0x02 Base clock divided by 3 0x01 Base clock divided by 2 0x00 Reserved The frequency of SDCLK is set by the following formula: Clock Frequency = (Base Clock) / (divisor)

Table continues on the next page...

Field	Function
24-25 USDCLKFS	Upper bits of SDCLK frequency select. This field is used to expand SDCLKFS to 10 bits. These two bits occupies most significant portion of 10-bit SDCLKFS.
26 CGS	Clock Generator Select. This field selects 10-bit SDCLKFS clock mode. 0b - Divided clock mode is selected 1b - Programmable clock mode is selected
27 —	Reserved.
28 SDCLKEN	SD clock enable. The host controller shall stop SDCLK when writing this bit to 0. SDCLK frequency can be changed when this bit is 0. Then, the host controller shall maintain the same clock frequency until SDCLK is stopped (Stop at SDCLK=0). If the card inserted in the present state register is cleared, this bit should be cleared by the host driver to save power.
29-31 —	Reserved.

18.4.15 Interrupt status register (IRQSTAT)

18.4.15.1 Offset

Register	Offset
IRQSTAT	30h

18.4.15.2 Function

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 most 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

Table 18-8. eSDHC 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

Table 18-9. eSDHC 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

Table 18-10. eSDHC Status for Command CRC Error/Command Timeout Error Bit Combinations

Command CRC Error	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

18.4.15.3 Diagram

Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved		RTOE	DMAE	Reserved	TNE	ADMAE	AC12E	Reserved	DEBE	DCE	DTOE	CIE	CEBE	CCE	CTOE
W			W1C	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

Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved			RTE	Reserved			CINT	CRM	CINS	BRR	BWR	DINT	BGE	TC	CC
W				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

18.4.15.4 Fields

Field	Function
0-1 —	Reserved.
2	Register access timeout error. This bit indicate that register access has timed-out. 0b - No timeout error

Table continues on the next page...

Field	Function
RTOE	1b - Timeout error
3 DMAE	DMA error DMA error. This bit indicates that DMA (SDMA or ADMA) transfer has failed. 0b - No error 1b - Error
4 —	Reserved.
5 TNE	Tuning Error. This bit is set when an unrecoverable error is detected in a tuning circuit except during tuning procedure (Occurrence of an error during tuning procedure is indicated by Sampling Clock Select). By detecting Tuning Error, Host Driver needs to abort a command executing and perform tuning. The Tuning Error is higher priority than the other error interrupts generated during data transfer. By detecting Tuning Error, the Host Driver should discard data transferred by a current read/write command and retry data transfer after the Host Controller retrieved from tuning circuit error. This bit might not be set in some cases, but SD command or Data error might be set; Driver should consider it as tuning circuit error and perform tuning procedure. 0b - No error 1b - Error
6 ADMAE	ADMA error. This bit is set when the host controller detects errors during ADMA operation. The state of the ADMA at an error occurrence is saved in the ADMA error status register. 0b - No error 1b - Error
7 AC12E	Auto CMD12 error. Occurs when detecting that one of the bits in the Auto CMD12 error status register (AUTOC12ERR) 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. 0b - No error 1b - Error
8 —	Reserved.
9 DEBE	Data end bit error. Occurs either when detecting 0 at the end bit position of read data, which uses the DAT line, or at the end bit position of the CRC. 0b - No error 1b - Error
10 DCE	Data CRC error. Occurs when detecting a CRC error when transferring read data, which uses the DAT line, or when detecting the write CRC status having a value other than 010. 0b - No error 1b - Error
11 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 • Write CRC status time-out • Read data time-out 0b - No error 1b - Time out
12 CIE	Command index error. Occurs if a command index error occurs in the command response. 0b - No error 1b - Error
13	Command end bit error. Occurs when detecting that the end bit of a command response is 0.

Table continues on the next page...

Field	Function
CEBE	0b - No error 1b - End bit error generated
14 CCE	Command CRC error. A 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 eSDHC detects a CMD line conflict by monitoring the CMD line when a command is issued. If the eSDHC drives the CMD line to 1, but detects 0 on the CMD line at the next SDCLK edge, then the eSDHC should abort the command (stop driving CMD line) and set this bit to 1. The command timeout error should also be set to 1 to distinguish CMD line conflict. 0b - No error 1b - CRC error generated
15 CTOE	Command timeout error Command timeout error. Occurs only if no response is returned within 64 SDCLK cycles from the end bit of the command. If the eSDHC detects a CMD line conflict, in which case a command CRC error should also be set (as shown in), this bit should be set without waiting for 64 SDCLK cycles. This is because the command will be aborted by the eSDHC. 0b - No error 1b - Time out
16-18 —	Reserved.
19 RTE	Re-tuning event. This status is set if re-tuning request in the eSDHC control register changes from 0 to 1. eSDHC requests host driver to perform re-tuning for next data transfer. Current data transfer (not large block count) can be completed without re-tuning. 0b - Re-tuning is not required 1b - Re-tuning should be performed
20-22 —	Reserved.
23 CINT	Card interrupt. Writing this bit to 1 does not clear this bit. It is cleared by resetting the SD card interrupt factor. In 1-bit mode, the host controller should detect the card interrupt without SD clock to support wakeup. In 4-bit mode, the card interrupt signal is sampled during the interrupt cycle, so there are some sample delays between the interrupt signal from the SD card and the interrupt to the host system. It is necessary to define how to handle this delay. When this status has been set and the host driver needs to start this interrupt service, IRQSTATEN[CINTSEN] should be set to 0 in order to clear the card interrupt statuses latched in the eSDHC and to stop driving the interrupt signal to the host system. After completion of the card interrupt service (It should reset interrupt factors in the SD card and the interrupt signal may not be asserted), set the card interrupt status enable bit to 1 and start sampling the interrupt signal again. 0b - No card interrupt 1b - Generate card interrupt
24 CRM	Card removal. This status is set if the PRSSTAT[CINS] changes from 1 to 0. When the host driver writes this bit to 1 to clear this status, the status of the PRSSTAT[CINS] should be confirmed. Because the card detect state may possibly be changed when the host driver clear this bit and interrupt event may not be generated. **NOTE:** The Software Reset For All in the System Control register clears this bit. Software should issue partial reset(Reset for Command and Reset for Data) instead of Reset for All, if it wants to reset eSDHC without clearing this bit. eSDHC does not implement de-bouncing circuit on card detect pin, so Software should check Card Inserted in Present State register to confirm card insertion/removal status. 0b - Card state unstable or inserted

Table continues on the next page...

Field	Function
	1b - Card removed
25 CINS	Card insertion. This status is set if PRSSTAT[CINS] changes from 0 to 1. When the host driver writes this bit to 1 to clear this status, the status of the PRSSTAT[CINS] should be confirmed. Because the card detect state may possibly be changed when the host driver clear this bit and interrupt event may not be generated. **NOTE:** The Software Reset For All in the System Control register clears this bit. Software should issue partial reset(Reset for Command and Reset for Data) instead of Reset for All, if it wants to reset eSDHC without clearing this bit. eSDHC does not implement de-bouncing circuit on card detect pin, so Software should check Card Inserted in Present State register to confirm card insertion/removal status. 0b - Card state unstable or removed 1b - Card inserted
26 BRR	Buffer read ready Buffer read ready. This status bit is set if PRSSTAT[BREN] changes from 0 to 1. Refer to the description of the buffer read enable bit in Present state register (PRSSTAT) for additional information. 0b - Not ready to read buffer 1b - Ready to read buffer
27 BWR	Buffer write ready Buffer write ready. This status bit is set if the PRSSTAT[BWEN] changes from 0 to 1. Refer to the description of the buffer write enable bit in Present state register (PRSSTAT) for additional information. 0b - Not ready to write buffer 1b - Ready to write buffer
28 DINT	DMA interrupt DMA interrupt. Occurs only when the 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 single DMA or ADMA finishes data transferring, this bit will be set. 0b - No DMA Interrupt 1b - DMA Interrupt is generated
29 BGE	Block gap event. If PROCTL[SABGREQ] is set, this bit is set when a read or write transaction is stopped at a block gap. If PROCTL[SABGREQ] 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 DAT 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 a write transaction, this bit is set at the falling edge of write transfer active status (after getting CRC status at SD Bus timing). 0b - No block gap event 1b - Transaction stopped at block gap
30 TC	Transfer complete. This bit is set when a read/write transfer and a command with busy is completed. While performing tuning procedure (Execute Tuning is set to 1), Transfer Complete is not set for CMD19 execution. 0b - Transfer not complete 1b - Transfer complete
31 CC	Command complete. This bit is set when the end bit of the command response is received (except Auto CMD12). Refer to PRSSTAT[CIHB]. 0b - Command not complete 1b - Command complete

18.4.16 Interrupt status enable register (IRQSTATEN)

18.4.16.1 Offset

Register	Offset
IRQSTATEN	34h

18.4.16.2 Function

Setting the bits to 1 in the IRQSTATEN 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 (that is, when the bit in this register is cleared, the corresponding bit in interrupt status register, IRQSTAT, is always 0).

NOTE

- Depending on how PROCTL[IABG] is set, eSDHC 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 IRSTAT[CTOESEN] and IRSTAT[CCESEN] to 1.

18.4.16.3 Diagram

Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved		RTOESEN	DMAESEN	Reserved	TNESEN	ADMAESEN	AC12ESEN	Reserved	DEBESEN	DCESEN	DTOESEN	CIESEN	CEBESEN	CCESEN	CTOESEN
W																
Reset	0	0	1	1	0	1	1	1	0	1	1	1	1	1	1	1

Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved			RTESEN	Reserved			CINTSEN	CRMSEN	CINSEN	BRRSEN	BWRSEN	DINTSEN	BGESEN	TCSEN	CCSEN
W																
Reset	0	0	0	1	0	0	0	1	1	1	1	1	1	1	1	1

18.4.16.4 Fields

Field	Function
0-1 —	Reserved.
2 RTOESEN	Register access timeout status enable 0b - Masked 1b - Enabled
3 DMAESEN	DMA error status enable. 0b - Masked 1b - Enabled
4 —	Reserved.
5 TNESEN	Tuning error status enable 0b - Masked 1b - Enabled
6 ADMAESEN	ADMA error status enable. 0b - Masked 1b - Enabled
7 AC12ESEN	Auto CMD12 error status enable. 0b - Masked 1b - Enabled
8 —	Reserved.
9 DEBESEN	Data end bit error status enable. 0b - Masked 1b - Enabled
10 DCESEN	Data CRC error status enable. 0b - Masked 1b - Enabled
11 DTOESEN	Data timeout error status enable. 0b - Masked 1b - Enabled
12 CIESEN	Command index error status enable. 0b - Masked 1b - Enabled
13 CEBESEN	Command end bit error status enable. 0b - Masked 1b - Enabled
14 CCESEN	Command CRC error status enable. 0b - Masked 1b - Enabled

Table continues on the next page...

Field	Function
15 CTOESEN	Command timeout error status enable. 0b - Masked 1b - Enabled
16-18 —	Reserved.
19 RTESEN	Re-tuning event status enable. 0b - Masked 1b - Enabled
20-22 —	Reserved.
23 CINTSEN	Card interrupt status enable. If this bit is set to 0, the eSDHC 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. 0b - Masked 1b - Enabled
24 CRMSEN	Card removal status enable. 0b - Masked 1b - Enabled
25 CINSEN	Card insertion status enable. 0b - Masked 1b - Enabled
26 BRRSEN	Buffer read ready status enable. 0b - Masked 1b - Enabled
27 BWRSEN	Buffer write ready status enable. 0b - Masked 1b - Enabled
28 DINTSEN	DMA interrupt status enable. 0b - Masked 1b - Enabled
29 BGESEN	Block gap event status enable. 0b - Masked 1b - Enabled
30 TCSEN	Transfer complete status enable. 0b - Masked 1b - Enabled
31 CCSEN	Command complete status enable. 0b - Masked 1b - Enabled

18.4.17 Interrupt signal enable register (IRQSIGEN)

18.4.17.1 Offset

Register	Offset
IRQSIGEN	38h

18.4.17.2 Function

The IRQSIGEN 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.

18.4.17.3 Diagram

Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Reserved		RTOEIEN	DMAEIEN	Reserved	TNEIEN	ADMAEIEN	AC12EIEN	Reserved	DEBEIEN	DCEIEN	DTOEIEN	CIEIEN	CEBEIEN	CCEIEN	CTOEIEN
Reset	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Reserved			RTEIEN	Reserved			CINTIEN	CRMIEN	CINSIEN	BRRIEN	BWRIEN	DINTIEN	BGEIEN	TCIEN	CCIEN
Reset	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

18.4.17.4 Fields

Field	Function
0-1 —	Reserved.
2	Register timeout error interrupt enable 0b - Masked

Table continues on the next page...

Field	Function
RTOEIEN	1b - Enable
3 DMAEIEN	DMA error interrupt enable. 0b - Masked 1b - Enabled
4 —	Reserved.
5 TNEIEN	Tuning error interrupt enable 0b - Masked 1b - Enable
6 ADMAEIEN	ADMA Error Interrupt Enable.
7 AC12EIEN	Auto CMD12 error interrupt enable. 0b - Masked 1b - Enabled
8 —	Reserved.
9 DEBEIEN	Data end bit error interrupt enable. 0b - Masked 1b - Enabled
10 DCEIEN	Data CRC error interrupt enable. 0b - Masked 1b - Enabled
11 DTOEIEN	Data timeout error interrupt enable. 0b - Masked 1b - Enabled
12 CIEIEN	Command index error interrupt enable. 0b - Masked 1b - Enabled
13 CEBEIEN	Command end bit error interrupt enable. 0b - Masked 1b - Enabled
14 CCEIEN	Command CRC error interrupt enable. 0b - Masked 1b - Enabled
15 CTOEIEN	Command timeout error interrupt enable. 0b - Masked 1b - Enabled
16-18 —	Reserved.
19 RTEIEN	Re-tuning event interrupt enable 0b - Masked 1b - Enabled
20-22 —	Reserved.

Table continues on the next page...

Field	Function
23 CINTIEN	Card interrupt interrupt enable. 0b - Masked 1b - Enabled
24 CRMIEN	Card removal interrupt enable. 0b - Masked 1b - Enabled
25 CINSIEN	Card insertion interrupt enable. 0b - Masked 1b - Enabled
26 BRRIEN	Buffer read ready interrupt enable. 0b - Masked 1b - Enabled
27 BWRIEN	Buffer write ready interrupt enable. 0b - Masked 1b - Enabled
28 DINTIEN	DMA interrupt enable. 0b - Masked 1b - Enabled
29 BGEIEN	Block gap event interrupt enable. 0b - Masked 1b - Enabled
30 TCIEN	Transfer complete interrupt enable. 0b - Masked 1b - Enabled
31 CCIEN	Command complete interrupt enable. 0b - Masked 1b - Enabled

18.4.18 Auto CMD Error Status Register / System Control 2 Register (AUTOCERR_SYSCTL2)

18.4.18.1 Offset

Register	Offset
AUTOCERR_SYSCTL2	3Ch

18.4.18.2 Function

When the Auto CMD12 error status bit in the AUTOC12ERR is set, the host driver checks this register to identify the kind of error indicated by the Auto CMD12. This register is valid only when the auto CMD12 error status bit is set.

Table 18-11. Relationship Between Command CRC Error and Command Timeout Error for Auto CMD12

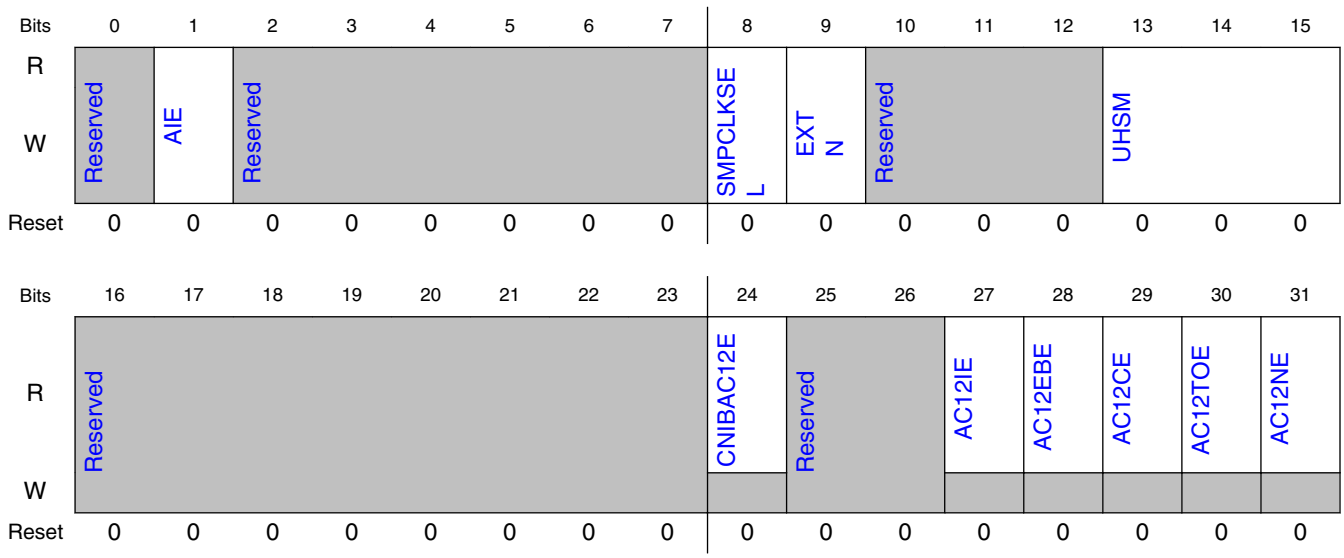
Auto CMD12 CRC Error	Auto CMD12 Timeout Error	Type of Error
0	0	No Error
0	1	Response Timeout Error
1	0	Response CRC Error
1	1	CMD line conflict

Changes in Auto CMD12 error status register (AUTOC12ERR) can be classified in three scenarios:

1. When the eSDHC is going to issue an Auto CMD12
   - Set bit 0 to 1 if the Auto CMD12 cannot 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 cannot 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 should be sampled when the driver is not writing to the command register. So it is suggested to read this register only when IRQSTAT[AC12E] 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.

18.4.18.3 Diagram



18.4.18.4 Fields

Field	Function
0 —	Reserved.
1 AIE	Asynchronous Interrupt Enable. This bit can be set to 1 if a card supports asynchronous interrupts and asynchronous interrupt support is set to 1 in capability register. If this bit is set to 1, host driver can stop SDCLK during asynchronous inrrupt period to save power. During this period, eSDHC continues to deliver the card interrupt to the system when it is assrted by card. 0b - Disabled 1b - Enabled
2-7 —	Reserved.
8 SMPCLKSEL	Sampling clock select. This bit is set by eSDHC during tuning procedure and valid after the completion of tuning (when execute tuning is cleared). Setting 1 by eSDHC means that tuning is completed successfully and setting 0 means that tuning is failed. Host driver should not write to this bit. Change of this bit is not allowed while eSDHC is receiving response or a read data block. 0b - Tuning procedure unsuccessful 1b - Tuning procedure completed successfully
9 EXTN	Execute Tuning. This bit is set to 1 by host driver to start tuning procedure and automatically cleared by eSDHC when tuning procedure is completed. The result of tuning is indicated to sampling clock select field. Tuning procedure is aborted by writing 0. 0b - Not tuned or tuning not compelted 1b - Execute tuning
10-12 —	Reserved.

Table continues on the next page...

Field	Function
13-15 UHSM	UHS mode select UHS mode select. This field is used to select one of UHS-1 modes for SD 3.0 card; and select HS200or DDR mode for eMMC device. Host driver should reset SDCLKEN in system control register before changing this field, and then set SDCLKEN again. 000b - SDR12 for SD, or max 52 MHz mode for SD 2.0 / eMMC 4.2 or older spec 001b - SDR25 for SD 010b - SDR50 for SD 011b - SDR104 for SD, HS200 for eMMC 100b - DDR 101b - Rest all the fields are reserved
16-23 —	Reserved.
24 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. 0b - No error 1b - Not Issued
25-26 —	Reserved.
27 AC12IE	Auto CMD index error. Occurs if the command index error occurs in response to a command. 0b - No error 1b - Error, the CMD index in response is not CMD12
28 AC12EBE	Auto CMD end bit error. Occurs when detecting that the end bit of command response is 0 which should be 1. 0b - No error 1b - End bit error generated
29 AC12CE	Auto CMD CRC error. Occurs when detecting a CRC error in the command response. 0b - No CRC error 1b - CRC error met in Auto CMD12 response
30 AC12TOE	Auto CMD 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. 0b - No error 1b - Time out
31 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 eSDHC 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. This bit is set to 0 when Auto CMD Error is generated by Auto CMD23. 0b - Executed 1b - Not executed

18.4.19 Host controller capabilities register (HOSTCAPBLT)

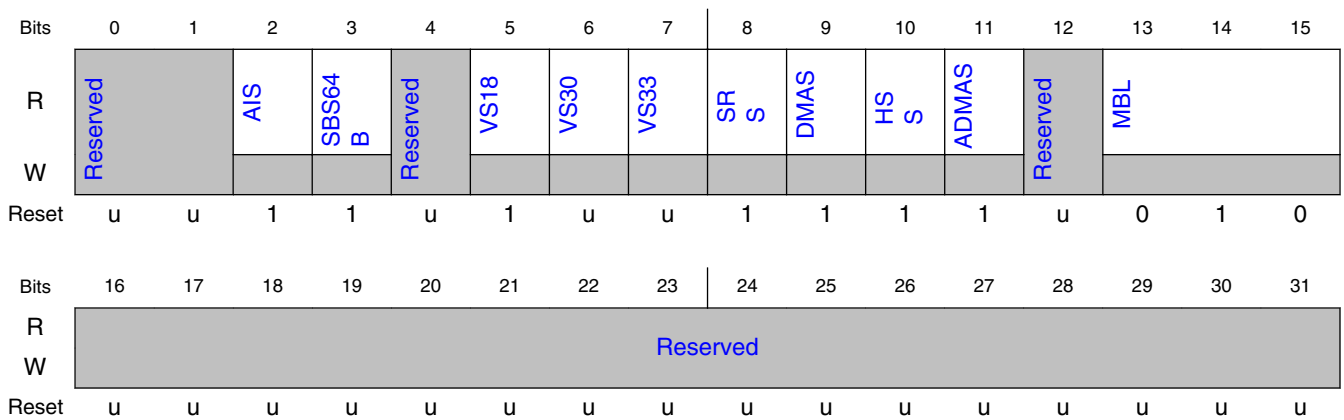
18.4.19.1 Offset

Register	Offset
HOSTCAPBLT	40h

18.4.19.2 Function

The HOSTCAPBLT provides the host driver with information specific to the eSDHC implementation. The value in this register is the power-on-reset value, and does not change with a software reset. Any write to this register is ignored.

18.4.19.3 Diagram



18.4.19.4 Fields

Field	Function
0-1 —	Reserved.
2 AIS	Asynchronous Interrupt Support. This bit indicates whether the eSDHC supports SDIO asynchronous interrupt. Refer to SDIO Specification Version 3.0 0b - Asynchronous interrupt not supported 1b - Asynchronous interrupt supported
3 SBS64B	64-bit system bus support. This bit indicates that system supports 64-bit address descriptor mode and is connected to 64-bit adress system bus. 0b - 64-bit system bus not supported 1b - 64-bit system bus supported

Table continues on the next page...

Field	Function
4 —	Reserved.
5 VS18	Voltage support 1.8V. This bit should depend on the host system ability. 0b - 1.8V not supported 1b - 1.8V supported
6 VS30	Voltage Support 3.0V. This bit shall depend on the Host System ability. 0b - 3.0V not supported 1b - 3.0V supported
7 VS33	Voltage support 3.3V. This bit should depend on the host system ability. 0b - 3.3V not supported 1b - 3.3V supported
8 SRS	Suspend/resume support. This bit indicates whether the eSDHC 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 should not issue either suspend or resume commands. 0b - Not supported 1b - Supported
9 DMAS	DMA support. This bit indicates whether the eSDHC is capable of using the DMA to transfer data between system memory and the data buffer directly. 0b - DMA not supported 1b - DMA supported
10 HSS	High speed support. This bit indicates whether the eSDHC supports high speed mode and the host system can supply a SD clock frequency from 25-50 MHz. 0b - High speed not supported 1b - High speed supported
11 ADMAS	ADMA support. This bit indicates whether the eSDHC supports the ADMA feature. 0b - Advanced DMA not supported 1b - Advanced DMA supported
12 —	Reserved.
13-15 MBL	Maximum block length. This value indicates the maximum block size that the host driver can read and write to the buffer in the eSDHC. The buffer should transfer block size without wait cycles. 000b - 512 bytes 001b - 1024 bytes 010b - 2048 bytes 011-111b - Reserved
16-31 —	Reserved.

18.4.20 Watermark level register (WML)

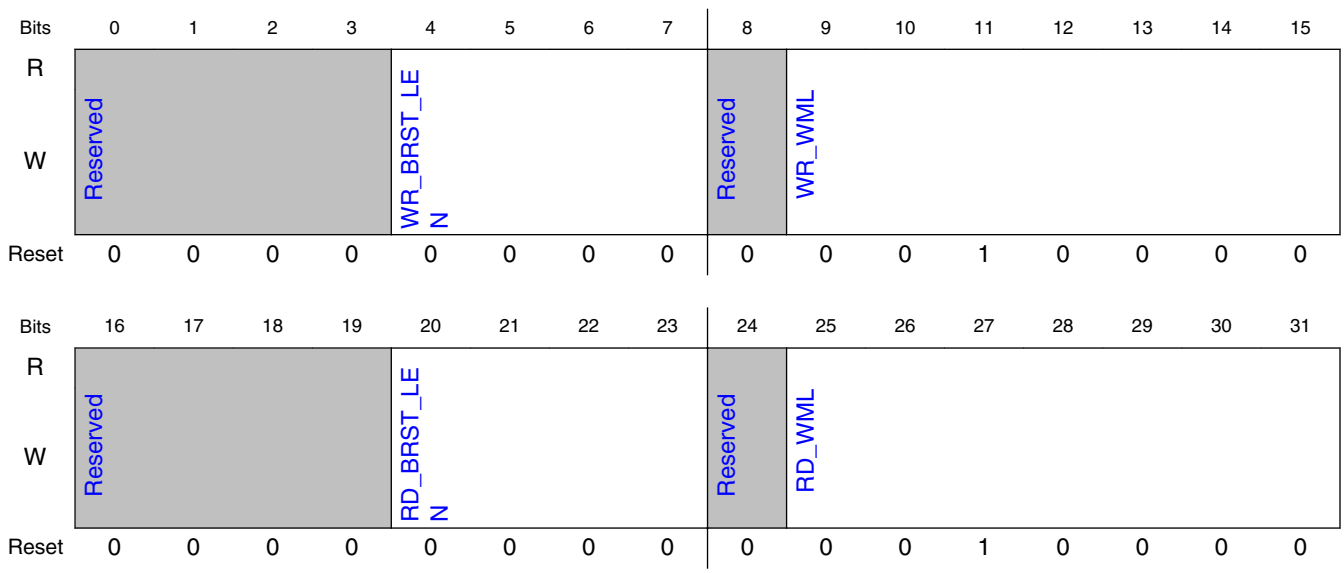
18.4.20.1 Offset

Register	Offset
WML	44h

18.4.20.2 Function

Both write and read watermark levels (FIFO threshold) are configurable in the WML. They can range from 1- 128 words. Both write and read burst lengths are also configurable. They can range from 1- 16 beats.

18.4.20.3 Diagram



18.4.20.4 Fields

Field	Function
0-3 —	Reserved.
4-7 WR_BRST_LEN	Max write burst length. Burst length desirable for write on system bus when DMA is used. This is the maximum burst length, actual burst length may be less than this depending on other factors, such as block boundary

Table continues on the next page...

Field	Function
	0000b - 16 transfers in a single burst 0001b - 1 transfer in a single burst 0010b - 2 transfers in a single burst 1111b - 15 transfers in a single burst
8 —	Reserved.
9-15 WR_WML	Write watermark level. The number of words (32-bit) used as the watermark level (FIFO threshold) for SD write in CPU polling mode. 0000000b - 128 words 0000001b - 1 word 0000010b - 2 words 1111111b - 127 words
16-19 —	Reserved.
20-23 RD_BRST_LEN	Max read burst length. Burst length desirable for read on system bus when DMA is used. This is the maximum burst length, actual burst length may be less than this depending on other factors, such as block boundary 0000b - 16 transfers in a single burst 0001b - 1 transfers in a single burst 0010b - 2 transfers in a single burst 1111b - 15 transfers in a single burst
24 —	Reserved.
25-31 RD_WML	Read watermark level. The number of words (32-bit) used as the watermark level (FIFO threshold) for SD read in CPU polling mode. 0000000b - 128 words 0000001b - 1 word 0000010b - 2 words 1111111b - 127 words

18.4.21 Force event register (FEVT)

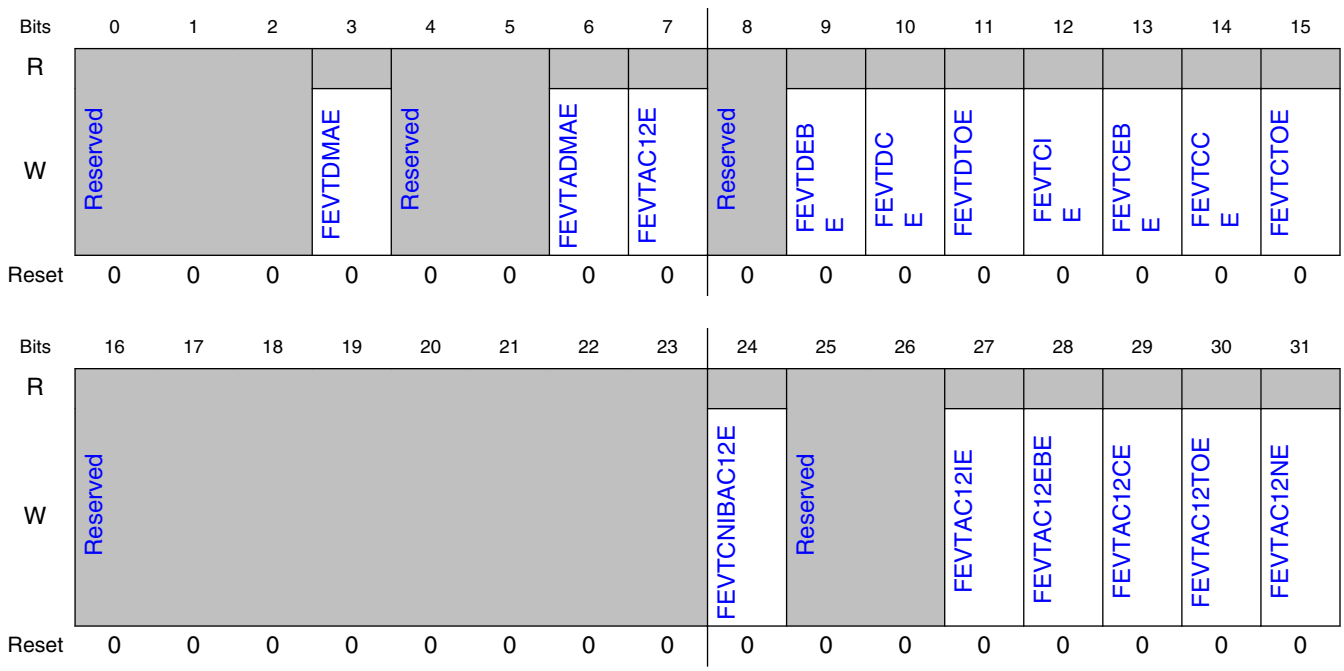
18.4.21.1 Offset

Register	Offset
FEVT	50h

18.4.21.2 Function

The FEVT is not a physically implemented register. Rather, it is an address at which the interrupt status register (IRQSTAT) can be written if the corresponding bit of the interrupt status enable register (IRQSTATEN) 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 (IRQSTAT). A read from this register always results in zeros.

18.4.21.3 Diagram



18.4.21.4 Fields

Field	Function
0-2 —	Reserved.
3 FEVTDMAE	Force event DMA error. Forces the IRQSTAT[DMAE] to be set.
4-5 —	Reserved.
6	Force event ADMA error. Forces the IRQSTAT[ADMAE] to be set.

Table continues on the next page...

Field	Function
FEVTADMAE	
7 FEVTAC12E	Force event Auto CMD12 error. Forces IRQSTAT[AC12E] to be set.
8 —	Reserved.
9 FEVTDEBE	Force event data end bit error. Forces IRQSTAT[DEBE] to be set.
10 FEVTDCE	Force event data CRC error. Forces IRQSTAT[DCE] to be set.
11 FEVTDTOE	Force event data time out error. Forces IRQSTAT[DTOE] to be set.
12 FEVTCIE	Force event command index error. Forces the IRQSTAT[CCE] to be set.
13 FEVTCEBE	Force event command end bit error. Forces IRQSTAT[CEBE] to be set.
14 FEVTCCE	Force event command CRC error. Forces IRQSTAT[CCE] to be set.
15 FEVTCTOE	Force event command time out error. Forces IRQSTAT[CTOE] to be set.
16-23 —	Reserved.
24 FEVTCNIBAC12E	Force event command not executed by Auto CMD12 error. Forces AUTOC12ERR[CNIBAC12E] to be set.
25-26 —	Reserved.
27 FEVTAC12IE	Force event Auto CMD12 index error. Forces AUTOC12ERR[AC12IE] to be set.
28 FEVTAC12EBE	Force event Auto CMD12 end bit error. Forces AUTOC12ERR[AC12EBE] to be set.
29 FEVTAC12CE	Force event Auto CMD12 CRC error. Forces AUTOC12ERR[AC12CE] to be set.
30 FEVTAC12TOE	Force event Auto CMD12 time out error. Forces the AUTOC12ERR[AC12TOE] to be set.
31 FEVTAC12NE	Force event Auto CMD12 not executed. Forces AUTOC12ERR[AC12NE] to be set.

18.4.22 ADMA error status register (ADMAES)

18.4.22.1 Offset

Register	Offset
ADMAES	54h

18.4.22.2 Function

When an ADMA error interrupt occurs, the ADMA error states field in the ADMAES holds the ADMA state and the ADMA system address register (ADSADDR) 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 (ADSADDR) is the error descriptor address
- ST_FDS: Current location set in the ADMA system address register (ADSADDR) is the error descriptor address
- ST_CADR: This state is never set because it only increments the descriptor pointer and does not generate an ADMA error
- ST_TFR: Previous location set in the ADMA system address register (ADSADDR) is the error descriptor address

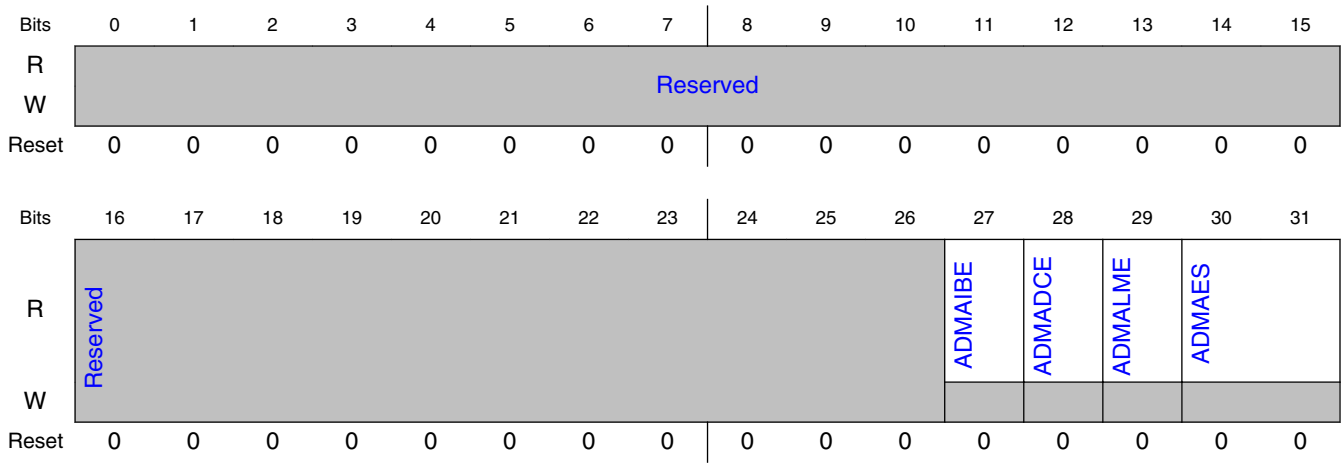
In case of a write operation, the host driver should use the ACMD22 to get the number of the written block rather than using this information, since unwritten data may exist in the host controller.

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 18-12. ADMA Error State Coding

D30-D31	ADMA Error State (When Error Has Occurred)	Contents of ADMA System Address Register
00	IDLE (idle)	Current descriptor address on which ADMA error occured
01	FETCH_DESC (fetch descriptor)	Current descriptor address on which ADMA error occured
10	DATA_XFER (data transfer)	Current descriptor address on which ADMA error occured
11	WAIT_STOP (Wait for ADMA to stop)	Current descriptor address on which ADMA error occured

18.4.22.3 Diagram



18.4.22.4 Fields

Field	Function
0-26 —	Reserved.
27 ADMAIBE	ADMA internal bus error. This bit indicates that a system bus error occurred while ADMA transaction was underway. It is set when error response is received on the system bus. 0b - No error 1b - Error
28 ADMADCE	ADMA descriptor error. This error occurs when invalid descriptor fetched by ADMA. 0b - No error 1b - Error
29 ADMALME	ADMA length mismatch error. This error occurs in the following two cases: • While XFERTYP[BCEN] 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 0b - No error 1b - Error
30-31 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 the table above for more details.

18.4.23 ADMA system address register (ADSADDR)

18.4.23.1 Offset

Register	Offset
ADSADDR	58h

18.4.23.2 Function

The ADSADDR contains the physical system memory address used for ADMA transfers.

18.4.23.3 Diagram

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

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

18.4.23.4 Fields

Field	Function
0-31 ADS_ADDR	ADMA system address. This register holds 32-bit address of executing command of the descriptor table. At the start of ADMA, the host driver should set start address of the descriptor table. The ADMA increments this register address, which points to next line, when every fetching a descriptor line. When the ADMA error interrupt is generated, this register should hold valid descriptor address depending on the ADMA state. The host driver should program descriptor table on 32-bit boundary and set 32-bit boundary address to this register. It can be accessed only when no transaction is executing (that is, after a transaction has stopped). The host driver should initialize this register before starting an ADMA transaction.

18.4.24 Host controller version register (HOSTVER)

18.4.24.1 Offset

Register	Offset
HOSTVER	FCh

18.4.24.2 Function

The HOSTVER contains the vendor host controller version information. All bits are read only and will read the same as the power-reset value.

18.4.24.3 Diagram

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

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

18.4.24.4 Fields

Field	Function
0-15 —	Reserved.
16-23 VVN	Vendor version number. These status bits are reserved for the vendor version number. The host driver should not use this status. Patterns not shown are reserved. 00000000b - eSDHC Version 1.0 00010000b - eSDHC Version 2.0 00010001b - eSDHC Version 2.1 00010010b - eSDHC Version 2.2 00010011b - eSDHC Version 2.3

Table continues on the next page...

Field	Function
	00100000b - eSDHC Version 3.0 00100001b - eSDHC Version 3.1 00100010b - eSDHC Version 3.2
24-31 SVN	Specification version number. These status bits indicate the host controller specification version. Patterns not shown are reserved. 00000000b - SD Host Specification Version 1.0 00000001b - SD Host Specification Version 2.0, supports test event register , and ADMA 00000010b - SD Host Specification Version 3.0

18.4.25 DMA error address register (DMAERRADDR)

18.4.25.1 Offset

Register	Offset
DMAERRADDR	104h

18.4.25.2 Function

The DMAERRADDR contains the address of the transaction on which DMA error occured.

18.4.25.3 Diagram

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

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

18.4.25.4 Fields

Field	Function
0-31 DMA_ADDRn	DMA error address. This field contains the system address of the transaction on which DMA error occured.

18.4.26 DMA error attribute register (DMAERRATTR)

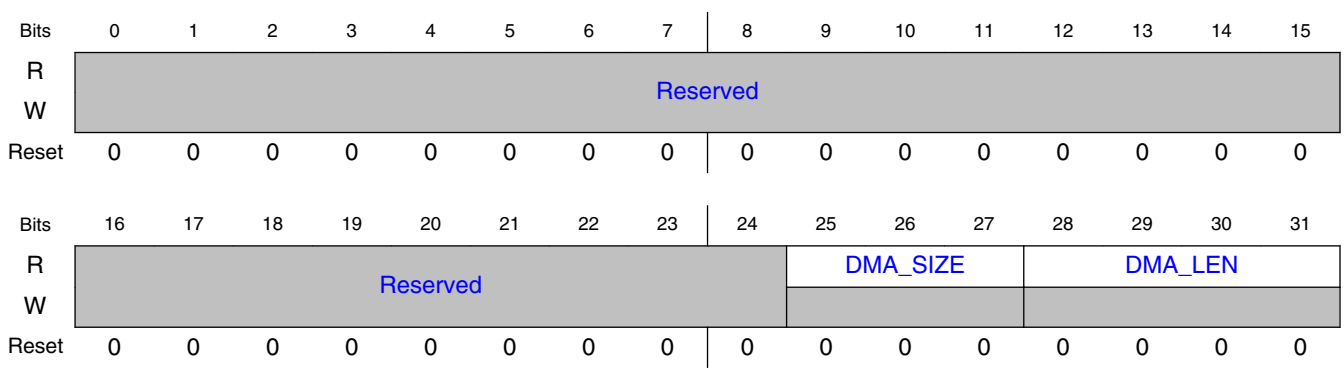
18.4.26.1 Offset

Register	Offset
DMAERRATTR	10Ch

18.4.26.2 Function

The DMAERRATTR contains attributes of the transaction on which DMA error occured.

18.4.26.3 Diagram



18.4.26.4 Fields

Field	Function
0-24	Reserved.

Table continues on the next page...

Field	Function
—	
25-27 DMA_SIZE	System bus burst size. This field contains burst size of the transaction on which DMA error occured.
28-31 DMA_LEN	System bus burst length. This field contains burst length of the transaction on which DMA error occured.

18.4.27 Host controller capabilities register 2 (HOSTCAPBLT2)

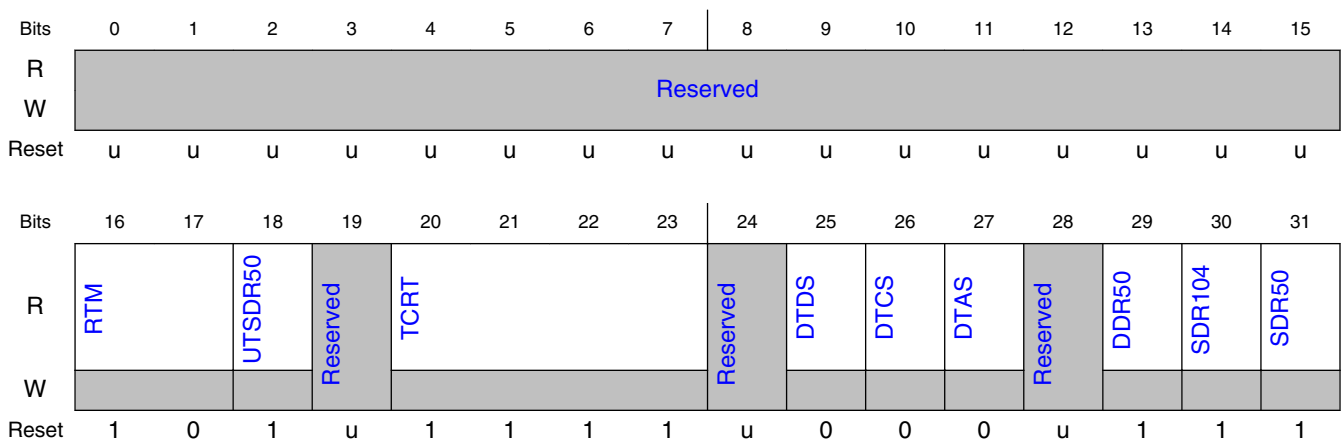
18.4.27.1 Offset

Register	Offset
HOSTCAPBLT2	114h

18.4.27.2 Function

This register provides the host driver with information specific to the eSDHC implementation. The value in this register is the power-on-reset value, and does not change with a software reset. Any write to this register is ignored.

18.4.27.3 Diagram



18.4.27.4 Fields

Field	Function
0-15 —	Reserved.
16-17 RTM	Re-tuning modes. This bit indicates the supported re-tuning modes for UHS. 00b - Mode 1 - Software Timer 01b - Mode 2 - Software Timer and Re-tuning request 10b - Mode 3 - Software Timer, and Auto Re-tuning during data transfer 11b - Reserved
18 UTSDR50	Use tuning for SDR50. This bit indicates whether the host support tuning for SDR50 mode. 0b - Tuning for SDR50 mode not supported 1b - Tuning for SDR50 mode supported
19 —	Reserved.
20-23 TCRT	Timer Count for Re-Tuning : This field indicates an initial value of Re-Tuning timer for retuning mode 1 to 3. 0000b - Re-Tuning timer disabled 0001b - 1 second 0010b - 2 seconds 0011b - 4 seconds ... 1011b - 1024 seconds 1100-1110b - Reserved 1111b - Get timer information from other source.
24 —	Reserved.
25 DTDS	Driver type D support. This bit indicates whether the system is capable of using driver type D. 0b - Driver Type D not supported 1b - Driver Type D Supported
26 DTCS	Driver type C support. This bit indicates whether the system is capable of using driver type C. 0b - Driver Type C not supported 1b - Driver Type C Supported
27 DTAS	Driver type A support. This bit indicates whether the system is capable of using driver type A. 0b - Driver Type A not supported 1b - Driver Type A Supported
28 —	Reserved.
29 DDR50	DDR50 support. This bit indicates whether the eSDHC supports the DDR mode. 0b - DDR mode not supported 1b - DDR mode supported
30 SDR104	SDR104 Support. This bit indicates whether the eSDHC supports the SDR104. 0b - SDR104 not supported 1b - SDR104 supported
31 SDR50	SDR50 support. This bit indicates whether the eSDHC supports the SDR50. 0b - SDR50 not supported 1b - SDR50 supported

18.4.28 Tuning block control register (TBCTL)

18.4.28.1 Offset

Register	Offset
TBCTL	120h

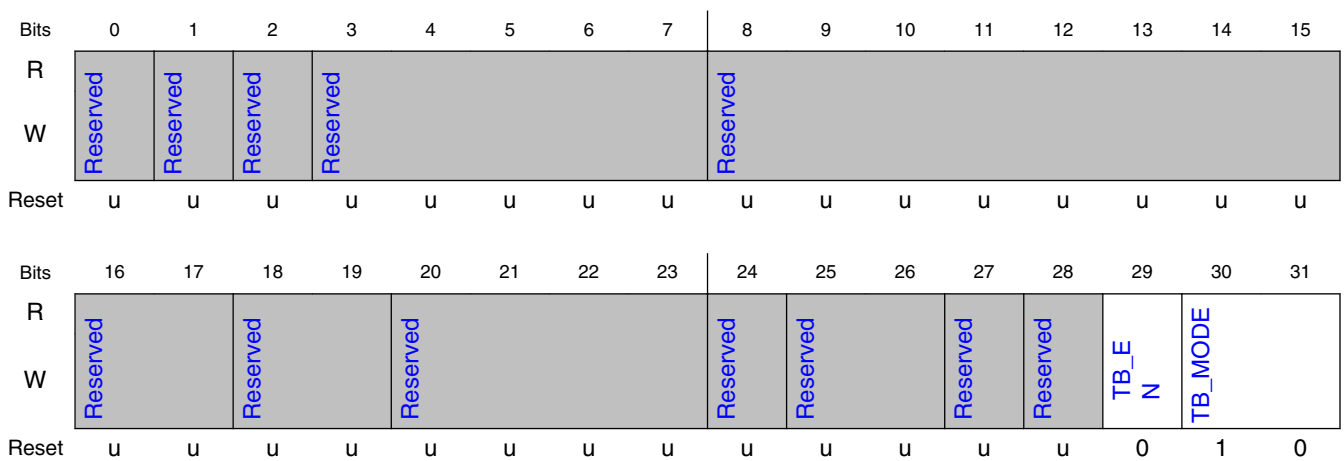
18.4.28.2 Function

This register contains fields for controlling the tuning block.

NOTE

Writing or reading to reserved fields of this register does not guarantee specific values will be written or read.

18.4.28.3 Diagram



18.4.28.4 Fields

Field	Function
0	Reserved.

Table continues on the next page...

Field	Function
—	
1 —	Reserved.
2 —	Reserved.
3-7 —	Reserved.
8-15 —	Reserved.
16-17 —	Reserved.
18-19 —	Reserved.
20-23 —	Reserved.
24 —	Reserved.
25-26 —	Reserved.
27 —	Reserved.
28 —	Reserved.
29 TB_EN	Tuning block enabled. Tuning block should be enabled for high speed SDR mode more than 50 MHz SD clock frequency. This bit is not reset by software reset for all. 0b - Tuning block is disabled 1b - Tuning block is enabled
30-31 TB_MODE	Tuning Mode Tuning Mode. Selects tuning mode when tuning block is enabled. Refer to re-tuning modes in Table 18-13 00b - Mode 1 - Software Timer 01b - Mode 2 - Software Timer, and Re-tuning request 10b - Mode 3 - Software Timer, and Auto Re-tuning during data transfer 11b - SW tuning mode - Software tuning mode where start and end point of data window needs to be programmed in TBPTR register and use software timer for re-tuning

18.4.29 Tuning block status register (TBSTAT)

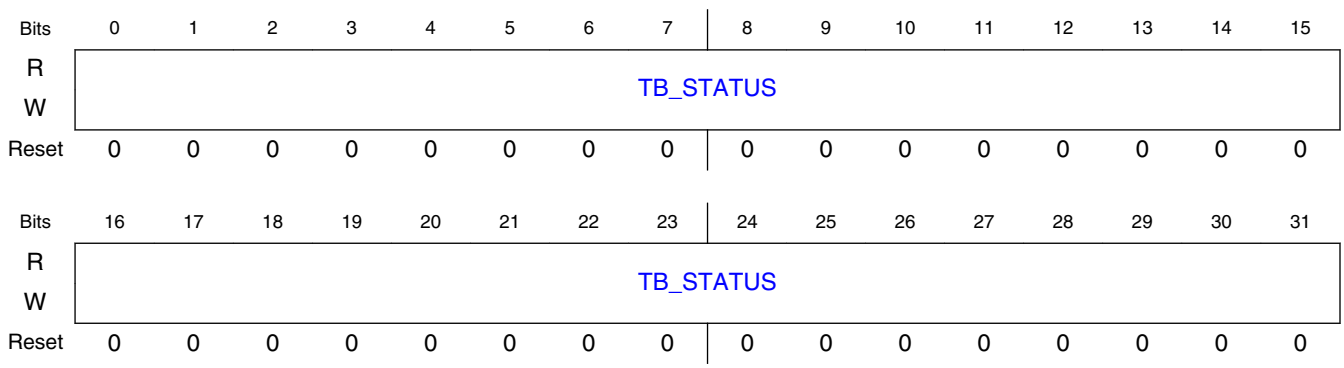
18.4.29.1 Offset

Register	Offset
TBSTAT	124h

18.4.29.2 Function

This register contains the status of the tuning block.

18.4.29.3 Diagram



18.4.29.4 Fields

Field	Function
0-31 TB_STATUS	Tuning Status.

18.4.30 Tuning block pointer register (TBPTR)

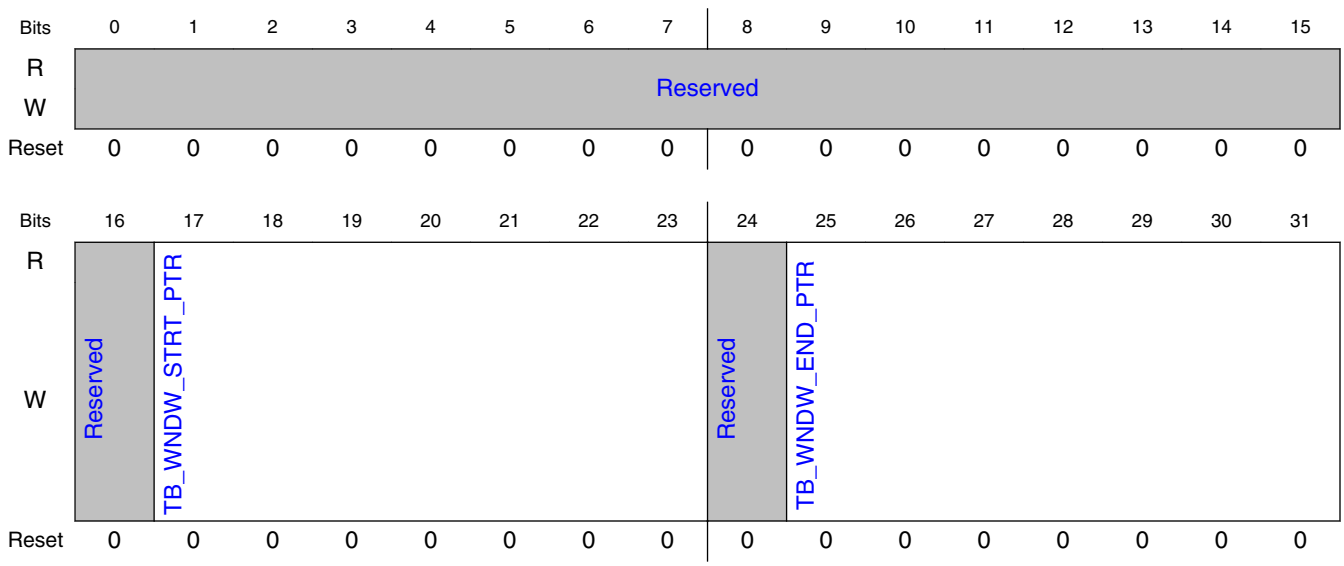
18.4.30.1 Offset

Register	Offset
TBPTR	128h

18.4.30.2 Function

This register contains fields for controlling tuning block pointers.

18.4.30.3 Diagram



18.4.30.4 Fields

Field	Function
0-16 —	Reserved.
17-23 TB_WNDW_STRT_PTR	Tuning window start pointer. Selects window start pointer for software tuning mode (when TBCTL[TB_MODE]=3).
24 —	Reserved.
25-31 TB_WNDW_END_PTR	Tuning window start pointer. Selects window end pointer for software tuning mode (when TBCTL[TB_MODE]=3)

18.4.31 SD direction control register (SDDIRCTL)

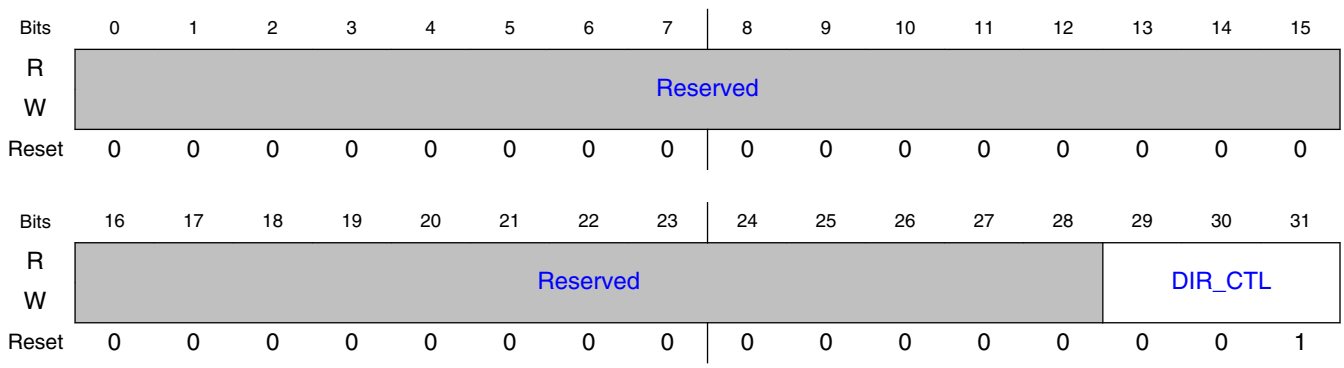
18.4.31.1 Offset

Register	Offset
SDDIRCTL	140h

18.4.31.2 Function

This register contains control for CMD and DAT lines direction.

18.4.31.3 Diagram



18.4.31.4 Fields

Field	Function
0-28 —	Reserved.
29-31 DIR_CTL	Direction control Specify the turnaround time required for external transceiver after the assertion of direction pins in number of SD clocks 000b - No turnaround time required 001b - 1 SD clock period for turnaround 010b - 2 SD clock periods for turnaround ... 111b - 7 SD clock periods for turnaround

18.4.32 SD Clock Control Register (SDCLKCTL)

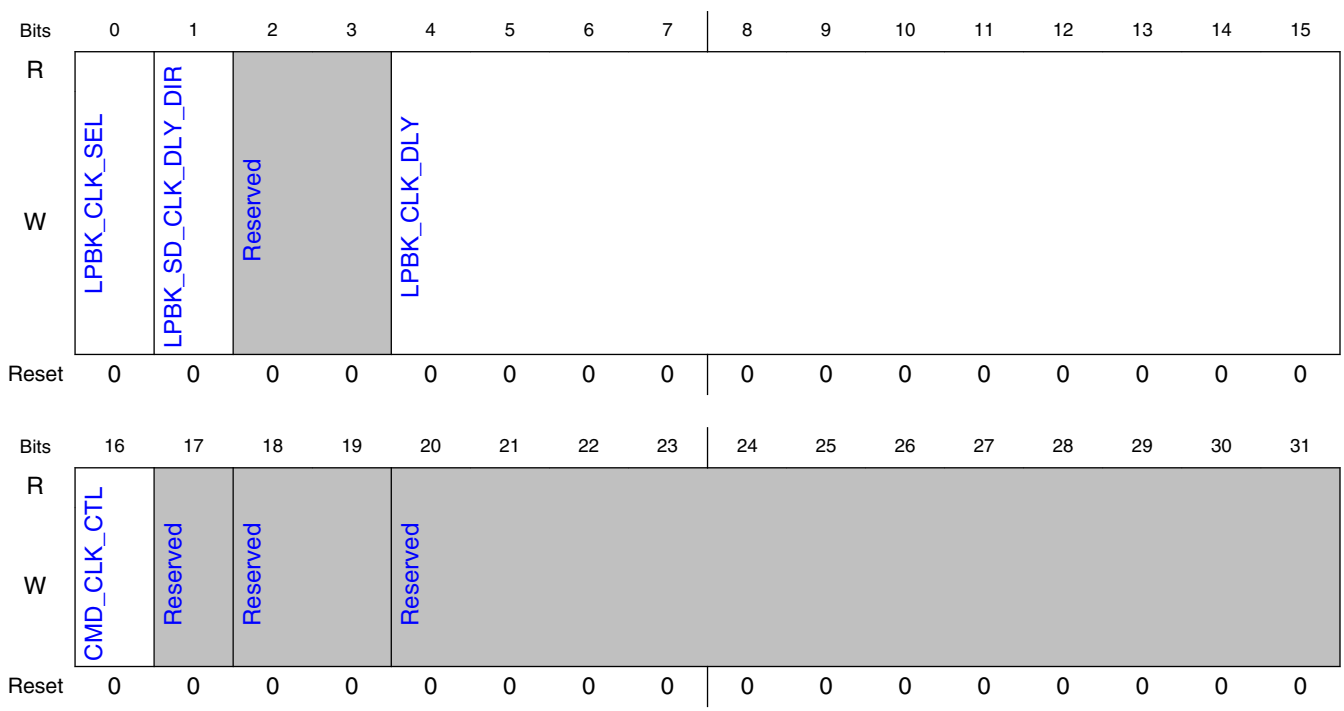
18.4.32.1 Offset

Register	Offset
SDCLKCTL	144h

18.4.32.2 Function

This register contains fileds for controlling SD external and loopback clock.

18.4.32.3 Diagram



18.4.32.4 Fields

Field	Function
0 LPBK_CLK_SEL	SD Loopback Clock Select. This field specifies whether SD clock is loopbacked from external pin or from internal pad. 0b - SD card clock is loopbacked from internal pad 1b - SD card clock is loopbacked from external pin
1 LPBK_SD_CLK_DLY_DIR	SD Loopback Clock Delay Direction. This field specifies whether SD loopback card clock is delayed in positive or negative direction. 0b - SD card loopback clock is delayed by LPBK_CLK_DLY value 1b - SD card loopback clock is early by LPBK_CLK_DLY value
2-3 —	Reserved.
4-15 LPBK_CLK_DLY	SD Loopback Clock Delay. This field specifies the number of periphal clocks (based on ESDHCCTL[PCS]) by which SD loopback card clock is delayed. 000000000000b - No delay in SD card clock 000000000001b - 1/2 peripheral clocks delay introduced in SD card clock 000000000010b - 2/2 peripheral clocks delay introduced in SD card clock ... 111111111111b - 4095/2 peripheral clocks delay introduced in SD card clock
16 CMD_CLK_CTL	Command Logic Clock Control. This field specifies controls clock to the command logic. 0b - Command logic clock is same as data logic clock 1b - Command logic clock is 25% shifted early from data logic clock
17 —	Reserved.
18-19 —	Reserved.
20-31 —	Reserved.

18.4.33 eSDHC control register (ESDHCCTL)

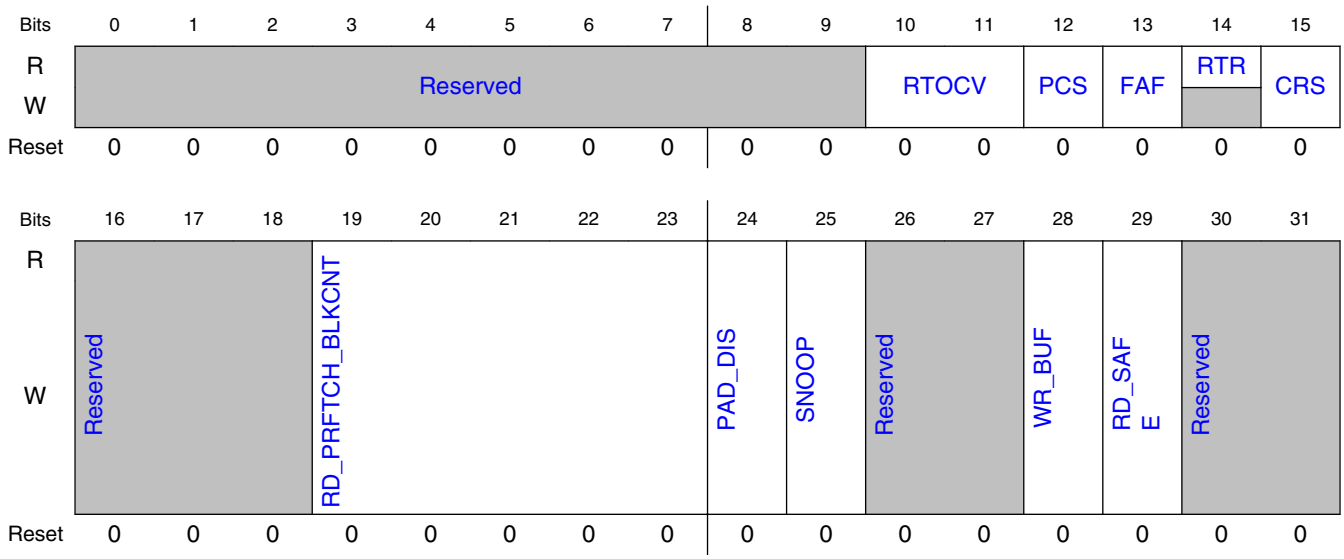
18.4.33.1 Offset

Register	Offset
ESDHCCTL	40Ch

18.4.33.2 Function

This register contains fields for controlling DMA transfers.

18.4.33.3 Diagram



18.4.33.4 Fields

Field	Function
0-9 —	Reserved.
10-11 RTOCV	Register timeout count value. This field define the timeout value for register access. 00b - Timeout count value for register access is 2^10 clocks 01b - Timeout count value for register access is 2^11 clocks 10b - Timeout count value for register access is 2^12 clocks 11b - Timeout count value for register access is 2^13 clocks
12 PCS	Peripheral clock select. This bit selects the clock used for generating SD clock. This bit is not reset by software reset for all. 0b - Platform clock is used 1b - Peripheral/2 clock is used
13 FAF	Flush asynchronous FIFO. This bit is used to flush asynchronous FIFO. It can be set by software and will be auto cleared by hardware. 0b - No Flush 1b - Flush async FIFO
14 RTR	Re-tuning request. This bit indicates re-tuning request status. eSDHC 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

Table continues on the next page...

Field	Function
	is issued with setting execute tuning in the system control 2 register. 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 tuning block enable in the tuning control register is set to 0 (using fixed sampling clock). This is read-only field. 0b - No re-tuning request 1b - Re-tuning request is set
15 CRS	Clock register select. This bit selects the clock division format of SDCLKFS and DVS fields of system control register. 0b - SDCLKFS is defined as 8-bit field, and DVS is active in system control register 1b - SDCLKFS is defined as 10-bit field, CGS is active in system control register
16-18 —	Reserved.
19-23 RD_PRFTCH_BLKCNT	Read prefetch block count. For DMA read (SD write), this field specifies the maximum number of SD blocks that the host can prefetch from the system memory. It can have following values: 00000b - No prefetch 00001b - 1 SD block prefetch 00010b - 2 SD block prefetch 11111b - 31 SD block prefetch
24 PAD_DIS	Pad disable Pad disable. Padding disable control for DMA transaction with non-word (4-bytes) aligned block size. For more details refer to Data buffer and block size. 0b - DMA will pad data at the end each block transfer. 1b - DMA will not pad data at the end of each block transfer.
25 SNOOP	Snoop attribute. Snoop enable for DMA transaction. 0b - DMA transactions are not snooped by the CPU data cache. 1b - DMA transactions are snooped by the CPU data cache.
26-27 —	Reserved.
28 WR_BUF	Write bufferable. DMA always initiate write transaction on System bus corresponding to last in every SD block as non-bufferable. This bit specifies whether all non-last write transactions will be bufferable or not. 0b - Non-last write transactions are not bufferable. 1b - Non-last write transactions are bufferable.
29 RD_SAFE	Read safe. This bit should be set only if the target of read dma operation is a well behaved memory which is not affected by the read operation and will return the same data if read again from the same location. This means that unaligned reading operation can be rounded up to enable more efficient read operations. 0b - It is not safe to read more bytes that were intended. 1b - It is safe to read more bytes that were intended.
30-31 —	Reserved.

18.5 Functional description

The eSDHC block is partitioned in five major sub-blocks as shown in Figure 18-2.

- SD interface and control unit

This block interfaces with the SD bus. It is mainly responsible for controlling the SD bus operation and transferring data from Data Buffer and SD bus. It also interacts with System Interface and Control Unit.

- System interface and control unit

  This block interfaces with the system interfaces (that is, the system bus in DMA mode and register bus in CPU polling mode) and transfers the card data from the data buffer and system.

- Register bank

  This block interfaces with register bus and contains all the registers. It controls the overall operation of eSDHC and also provides status through various registers.

- Clock and reset

  This block generates divided clock for SD interface and provides appropriate reset to all the blocks.

- SD monitor

  This block monitors the SD bus and provides status to register banks, such as card interrupt, write protect and card detect.

18.5.1 System interface and control unit (SysICU)

The SysICU block is further partitioned into three major sub-blocks:

- System control block

  This sub-block receives data transfer request from the register bank, controls data transfer for whole transaction and generates control signals for DMA and buffer control operation on per block basis.

- Buffer control block

  This sub-block handles buffer port register access in CPU polling mode and contains the buffer FIFO to store transfer data. It controls the data transfer per-block basis in the CPU polling mode.

- DMA block

  This sub-block generates DMA transactions on system bus to transfer the data between system memory and data buffer.

The following sections provide a brief functional description of the major system blocks, including the data buffer, DMA system interface, register bank as well as IP bus interface, dual-port memory wrapper, data/command controller, clock and reset manager, and clock generator.

18.5.1.1 Data buffer

The eSDHC uses one configurable data buffer, so that data can be transferred between the system/register bus and SD card in an optimized manner to maximize throughput between the two clock domains. See the figure below for illustration of the buffer scheme.

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 any number from 1-128 words for CPU polling mode. The burst lengths for read and write are also configurable, and can be any number from 1-16 for DMA mode.

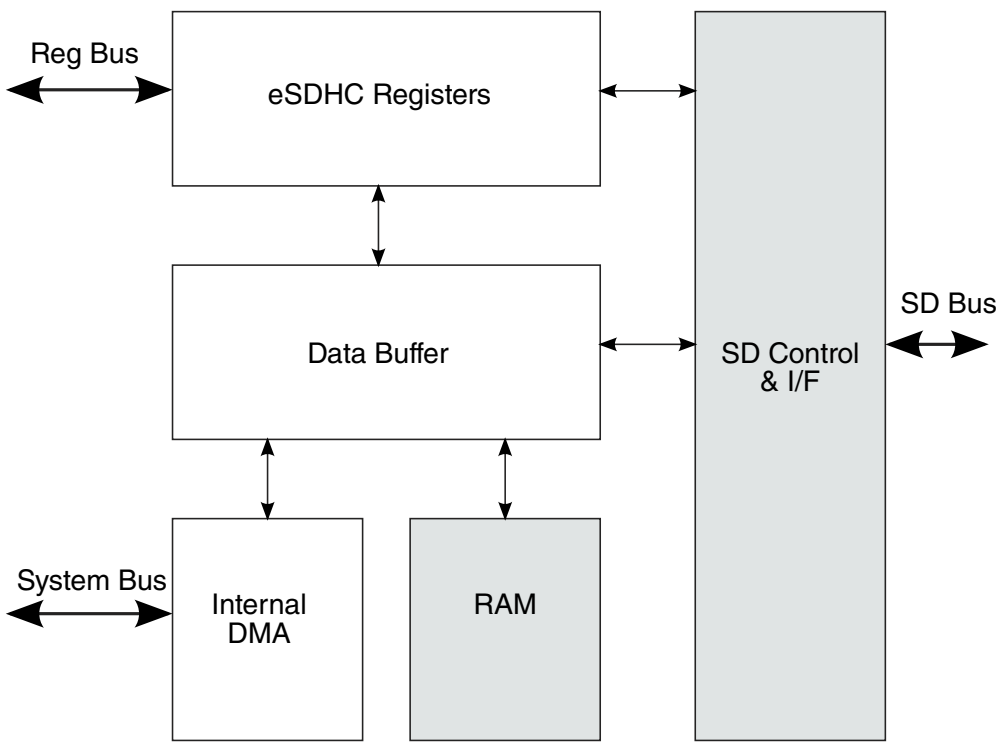


Figure 18-3. eSDHC buffer scheme

There are two transfer modes to access the data buffer:

- CPU polling mode

Data is transferred over register bus. For a host read operation, when the number of words received in the buffer meets or exceeds the RD_WML watermark value, then by monitoring (polling or interrupt) the BRR bit the host driver can read the buffer data port register (DATPORT) to fetch the amount of words set in the RD_WML register from the buffer. For block size integral multiple of watermark level value set in watermark level register (WML), driver must access buffer data port register (DATPORT) exactly the same number of times as the watermark level. However, if the block size is not the times of the watermark level value, the software must access exactly the remained number of words at the end of each block. For example, for read operation, if the RD_WML is 4, indicating the watermark level is 16 bytes, block size is 40 bytes, and the block count is 2, then the access times to Data Buffer Port Register for the burst sequence in the whole transfer process must be 4, 4, 2(for first block); 4, 4, 2(for second block). The write operation is similar.

- DMA mode (includes simple and advanced DMA accesses)

  Data is transferred over system bus. DMA interrupt is generated after all the data is transferred to/from the buffer.

18.5.1.1.1 Write operation sequence

There are two ways to write data into the buffer when the user transfers data to the card:

- By processor core polling through IRQSTAT[BWR] (interrupt or polling)
- By using the DMA

When the DMA is not used, (CPU polling mode, that is, the XFERTYP[DMAEN] is not set when the command is sent), and when the amount of buffer space exceeds the value set in the WR_WML register, PRSSTAT[BWR] is set. The buffer write ready interrupt is generated if it is enabled by software.

When DMA is used, the eSDHC 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 eSDHC aborts the data transfer and abandons the current block. If the current data transfer is in multi block mode, the eSDHC does not automatically send CMD12, even though the XFERTYP[AC12EN] is set. The host driver needs to send CMD12 in this scenario. It is recommended that a Software Reset for Data be applied before the transfer is re-started after error recovery.

18.5.1.1.2 Read operation sequence

There are two ways to read data from the buffer when the user transfers data to the card:

- By processor core polling through IRQSTAT[BRR] (interrupt or polling)
- By using the DMA

When DMA is not used (CPU polling mode, that is, the XFERTYP[DMAEN] is not set when the command is sent), and when the amount of data exceeds the value set in the RD_WML register, that is available and ready for system fetching data, PRSSTAT[BRR] is set. The buffer read ready interrupt will be generated if it is enabled by software.

When DMA is used, the eSDHC 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 eSDHC will abort the data transfer and abandon the current block. If the current data transfer is in multi block mode, the eSDHC will not automatically send CMD12, even though the XFERTYP[AC12EN] is set. The host driver should send CMD12 in this scenario. It is recommended that a software reset for data be applied before the transfer is re-started after error recovery.

18.5.1.1.3 Data buffer and block size

In eSDHC, the data buffer can hold up to 128 words (32-bit).

The watermark levels for both write and read can be configured for CPU polling mode. The watermark level can be from one word to a maximum of 128 words. For both DMA read and write, the burst length can be configured from one to a maximum of 16. The host driver may configure watermark level and burst length value according to the system situation and requirement in the watermark level register (WML).

During a multi-block data transfer, the block length may be set to any value between 1 and 2048 bytes inclusive that satisfies the requirements of the external card. The only restriction to it can be from the external card, which may not support that large a block or partial access to block (which is not an integer times of 512 bytes).

For CPU polling mode, when block size not a multiple of four; that is, not word aligned, eSDHC requires stuff bytes at the end of each block, as defined in the host specification. For example, if the block size is 5 bytes and there are two blocks to write, there must be two register bus writes to buffer data port register (DATPORT) for each block; and for each block, the ending non-word aligned bytes should be stuffed in buffer data port register (DATPORT) write to make it word aligned. For this example, 3 bytes should be stuffed. eSDHC will transfer only the required number of bytes to the card and ignore the stuff bytes as shown in the figure below. Read operation is similar.

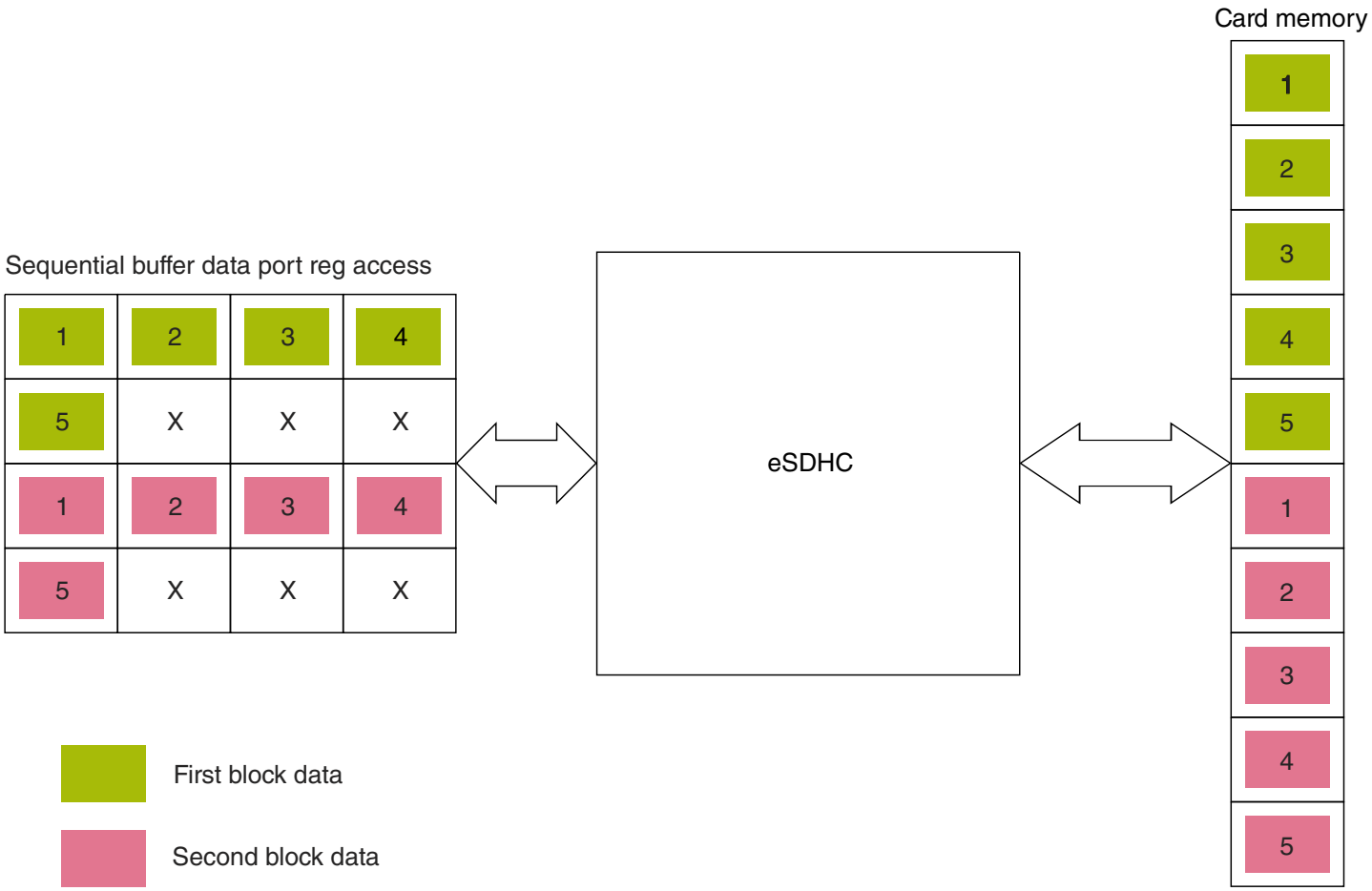


Figure 18-4. Byte stuffing for CPU polling mode

For DMA mode, when block size not a multiple of four; that is, not word aligned, eSDHC requires stuff bytes depending on the PAD_DIS field programmed in DMA Control register. The data transfer with byte stuffing enabled (when DMACTL[PAD_DIS]=0) is shown in Figure 18-5. And, data transfer with byte stuffing disabled (when DMACTL[PAD_DIS]=1) is shown in Figure 18-6. Transfer with byte stuffing disabled eliminates the software overhead of byte stuffing. Driver may program DMA Control register accordingly.

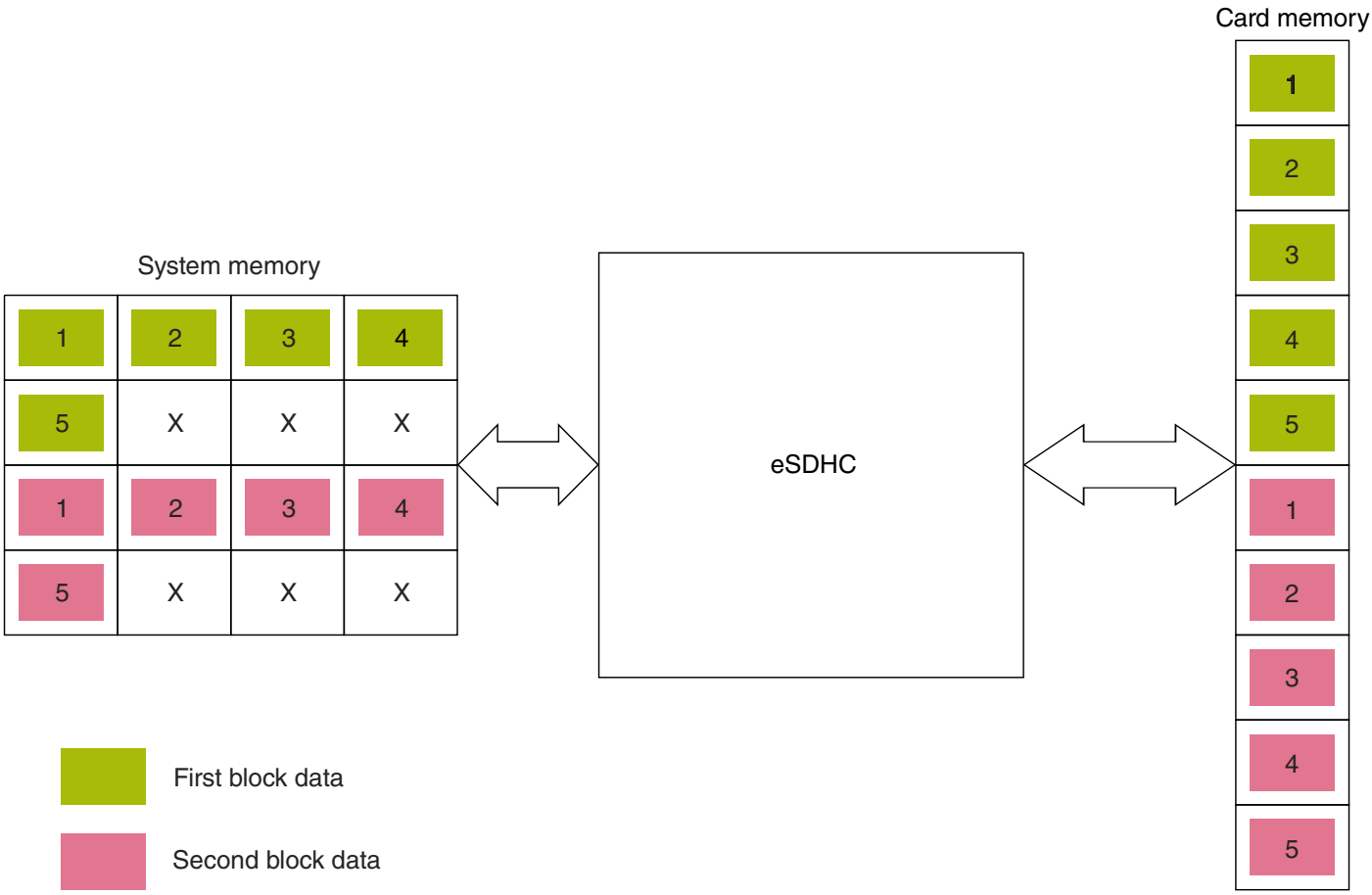


Figure 18-5. Byte stuffing for DMA mode when DMACTL[PAD_DIS]=0

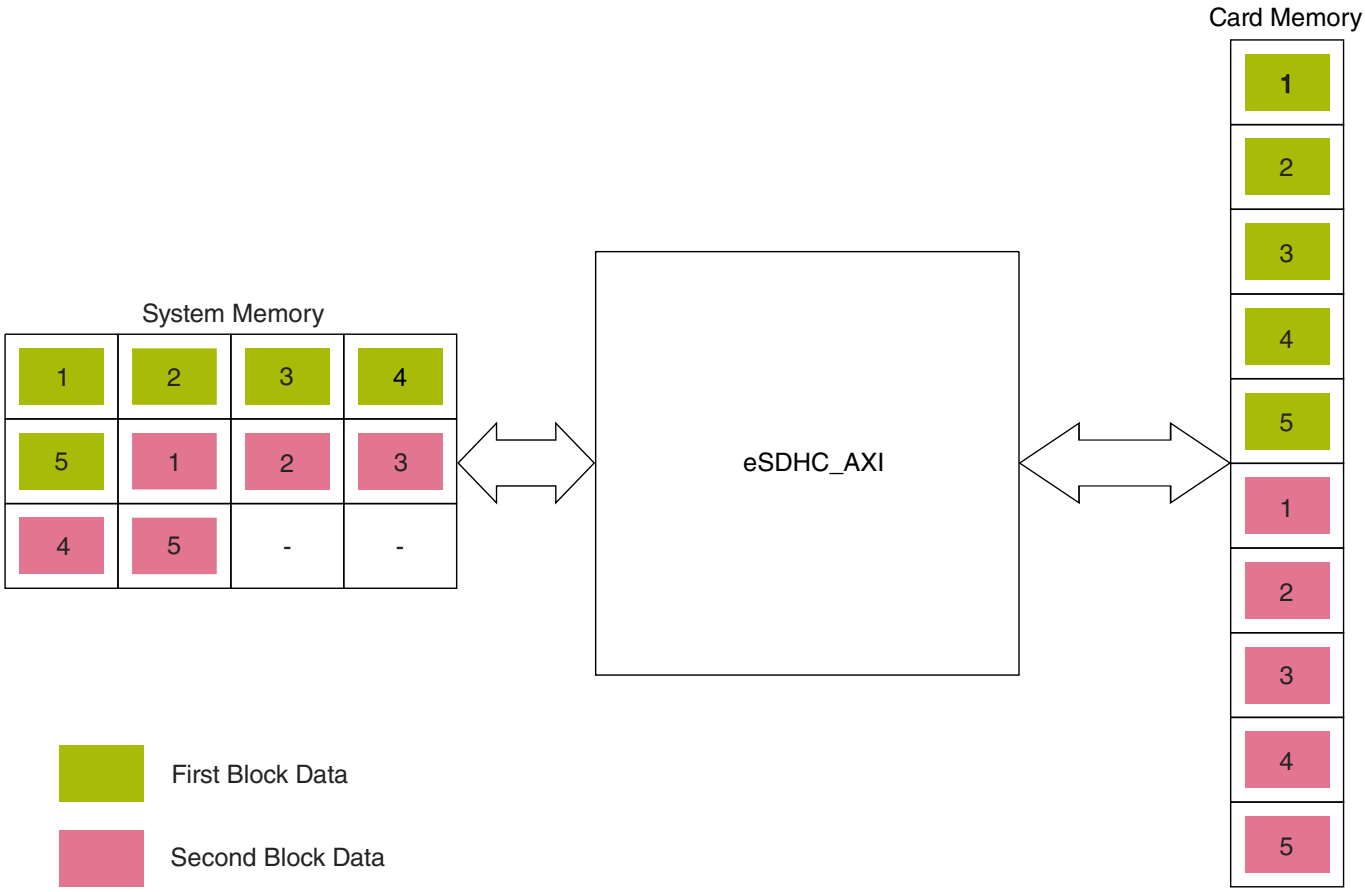


Figure 18-6. No byte stuffing for DMA mode when DMACTL[PAD_DIS]=1

18.5.1.1.4 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 cannot be divided evenly into a multiple of the block size, then based on the function and the card design, there are two ways to transfer the data. First option 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. Second option is to add dummy data in the last block to fill the block size provided the card manages the removal of the dummy data.

Figure below shows an example which explains the dividing of large data transfers. In this figure, assume a kind of WLAN SDIO card that only supports block size up to 64 bytes. Although the eSDHC supports a block size of up to 2048 bytes, the SDIO can only accept a block size less than 64 bytes. Thus, the data must be divided (see example below).

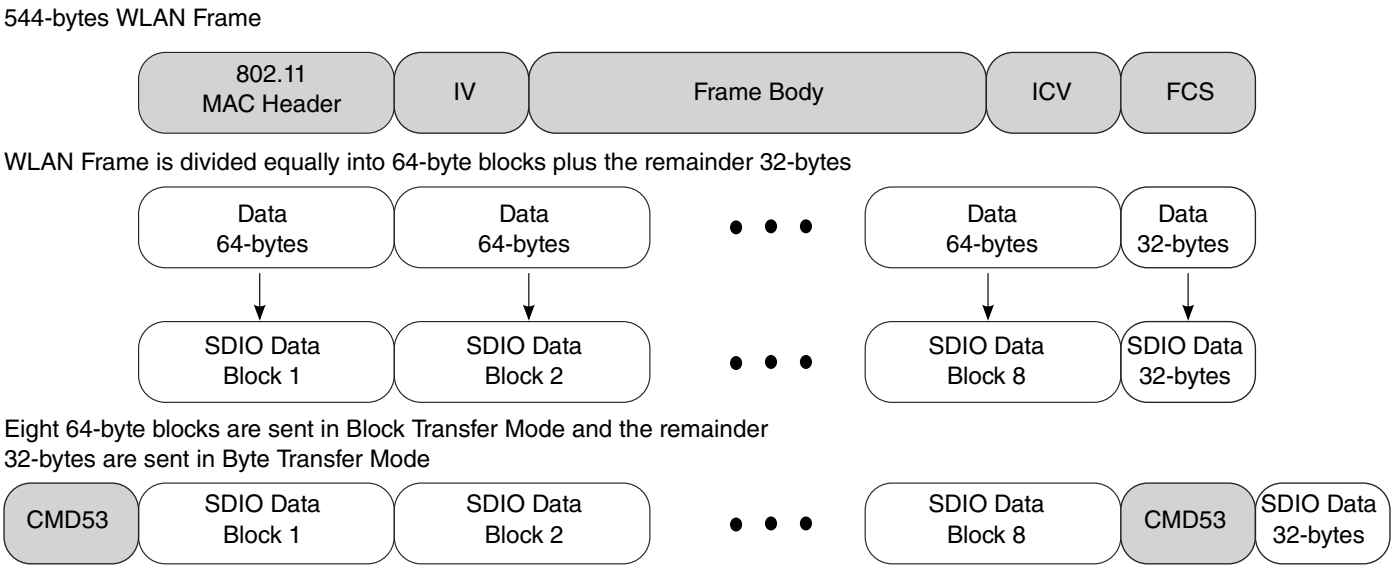


Figure 18-7. Example for dividing large data transfers

18.5.1.1.5 Byte order (endianness) of buffer data port register

For CPU polling mode, sequential and contiguous access is necessary to ensure the pointer address value is correct. Random or skipped access is not possible.

The byte order of buffer data port register (DATPORT), by reset is little endian mode. The data buffer is always little endian. The data is swapped inside the buffer, according to the endian mode configured by software in PROCTL[EMODE].

In little endian mode, the data is transferred between data buffer and buffer data port register (DATPORT) without any swapping. In big endian mode, the data between data buffer and buffer data port register (DATPORT) is byte-swapped as shown in the figure given below. For an SD write operation, byte order is swapped after data is fetched from the buffer port register and sent to data buffer. For a host read operation, byte order is swapped after the data is read from data buffer and sent to buffer port register.

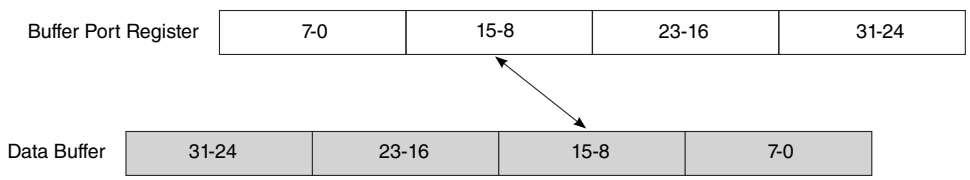


Figure 18-8. Data swap between buffer data port register and data buffer in big endian mode

18.5.1.2 DMA system interface

The DMA implements a DMA engine and the system master.

The eSDHC supports both SDMA and ADMA (ADMA1 and ADMA2). Figure below illustrates the DMA system interface block.

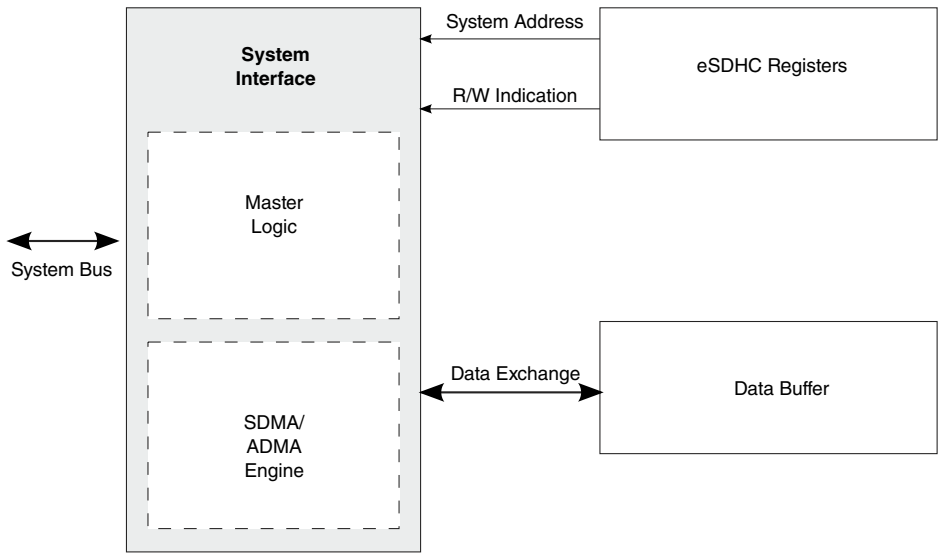


Figure 18-9. DMA system interface block

18.5.1.2.1 DMA burst length

The actual burst length on system bus depends on the shortest of following factors:

- Burst length configured in the burst length field of the watermark level register (WML)
- Block size boundary
- Data boundary configured in the current descriptor (if the ADMA is active)

18.5.1.2.2 System master interface

The System DMA engine can at times fail during data transfer.

When this error occurs:

- The DMA engine stops the transfer and goes to the error state.
- The internal data buffer stops accepting incoming data.
- The IRQSTAT[DMAE] is set to inform the driver.

Once the DMAE interrupt is received, the software sends a CMD12 to abort the current transfer and read the DMAERRADDR[DMA_ADDR] bits to get the starting address of the corrupted block. For error recovery, the software issues a data reset and re-starts the transfer from this address to recover the corrupted block.

18.5.1.3 Single DMA (SDMA)

SDMA is single operation DMA, that is, data is transferred for single operation only (unlike ADMA which has chained descriptor table) from the starting address programmed in SDMA system address register (DSADDR) for entire data transfer size (block count x block size) as programmed in the block attributes register (BLKATTR) if XFERTYP[BCEN] is set.

18.5.1.3.1 SDMA error

SDMA will stop whenever an error is encountered on the system bus.

DMA error address register latches the transaction address on which the error occured.

18.5.1.4 Advanced DMA (ADMA)

Advanced DMA (ADMA) is a new DMA transfer algorithm, which is defined in the SD Host Controller Standard.

For single DMA, the data can be transferred for single operation only. 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. Higher speed DMA transfers are realized because the host MCU intervention is not needed during long DMA based data transfers.

The host controller has two types of ADMA: ADMA1 and ADMA2. 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 defined in SD Host Controller Standard and if 'End' flag is detected in the descriptor, ADMA stops after this descriptor is processed.

eSDHC supports ADMA1 and ADMA2 with 32-bit addressing.

18.5.1.4.1 ADMA concept and descriptor format

ADMA1 includes the following descriptors:

- Valid/invalid descriptor
- Nop descriptor
- Set data length and address descriptor
- Link descriptor
- Interrupt flag and End flag in descriptor

ADMA2 includes the following descriptors:

- Valid/Invalid descriptor
- Nop descriptor
- Rsv descriptor
- Set data length and address descriptor
- Link descriptor
- Interrupt flag and End flag in descriptor

Figure 18-10 explains the ADMA1 descriptor table format.

Figure 18-12 explains the ADMA2 descriptor table format. 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 should be set on word aligned (lower 2-bit is always set to 0). Data length is in byte unit.

ADMA starts 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 triggers 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 is the value for previous transfer, or 4 Kbyte if no set descriptor is ever met.

For ADMA2, Tran type descriptor contains both data length and transfer data address. Thus, 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
Addressor Data Length		000000		Act2	Act1	0	Int	End	Valid

Act2	Act1	Symbol	Comment	31-28	27-12
0	0	NOP	No Operation	Do not 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 descriptor.
Int	Int=1 generates DMA Interrupt when this descriptor is processed.

Figure 18-10. Format of the 32-bit address ADMA1 descriptor table

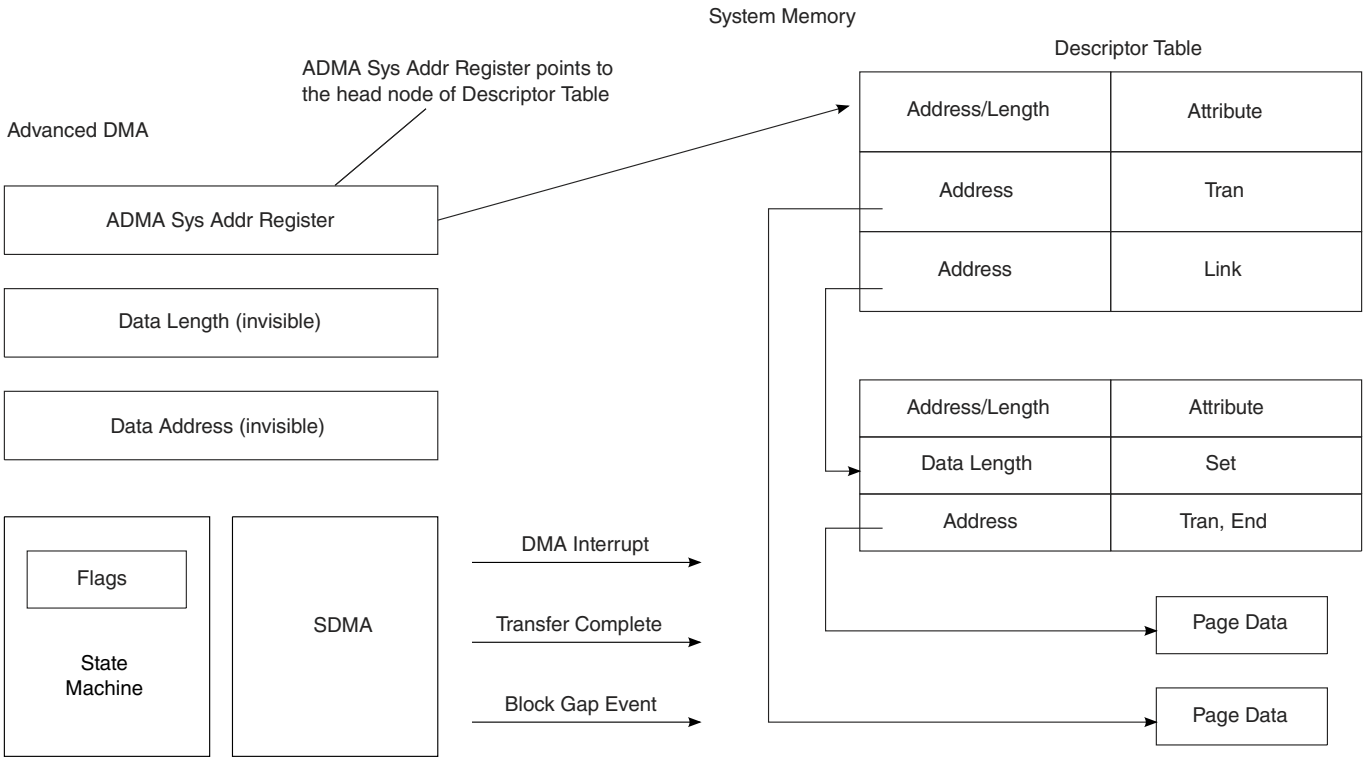


Figure 18-11. Concept and access method of ADMA1 descriptor table

Address Field		Length		Reserved		Attribute Field					
63	32	31	16	15	6	5	4	3	2	1	0
32-bit Address		16-bit Length		0000000000		Act2	Act1	0	Int	End	Valid

Act2	Act1	Symbol	Comment	Operation
0	0	NOP	No Operation	Do not care
0	1	Rsv	Reserved	Same as NOP. Read this line and go to next one
1	0	Tran	Transfer Data	Transfer data with address and length set in this descriptor line
1	1	Link	Link Descriptor	Link to another descriptor

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 descriptor.
Int	Int=1 generates DMA Interrupt when this descriptor is processed.

Figure 18-12. Format of the 32-bit address ADMA2 descriptor table

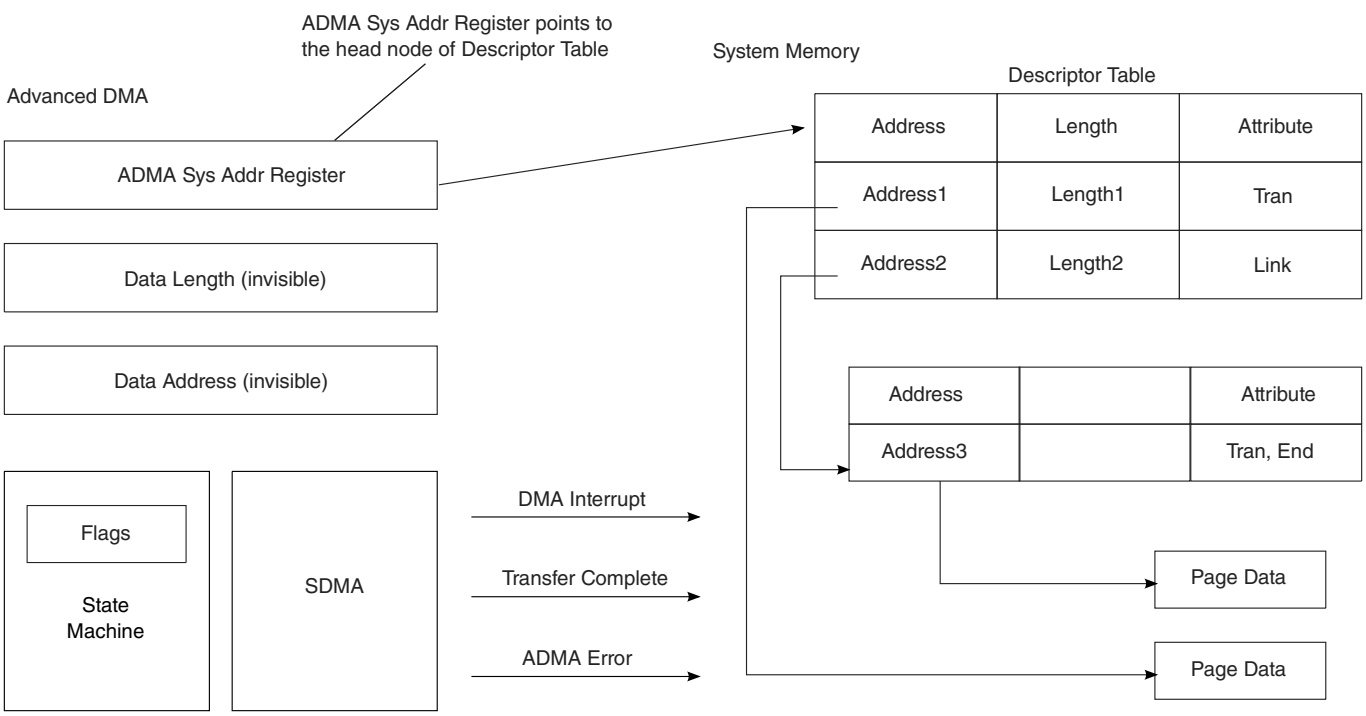


Figure 18-13. Concept and access method of ADMA2 descriptor table

18.5.1.4.2 ADMA interrupt

If the 'interrupt' flag of descriptor is set, ADMA will generate an interrupt, IRQSTAT[DINT] whenever the data transfer for the particular descriptor line is completed.

18.5.1.4.3 ADMA error

The ADMA stops whenever any of the following error is encountered:

- Fetching descriptor error
- System response error
- Data length mismatch error

ADMA descriptor error is generated when it fails to detect 'Valid' flag in the descriptor. If ADMA descriptor error occurs, the interrupt is not generated even if the 'Interrupt' flag of this descriptor is set.

When the BLKCNTEN bit is set, the data transfer length set in buffer must be equal to the whole data transfer length set in the descriptor nodes, otherwise data length mismatch error is generated.

If the BLKCNTEN bit is not set, the whole data transfer length set in descriptor should be times of block length, otherwise, when all data set in the descriptor nodes are done, the data length mismatch error will occur.

The DMA error address register (DMAERRADDR) latches the transaction address on which the error occured. The ADMA system address register (ADSADDR) latches the current descriptor line address which encountered the error.

18.5.2 SD interface and control unit (SDICU)

This block is partitioned into six major sub-blocks as shown in Figure 18-2.

- Transfer control block: This sub-block receives command request from the register bank and overall controls other SDICU sub-blocks for SD bus operation on transaction level.
- SD command block: This sub-block controls the SD CMD line for sending command out. It receives the command request from Transfer Control block and sends it on SD CMD line.
- SD response block: This sub-block monitors the SD CMD line for receiving response and detecting line errors. It sends the command and response status to transfer control for transaction operation.

- SD data out block: This sub-block controls the SD DAT lines for sending out the block write data from the SD data out FIFO.
- SD data in block: This sub-block monitors the SD DAT line for detecting read and write data block end and busy period end for transfer control module. It also detects data line error and card interrupt.
- Tuning block: This sub-block receives Tuning block data from card and tunes IP. It then forwards the sampled data to Async FIFO for further operation.

18.5.2.1 Command CRC

See the figure below for an illustration of the structure for the command CRC shift register.

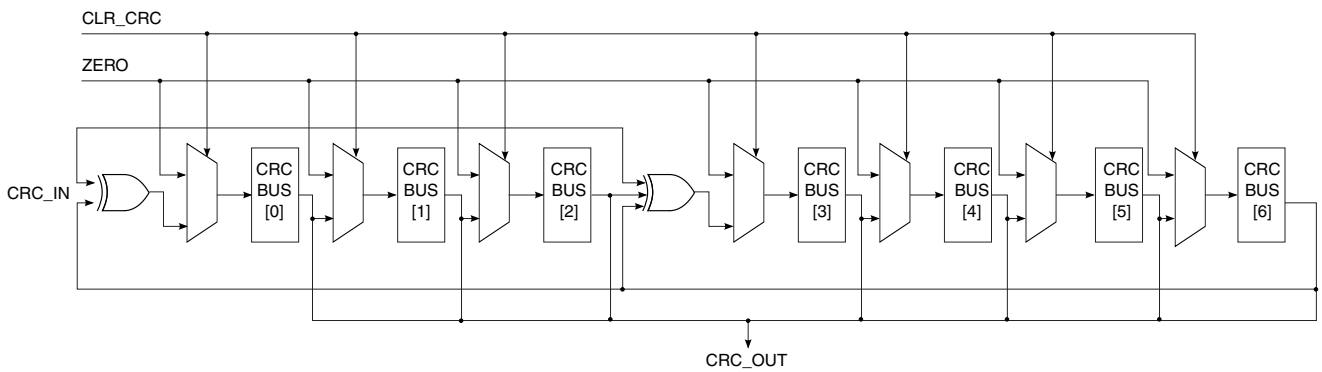


Figure 18-14. Command CRC shift register

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)]
```

18.5.2.2 Data CRC

The CRC polynomials for the DAT 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)]
```

18.5.2.3 Tuning block (SDICU)

Tuning block is used for SD UHS SDR50, SDR104 and eMMC HS200 modes.

Tuning block tunes the IP on tuning command (CMD19 for SD, CMD21 for eMMC) data sent by card and the sampled data is sent to async FIFO for further operation.

This block oversamples data from card and selects particular sampling point after completion of tuning procedure, refer to Tuning block procedure for tuning block usage. Various re-tuning modes are supported by eSDHC as described below.

Table 18-13. Re-tuning modes

Re-Tuning Mode	Re-Tuning Method	Data Length
1	Software timer	4MB (Max.)
2	Software timer, and re-tuning request	4MB (Max.)
3	Software timer, and auto re-tuning during data transfer	Any

There are two re-tuning timings: Re-tuning request controlled by eSDHC and expiration of a re-tuning timer(software timer) controlled by the host driver. By receiving either timing, the host driver executes the re-tuning procedure just before a next command issue.

The maximum data length per read/write command is restricted so that re-tuning procedures can be inserted during data transfers.

Re-tuning mode 1

eSDHC do not generate re-tuning request. In this case, the host driver should maintain all re-tuning timings by using a re-tuning timer. To enable inserting the re-tuning procedure during data transfers, the data length per read/write command shall be limited up to 4 MB.

Re-tuning mode 2

eSDHC indicates the re-tuning timing by re-tuning request during data transfers. Then the data length per read/write command shall be limited up to 4 MB. During non data transfer, re-tuning timing is determined by re-tuning timer.

Re-tuning mode 3

eSDHC takes care of the re-tuning during data transfer (auto re-tuning). Re-tuning request will not be generated and there is no limitation to data length per read/write command. During non data transfer, re-tuning timing is determined by re-tuning timer.

Re-tuning timer control example for re-tuning mode 1

The software timer starts counting by loading the initial value. When the timer expires, the host driver marks an expiration flag. On receiving a command request, the host driver checks the expiration flag. If the expiration flag is set, then the host driver should perform

the re-tuning procedure before issuing a command. If the expiration flag is not set, then the host driver issues a command without performing the re-tuning procedure. Every time the re-tuning procedure is performed, the timer loads the new initial value and the expiration flag is cleared.

Re-tuning timer control example for re-tuning mode 2 and mode 3

The software timer control is almost the same as re-tuning mode 1 except the timer loads the new initial value after data transfer (when receiving transfer complete). in case of mode 3, timer count for re-tuning is set either smaller value: tuning effective time after re-tuning procedure or after data transfer.

If a host system goes into power down mode, the host driver should stop the re-tuning timer and set the expiration flag to 1 when the host system resumes from power down mode.

18.5.3 Register bank

This block interfaces with register bus and it contains all the registers. It controls the overall operation of eSDHC and also provides status through various registers.

Register accesses is actually on the register bank. See the figure below for the block diagram.

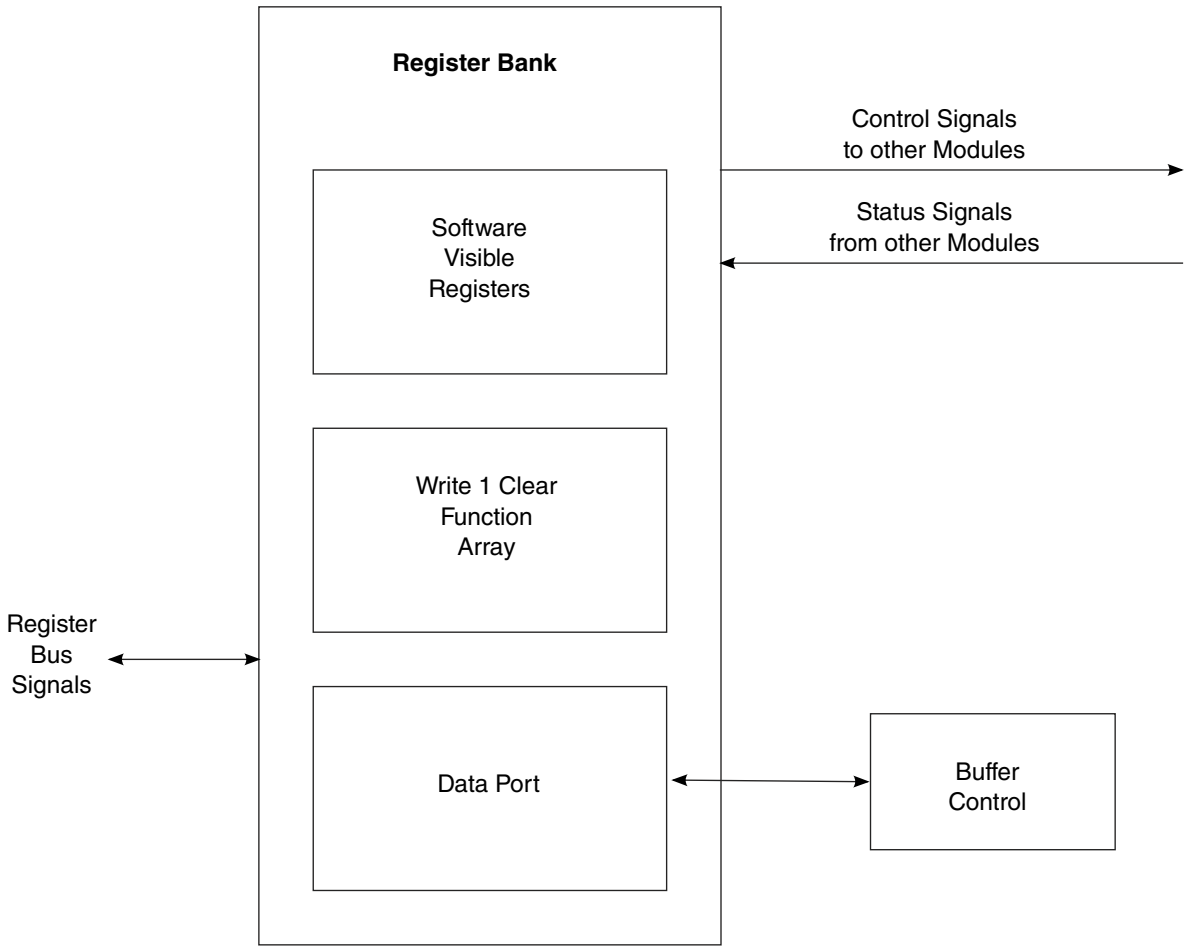


Figure 18-15. Register bank diagram

Partial access is not allowed on any register; that is, it should be 32-bit access only.

18.5.4 Clock and reset module

Clock and reset module generates divided clock for SD interface and provides appropriate reset to other modules.

There are four kinds of reset signals within eSDHC:

- Hardware reset
- Software reset for all
- Software reset for the data part
- Software reset for the command part

All these signals are fed into this module and stable signals are generated inside the module to reset all other modules.

If the internal data buffer is in danger, and the SD clock must be gated off to avoid buffer over/under-run, this module asserts the gate of the output SD clock to shut the clock off. After the buffer danger has recovered, and when the system access of the buffer catches up, the clock gate of this module opens and the SD clock becomes active again.

18.5.4.1 Clock generator

The clock generator generates the SDHC_CLK by source clock in two stages.

The figure below illustrates the structure of the divider. The term "Base" represents the frequency of the source clock.

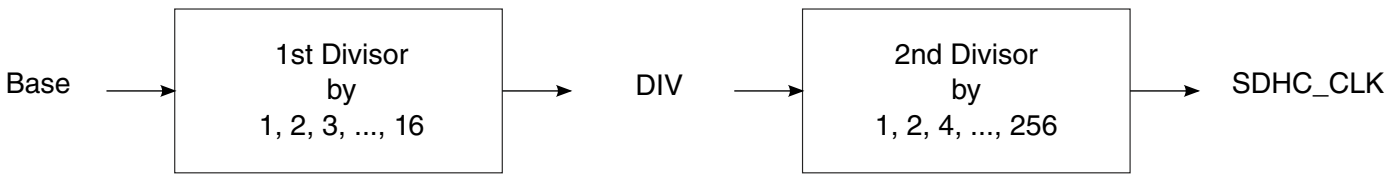


Figure 18-16. Two stages of the clock divider

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 clock (SDHC_CLK). This clock is the driving clock for sub modules SD Command and SD Data Out in SD Interface and Control Unit (refer to Figure 18-2) 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 SDHC_CLK is Base, and the next highest is Base/2, while the lowest frequency is Base/4096.

The figure below illustrates the sequence for changing the SD clock frequency.

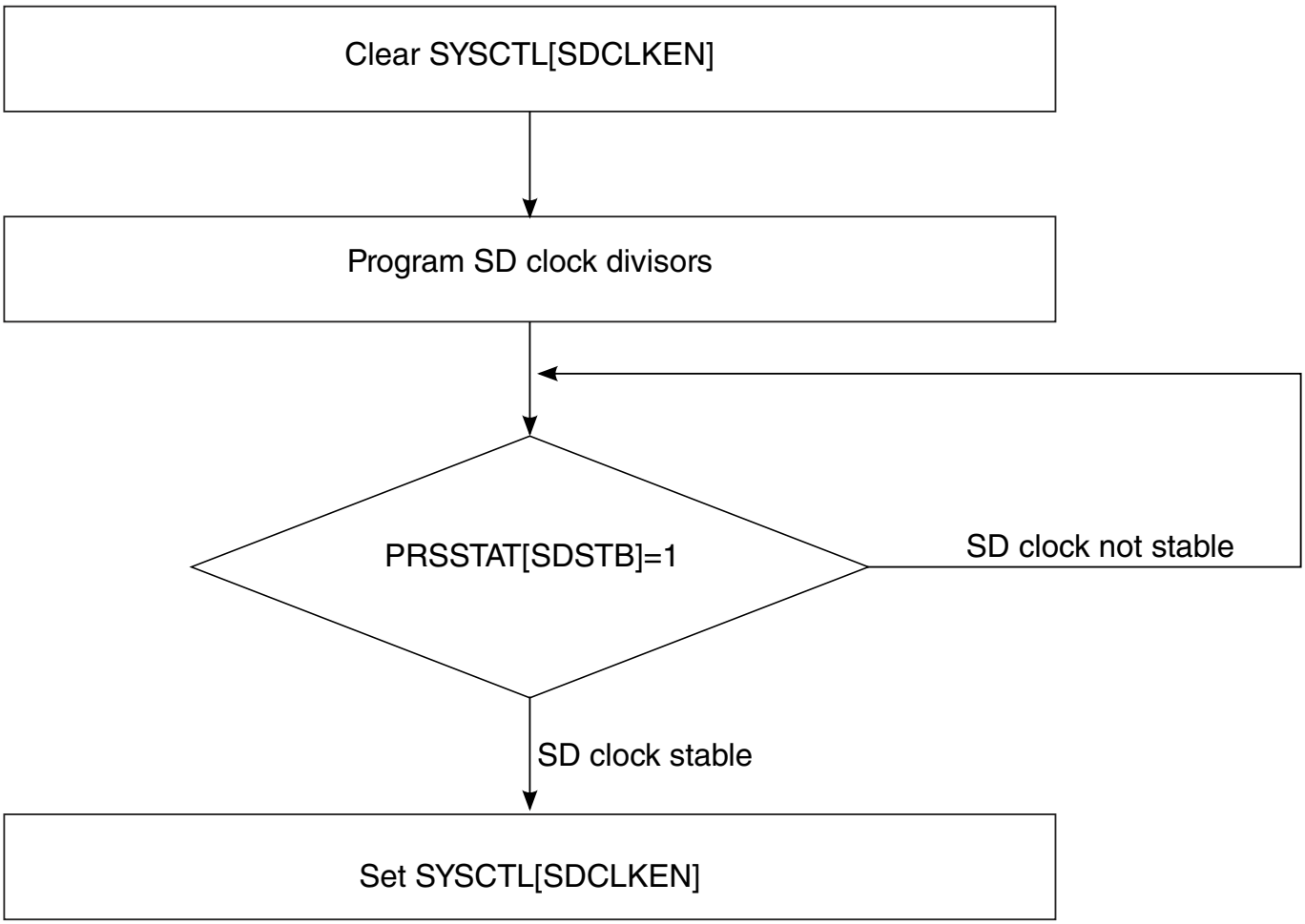


Figure 18-17. SD clock frequency change sequence

Base clock can be selected by programming ESDHCCTL[PCS]. It selects between platform clock and peripheral clock / 2. Base clock is divided by two of peripheral clocks when PCS=1.

1. Clear SYSCTL[SDCLKEN]
2. Wait for PRSSTAT[SDSTB] to be set
3. Program appropriate value of ESDHCCTL[PCS]
4. Set SYSCTL[SDCLKEN]
5. Wait for PRSSTAT[SDSTB] to be set

18.5.5 SD monitor

The module detects the CD_B (card detection) as well as the DAT3 signal. The transceiver reports the card insertion state according to the CD_B state, the signal level on the DAT3 signal.

NOTE

Do not use DAT3 pin as a CD pin.

The module detects the WP (write protect) signal. With the information of the WP state, the register bank ignores the command, accompanied by a write operation, when the WP switch is on.

18.5.5.1 SDIO card interrupt

18.5.5.1.1 Interrupts in 1-bit mode

In this case the DAT[1] pin is dedicated to providing the interrupt function.

An interrupt is asserted by pulling the DAT[1] low from the SDIO card, until the interrupt service is finished to clear the interrupt.

18.5.5.1.2 Interrupt in 4-bit mode

Since the interrupt and DAT[1] share pin 8 in 4-bit mode, an interrupt is sent by the card and recognized by the host only during a specific time.

This is known as the interrupt period. eSDHC only samples the level on pin 8 during the interrupt period. At all other times, the host ignores the level on pin 8, and treats 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 DAT[1] line is held low for one clock cycle with the last clock cycle pulling DAT[1] high. On completion of the interrupt period, the card releases the DAT[1] line into the high Z state. eSDHC samples DAT[1] during the interrupt period when the PROCTL[IABG] is set.

Refer to SDIO card specification v2.0 for further information about the SDIO card interrupt.

18.5.5.1.3 Card interrupt handling

When the CINTIEN bit in the interrupt signal enable register is set to 0, the eSDHC 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.

In 1-bit mode, eSDHC detects 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 is set, and the host driver needs to service this interrupt, so the SDIO bit in the interrupt control register of SDIO card is cleared. This is required to clear the SDIO interrupt status latched in the eSDHC 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 enable bit is set to 1, and the eSDHC starts sampling the interrupt signal again.

The figure below illustrates the SDIO card interrupt scheme and sequences of software and hardware events that take place during a card interrupt handling procedure.

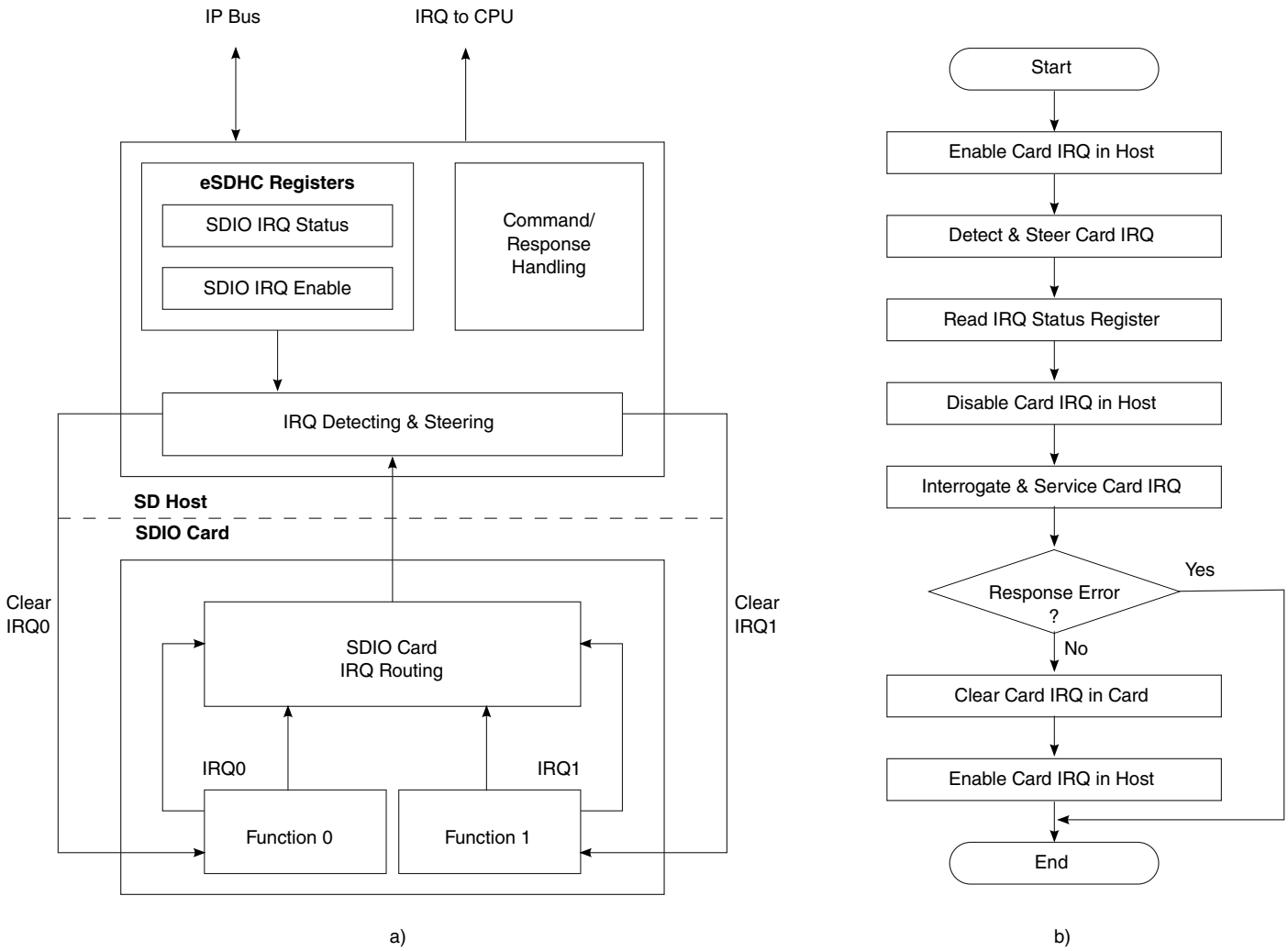


Figure 18-18. Card interrupt scheme and card interrupt detection and handling procedure

18.5.5.2 Card insertion and removal detection

When the DAT[3] pin is not used for card detection (for example, it is implemented in GPIO), the CD pin must be connected for card detection. Whether DAT[3] is configured for card detection or not, the CD pin is always a reference for card detection. Whether the DAT[3] pin or the CD pin is used to detect card insertion, the eSDHC sends an interrupt (if enabled) to inform the host system that a card is inserted.

18.5.5.3 Power management and wake up events

If no operation is expected to happen between eSDHC and the card through the SD bus, the user can completely disable the IP in the chip-level clock control module to save power.

When the user needs to use the eSDHC to communicate with the card, it can enable the clock and start the operation.

In some circumstances, when the clocks to the eSDHC 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 eSDHC can generate these interrupt even when there are no clocks enabled. The three interrupts which can be used as wake up events are:

- Card removal interrupt
- Card insertion interrupt
- Interrupt from SDIO card

The eSDHC offers a power management feature. By clearing the clock enabled bits in the system control register (SYSCTL), the clocks are gated in the low position to the eSDHC. For maximum power saving, the user can disable all the clocks to the eSDHC 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 eSDHC 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 register (PROCTL) for more information.

18.5.5.3.1 Setting wake up events

For eSDHC 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 are met:

- No read or write transfer is active
- Data and command lines are not active
- No interrupts are pending
- Internal data buffer is empty

18.6 Initialization/application of eSDHC

All communication between the system and the 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" and so on. In Broadcast mode, all cards are in the open-drain mode to avoid bus contention. Refer to Commands for /SD/SDIO and eMMC device 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/DAT I/O pads will turn to push-pull mode, to have the driving capability for maximum frequency operation. Refer to Commands for /SD/SDIO and eMMC device, for the commands of ac and adtc categories.

18.6.1 Command send and 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
(XFERTYP),
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 (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 (XFERTYP) 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 using polling. For an effective and formal way, the response is usually checked after the Command Complete Interrupt is received. By 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 should deal with 'fake' errors like this with caution.

18.6.2 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) .For eMMC devices, set the relative card address. All data communications in the card identification mode use the command line (CMD) only.

18.6.2.1 Card detect

The figure below illustrates a flow diagram showing the detection of , SDIO, and SD cards using the eSDHC.

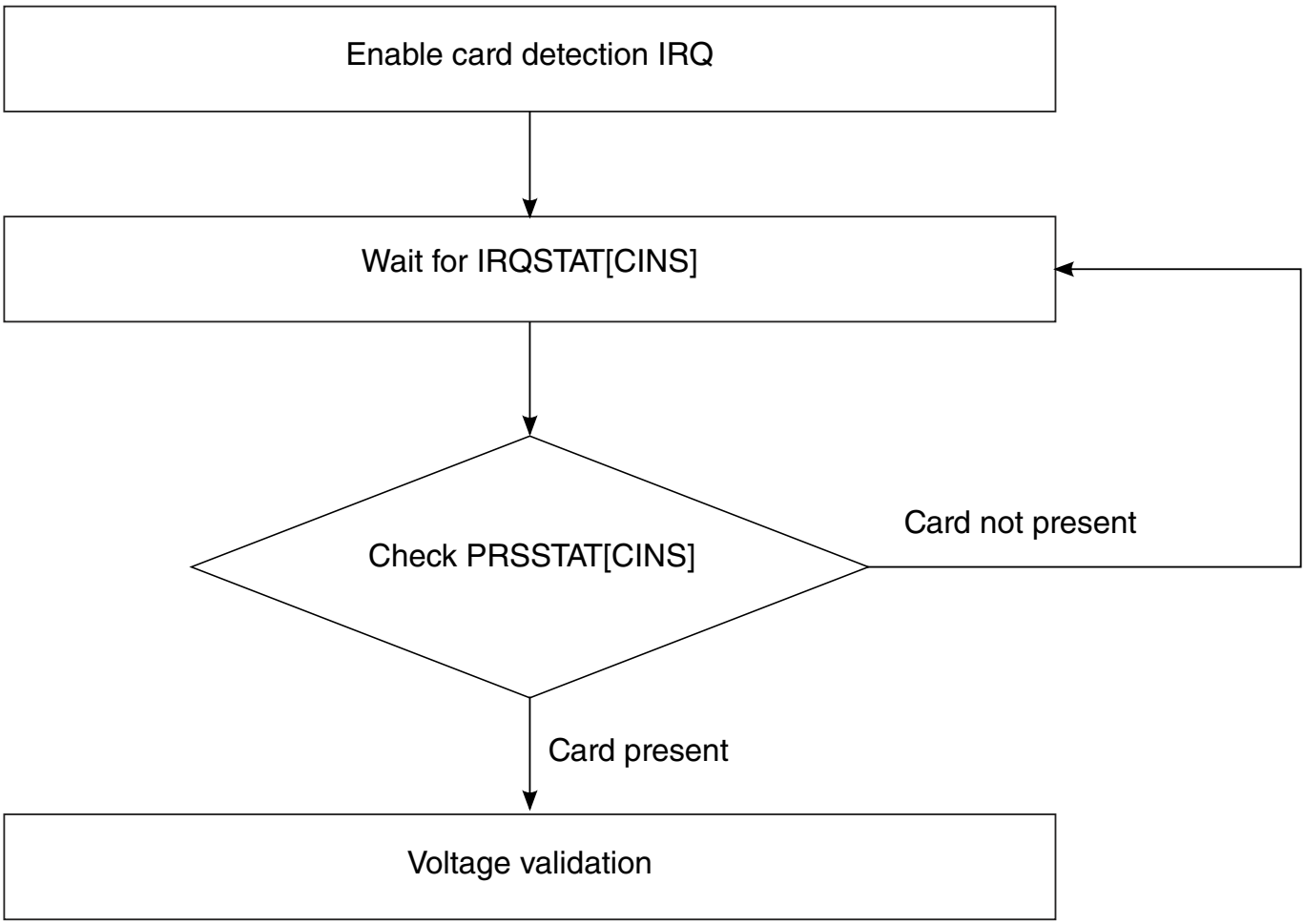


Figure 18-19. Flow diagram for card detection

The card detect sequence is as follows:

1. Set the IRQSTAT[CINSIEN] bit and IRQSIGEN[CINSIEN] bit to enable card detection interrupt.
2. When an interrupt from the eSDHC is received in IRQSTAT[CINS], check PRSSTAT[CINS] to confirm if it was caused by card insertion.

NOTE

This section is not valid for embedded devices.

18.6.2.2 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 proceed by the write operation on SYSCTL[RSTD], SYSCTL[RSTC], or SYSCTL[RSTA] bits 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 eMMC devices and 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. Figure below illustrates the software flow to reset both the eSDHC and the card.

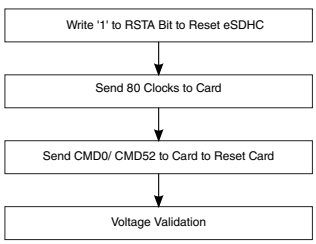


Figure 18-20. Flow chart for reset of the eSDHC and SD I/O card

```
software_reset()
{
set_bit(SYSCTRL, RSTA); // software reset the Host
set DTOCV and SDCLKFS bit fields to get the SDHC_CLK of frequency around 400 KHz
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>);
}
```

18.6.2.3 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 eMMC), 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 cannot 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 should 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 eMMC device
          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 ...
          else label the card as eMMC;
} // of else
}
```

18.6.2.4 Card registry

Card registry for the SD/SDIO/SD combo cards and eMMC devices 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 specification). 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 should 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 eMMC 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.

```
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 eMMC) {
                    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 eMMC ...
} while (response is not time-out);
}
```

18.6.3 Card access

This section describes access to the card.

18.6.3.1 Block write

18.6.3.1.1 Normal write

During a block write (CMD24 - 27), 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 should indicate the failure on the DAT 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 eMMC devices and SD cards, programming of the CID and CSD registers do 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 DAT 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 DAT line without interrupting the write operation. When re-selecting the card, it will reactivate the busy indication by pulling DAT to low if the programming is still in progress and the write buffer is unavailable.

The software flow to write to a card using DMA mode is as follows:

1. Check the card status, wait until the card is ready for data.
2. Set the card block length/size:
   a. For SD cards and eMMC devices, use SET_BLOCKLEN (CMD16)
   b. 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 eSDHC block length register to be the same as the block length set for the card in Step 2.
4. Set the eSDHC number block register (NOB), nob is 5 (for instance).
5. Disable the buffer write ready interrupt, configure the DMA settings and enable the eSDHC 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 other error that occurred during the auto12 command sending and response receiving.

The software flow to write to a card using CPU Polling mode is as follows:

1. Check the card status, wait until the card is ready for data.
2. Set the card block length/size:
   a. For SD cards and eMMC devices, use SET_BLOCKLEN (CMD16)
   b. 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 eSDHC block length register to be the same as the block length set for the card in Step 2.
4. Set the eSDHC number block register (NOB), nob is 5 (for instance).
5. Issue the command with data transfer. The AC12EN bit should also be set.
6. Wait for IRQSTAT[BWR] interrupt to be set.
7. Write to Buffer port register for no. of times programmed in WML[WR_WML] considering restriction mentioned in Software polling procedure.
8. Clear IRQSTAT[BWR].
9. Repeat 6-8 steps for rest of the data transfer.
10. Wait for the Transfer Complete interrupt.
11. Check the status bit to see if a write CRC error occurred, or some other error that occurred during the auto12 command sending and response receiving.

18.6.3.1.2 Write with pause

The write operation can be paused during the transfer.

Instead of stopping the SDHC_CLK at any time to pause all the operations, which is also inaccessible to the host driver, the driver can set the PROCTL[SABGREQ] 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 DAT0 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:
   a. For SD cards and eMMC devices, use SET_BLOCKLEN (CMD16)
   b. 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 eSDHC block length register to be the same as the block length set for the card in Step 2.

4. Set the eSDHC number block register (NOB), nob is 5 (for instance).
5. Disable the buffer write ready interrupt, configure the DMA settings and enable the eSDHC 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 BLKATTR[BLKCNT]. 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 should 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 eSDHC thinks that the system will switch to another function on the SDIO card, and flush the data buffer. The eSDHC 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, XFERTYP[MSBSEL] and XFERTYP[BCEN] are set as well as XFERTYP[AC12EN]. However, the eSDHC will automatically send a CMD12 to mark the end of the multi-block transfer.

18.6.3.2 Block read

18.6.3.2.1 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 using DMA mode is as follows:

1. Check the card status, wait until card is ready for data.
2. Set the card block length/size:
   a. For SD cards and eMMC devices, use SET_BLOCKLEN (CMD16)
   b. 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 eSDHC block length register to be the same as the block length set for the card in Step 2.
4. Set the eSDHC number block register (NOB), nob is 5 (for instance).
5. Disable the buffer read ready interrupt, configure the DMA settings and enable the eSDHC 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 other error, occurred during the auto12 command sending and response receiving.

The software flow to read from a card using CPU polling mode is as follows:

1. Check the card status, wait until card is ready for data.
2. Set the card block length/size:
   a. For SD cards and eMMC devices, use SET_BLOCKLEN (CMD16)
   b. 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 eSDHC block length register to be the same as the block length set for the card in Step 2.
4. Set the eSDHC number block register (NOB), nob is 5 (for instance).
5. Issue the command with data transfer. The AC12EN bit should also be set.
6. Wait for IRQSTAT[BRR] interrupt to be set.
7. Read from Buffer port register for number of times programmed in WML[RD_WML] considering restriction mentioned in Software polling procedure.
8. Clear IRQSTAT[BRR].
9. Repeat 6-8 steps for rest of the data transfer.
10. Wait for the Transfer Complete interrupt.
11. Check the status bit to see if a read CRC error occurred, or some other error, occurred during the auto12 command sending and response receiving.

18.6.3.2.2 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 support read wait (SRW bit in CCCR register is 1), the driver can set the PROCTL[SABGREQ] to pause the transfer between the data blocks. Before setting the SABGREQ bit, make sure the PROCTL[RWCTL] is set, otherwise the eSDHC 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:
   a. For SD cards and eMMC devices, use SET_BLOCKLEN (CMD16)
   b. 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 eSDHC block length register to be the same as the block length set for the card in Step 4.
6. Set the eSDHC number block register (NOB), nob is 5 (for instance).
7. Disable the buffer read ready interrupt, configure the DMA setting and enable the eSDHC 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 other 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 eSDHC 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. Even 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 eSDHC 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, XFERTYP[MSBSEL] and XFERTYP[BCEN] are set, as well as XFERTYP[AC12EN]. However, the eSDHC will automatically send the CMD12 to mark the end of multi-block transfer.

18.6.3.3 Suspend resume

The eSDHC supports the suspend resume operations of SDIO cards, although slightly different than the suggested implementation of suspend in the SDIO card specification.

18.6.3.3.1 Suspend

After setting the SABGREQ bit, the host driver may send a Suspend command to switch to another function of the SDIO card. The eSDHC 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 should not mark the Suspend command as a "Suspend", (that is, setting the CMDTYP bits to 01). Instead, the driver should 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 eSDHC 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, save the context registers in the system memory for later use, including the SDMA system address register (DSADDR) for DMA operation, and the block attributes register (BLKATTR).
3. Send the Suspend command to suspend the active function. The CMDTYP bit field must be 2'b00.
4. 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.
5. Send another normal I/O command to the suspended function. The CMDTYP of this command must be 2'b01, so the eSDHC can detect this special setting and be informed that the paused operation has successfully suspended. If the paused transfer is a read operation, the eSDHC stops driving DAT2 and goes to the idle state.

6. Begin operation for another function on the SDIO card.

18.6.3.3.2 Resume

To resume the data transfer, a resume command should be issued:

1. To resume the suspended function, restore the context register with the saved value in step #2 of the suspend operation above.
2. Send the resume command. In the transfer type register (XFERTYP), 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.

18.6.3.4 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.

To accomplish this:

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 (4 Kbyte 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 attributes register (BLKATTR).
6. Set the ADMA system address register (ADSADDR) to the address of the first descriptor and set PROCTL[DMAS] to 01 to select the ADMA.
7. Issue a write or read command with XFERTYP[DMAEN] set.

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.

18.6.3.5 Tuning block procedure

Tuning block is used for SD UHS SDR50, SDR104 and eMMC HS200 modes.

Host driver should execute tuning procedure before initiating data transfer with these speed modes.

The steps for using tuning block are:

1. Clear SYSCTL[SDCLKEN]
2. Wait for PRSSTAT[SDSTB] to be set
3. Set ESDHCCTL[FAF]
4. Wait for ESDHCCTL[FAF] to be cleared
5. Set appropriate AUTOCERR[UHSM]
6. Set TBCTL[TB_EN] and program appropriate TBCTL[TB_MODE]
7. Set SYSCTL[SDCLKEN]
8. Wait for PRSSTAT[SDSTB] to be set
9. Execute tuning procedure

While tuning procedure is being performed, eSDHC doesn't generate any other command or data interrupt except buffer read ready in IRQSTAT register.

When tuning error is received, host driver should abort the current data transfer and execute the tuning procedure.

ESDHCCTL[PCS] (peripheral clock select), and TBCTL[TB_EN] (tuning block enable) should always be set whenever using tuning block. PCS should be set before TB_EN.

NOTE

- Similar steps needs to be performed to disable tuning block, except programming different values in 5 and 6. Also 9 need not to be performed.
- For tuning mode operation, the SD clock divisor value must be within 3 to 16.

18.6.3.5.1 Tuning procedure for hardware tuning modes

The steps for tuning procedure, when TBCTL[TB_MODE] is programmed to either Mode 1, Mode 2 or Mode 3, are:

1. Set SYSCTL2[EXTN], execute tuning.
2. Issue SEND_TUNING_BLK Command (CMD19 for SD, CMD21 for eMMC).
3. Wait for IRQSTAT[BRR], buffer read ready, to be set.
4. Clear IRQSTAT[BRR].
5. Check SYSCTL2[EXTN] to be cleared.
6. Repeat steps 2-5, if EXTN is not cleared.
7. Check SYSCTL2[SMPCLKSEL], sampling clock select. It's set value indicates tuning procedure success, and clears indicate failure. In case of tuning failure, fixed

sampling scheme could be used by clearing TBCTL[TB_EN], tuning block enable bit.

18.6.3.5.2 Tuning procedure for software tuning mode

The steps for tuning procedure, when TBCTL[TB_MODE] is set to SW tuning mode, are:

1. Program the start and end pointer for the data window in the TPR register.
2. Set SYSCTL2[EXTN] and SYSCTL2[SMPCLKSEL], execute tuning.
3. Issue SEND_TUNING_BLK Command (CMD19 for SD, CMD21 for eMMC).
4. Wait for IRQSTAT[BRR], buffer read ready, to be set.
5. Clear IRQSTAT[BRR].
6. Check SYSCTL2[EXTN] to be cleared.
7. Check SYSCTL2[SMPCLKSEL], sampling clock select. It's set value indicates tuning procedure success, and clears indicate failure. In case of tuning failure, fixed sampling scheme could be used by clearing TBCTL[TB_EN], tuning block enable bit.

18.6.3.6 DDR

The steps for using DDR mode are:

1. Clear SYSCTL[SDCLKEN]
2. Wait for PRSSTAT[SDSTB] to be set
3. Program AUTOCERR[UHSM] to 4
4. If required, change the clock division ratio in SYSCTL register
5. Set SDCLKCTL[CMD_CLK_CTL] and SDCLKCTL[LPBK_CLK_SEL] for DDR mode
6. Set SYSCTL[SDCLKEN]
7. Wait for PRSSTAT[SDSTB] to be set
8. Again clear SYSCTL[SDCLKEN]
9. Wait for PRSSTAT[SDSTB] to be set
10. Set ESDHCCTL[FAF]
11. Wait for ESDHCCTL[FAF] to be cleared
12. Set SYSCTL[SDCLKEN]
13. Wait for PRSSTAT[SDSTB] to be set

NOTE

1. SD clock divisor should be even for DDR mode.

2. Similar steps needs to be performed to disable, except programming different value in 3

18.6.3.7 Transfer error

18.6.3.7.1 CRC transfer 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 should 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 should 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 eSDHC 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 eSDHC. In this case, the driver should re-send or re-obtain the last block with a single block transfer.

18.6.3.7.2 DMA transfer error

During the data transfer with internal single DMA, if the DMA engine encounters some error on the System bus, the DMA operation is aborted and DMA error interrupt is sent to the host system.

When acknowledged by such an interrupt, the driver should calculate the start address of data block in which the error occurs. The start address can be calculated by:

Read the DMA error address register (DMAERRADDR). Taking the block size and the start address intially preogrammed in SDMA Adress register, it is straight forward to obtain the start address of the corrupted block.

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.

18.6.3.7.3 ADMA transfer error

There are three kinds of possible ADMA errors; The System 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. System 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.

18.6.3.7.4 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 eSDHC 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 should send a CMD12 manually.

18.6.3.8 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 DAT[1] line during some special period.

The eSDHC only monitors the DAT[1] line and supports the SDIO interrupt.

When the SDIO interrupt is captured by the eSDHC, and the Host System is informed by the eSDHC asserting the eSDHC interrupt line, the interrupt service from the host driver is called.

As the interrupt factor is controlled by the external card, the interrupt from the SDIO card must be served before the CINT bit is cleared by written 1. Refer to Card interrupt handling for the card interrupt handling flow.

18.6.4 Switch function

The Switch command is used to enable high speed and change the bus width for eMMC devices.

For SDIO cards, the high speed mode is enabled by writing the EHS bit in the CCCR register after the SHS bit is confirmed. For SD cards, the high speed mode is queried and enabled by a CMD6 (with the mnemonic symbol as SWITCH_FUNC). For eMMC devices, the high speed mode is queried by a CMD8 and enabled by a CMD6 (with the mnemonic symbol as SWITCH).

The 4-bit bus width of the eMMC 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 flowcharts do not show current capability check, which is recommended in the function switch process.

18.6.4.1 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 eSDHC 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 eSDHC to generate the
card_clk of the desired value below 25MHz;
```

```
(data transactions like normal peers)
}
```

18.6.4.2 Query, enable and disable SD high speed mode

```
enable_sd_high_speed_mode(void)
{
set BLKCNT field to 1 (block), set BLKSIZE field to 64 (bytes);
send CMD6, with argument 0xFFFFF1 and read 64 bytes of data accompanying the R1 response;
wait data transfer done bit is set;
check if the bit 401 of received 512 bit is set;
if (bit 401 is '0') report the SD card does not support high speed mode and return;
send CMD6, with argument 0x80FFFFF1 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 eSDHC to generate the
card_clk of around 50MHz;
(data transactions like normal peers)
}
disable_sd_high_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 eSDHC to generate the
card_clk of the desired value below 25MHz;
(data transactions like normal peers)
}
```

18.6.4.3 Query, enable and disable eMMC high speed mode

```
enable_mmc_high_speed_mode(void)
{
send CMD9 to get CSD value of eMMC;
check if the value of SPEC_VER field is 4 or above;
if (SPEC_VER value is less than 4) report the eMMC 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 eMMC;
extract the value of CARD_TYPE field to check the 'high speed mode' in this eMMC 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 eMMC;
check if HS_TIMING byte (byte number 185) is 1;
if (HS_TIMING is not 1) report eMMC switching to high speed mode failed and return;
change clock divisor value or configure the system clock feeding into eSDHC 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 eMMC;
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 eSDHC to generate the
card_clk of the desired value below 20MHz;
(data transactions like normal peers)
}
```

18.6.4.4 Set eMMC bus width

```
change_mmc_bus_width(void)
{
send CMD9 to get CSD value of eMMC;
check if the value of SPEC_VER field is 4 or above;
if (SPEC_VER value is less than 4) report the eMMC does not support multiple bit width and
return;
send CMD6 with argument 0x3B70x00; (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)
}
```

18.6.5 ADMA operation

18.6.5.1 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 align);
}
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;
}
```

18.6.5.2 ADMA2 operation

```
Set_adma2_descriptor
{
if (to start data transfer) {
// Make sure the address is 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';
}
```

18.7 Interfacing Card

eSDHC and card IO voltage might be different. External on board voltage translator can be used to interface the card in that case.

Following diagram illustrates direct interfacing of eSDHC with card when voltage level is same at both ends. SD_CLK_SYNC_OUT and SD_CLK_SYNC_IN should be routed as close as possible to card, with minimum skew with respect to SD_CLK.

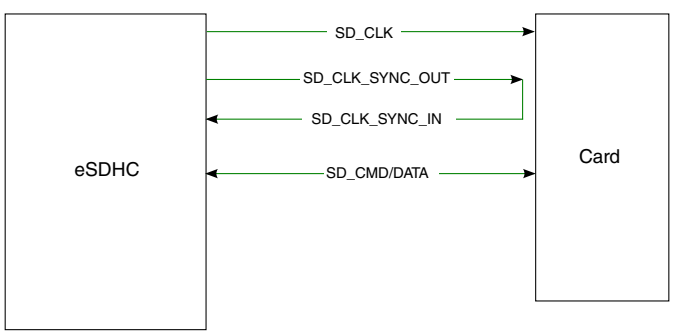


Figure 18-21. Direct Interfacing of eSDHC with Card

Below figure illustrates interfacing card with eSDHC through voltage translator when the voltages are different. CMD/DATA DIR and SD_VS pins might be required when interfacing with voltage translator that support DDR, and SDR more than 50 MHz modes. eSDHC SD_VS, or GPIO could be used to control SEL pin of the translator.

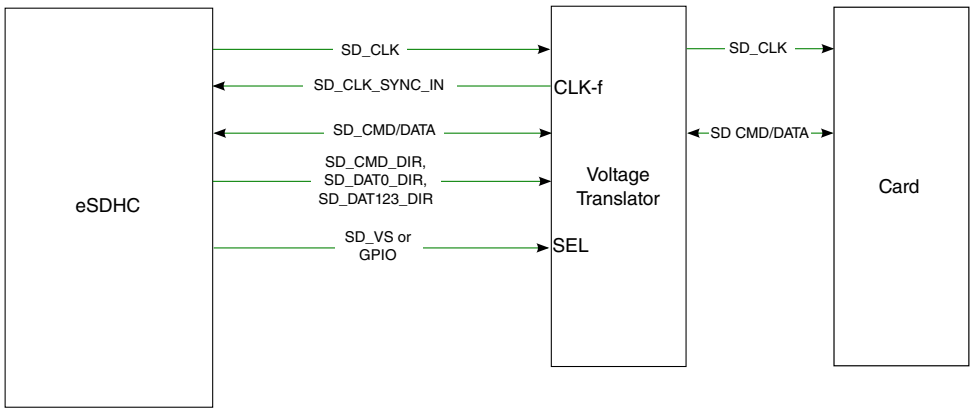


Figure 18-22. Card Interfacing with eSDHC through Voltage Translator

NOTE

CD, WP and DAT4-7 are not shown in above figures for simplicity.

18.8 Commands for /SD/SDIO and eMMC device

See the table below for the list of commands for the /SD/SDIO cards and eMMC device.

Refer the corresponding specifications for more details about the command information.

There are four kinds of commands defined to control the MultiMediaCard:

- Broadcast commands (bc), no response
- Broadcast commands with response (bcr), response from all cards simultaneously
- Addressed (point-to-point) commands (ac), no data transfer on the DAT
- Addressed (point-to-point) data transfer commands (adtc)

The access bits for the EXT_CSD access modes are shown in Table 18-15.

Table 18-14. Commands for /SD/SDIO cards and eMMC device

CMD INDEX	Type	Argument	Resp	Abbreviation	Description
CMD0	bc	[31:0] stuff bits	-	GO_IDLE_STATE	Resets all eMMC device and SD memory cards to idle state.
CMD1	bcr	[31:0] OCR without busy	R3	SEND_OP_COND	Asks all eMMC device 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.

Table continues on the next page...

Table 18-14. Commands for /SD/SDIO cards and eMMC device (continued)

CMD INDEX	Type	Argument	Resp	Abbreviation	Description
CMD3[1]	ac	[31:16] RCA [15:0] stuff bits	R1 R6 (SDIO)	SET/ SEND_RELATIVE_ADDR	Assigns relative address to the card.
CMD4	bc	[31:16] 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 "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 "The MultiMediaCard System Specification Version 4.0 Final draft 2" for more details.
CMD7	ac	[31:16] 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[4]	bcr	[31:12] reserved bits [11:8] supply voltage (VHS) [7:0] check pattern	R7	SEND_IF_COND	Sends SD memory card interface condition, which includes host supply voltage information and asks the card whether card supports voltage. Reserved bits shall be set to 0.
CMD8[5]	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.
CMD9	ac	[31:16] RCA [15:0] stuff bits	R2	SEND_CSD	Addressed card sends its card-specific data (CSD) on the CMD line.
CMD10	ac	[31:16] RCA [15:0] stuff bits	R2	SEND_CID	Addressed card sends its card-identification (CID) on the CMD line.

Table continues on the next page...

Table 18-14. Commands for /SD/SDIO cards and eMMC device (continued)

CMD INDEX	Type	Argument	Resp	Abbreviation	Description
CMD11	ac	[31:0] stuff bits	R1	VOLTAGE_SWITCH	Switch to 1.8V bus signalling level. Applicable for UHS SD card.
CMD12	ac	[31:0] stuff bits	R1b	STOP_TRANSMISSION	Forces the card to stop transmission.
CMD13	ac	[31:16] RCA [15:0] stuff bits	R1	SEND_STATUS	Addressed card sends its status register.
CMD14	Reserved				
CMD15	ac	[31:16] 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	adtc	[31:0] stuff bits	R1	SEND_TUNING_BLOCK	64 bytes tuning pattern is sent for SDR50 and SDR104. Applicable for UHS SD card.
CMD20	Reserved				
CMD21	adtc	[31:0] stuff bits	R1	SEND_TUNING_BLOCK	128 clocks of tuning pattern (64 byte in 4-bit mode or 128 byte in 8-bit mode) is sent for HS200 optimal sampling point detection. Applicable for eMMC device.
CMD22	Reserved				
CMD23	ac	[31:16] set to 0 [15:0] block count	R1	SET_BLOCKCOUNT	Defines the number of blocks which are going to be transferred in the immediately succeeding multiple block read or write command. If the argument is all 0s, the subsequent read/write operations are openended.
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 should 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 18-14. Commands for /SD/SDIO cards and eMMC device (continued)

CMD INDEX	Type	Argument	Resp	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	ERASE_WR_BLK_START	Sets the address of the first write block to be erased in SD card.
CMD32	ac	[31:0] data address	R1	TAG_SECTOR_START	Sets the address of the first sector of the erase group in eMMC device.
CMD33	ac	[31:0] data address	R1	ERASE_WR_BLK_END	Sets the address of the last write block of the continuous range to be erased in SD card.
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 in eMMC device.
CMD34	ac	[31:0] data address	R1	UNTAG_SECTOR	Removes one previously selected sector from the erase selection in eMMC device.
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 in eMMC device.
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 in eMMC device.
CMD37	ac	[31:0] data address	R1	UNTAG_ERASE_GROUP	Removes one previously selected erase group from the erase selection in eMMC device.
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 eMMC standard.
CMD40	bcr	[31:0] stuff bits	R5	GO_IRQ_STATE	Sets the system into interrupt mode in eMMC device.

Table continues on the next page...

Table 18-14. Commands for /SD/SDIO cards and eMMC device (continued)

CMD INDEX	Type	Argument	Resp	Abbreviation	Description
CMD41	Reserved				
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	As per SDIO spec CMD52 format	R5	IO_RW_DIRECT	Access a single register within the total 128k of register space in any I/O function.
CMD53	ac	As per SDIO spec CMD53 format	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.
CMD57-63	Reserved				
ACMD6	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[6]	adtc	[31:0] stuff bits	R1	SD_STATUS	Send the SD Memory Card status.
ACMD18[6]	adtc	[31:0] stuff bits	R1	SECURE_READ_MULTI_BLOCK	Protected Area Access Command: Reads continuously transfer data blocks from protected area of SD memory card. Refer Security Specification Version 2.00 for more details.
ACMD22[6]	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.
ACMD23[6]	ac	[31:0] stuff bits	R1	SET_WR_BLK_ERASE_COUNT	In SD cards, set the number of write blocks to be pre-erased before writing (to be used for faster multiple block write command) Default value is 1 (one write block).
ACMD25[6]	adtc	[31:0] stuff bits	R1	SECURE_WRITE_MULTI_BLOCK	Protected Area Access Command:

Table continues on the next page...

Table 18-14. Commands for /SD/SDIO cards and eMMC device (continued)

CMD INDEX	Type	Argument	Resp	Abbreviation	Description
					Writes continuously transfer data blocks to protected area of SD memory card. Refer Security Specification Version 2.00 for more details.
ACMD26[6]	adtc	[31:0] stuff bits	R1	SECURE_WRITE_MKB	System Area Access Command: Overwrite the existing media key block (MKB) on the system area of SD Memory Card with new MKB. This command is used in dynamic update MKB scheme. Refer Security Specification Version 2.00 for more details.
ACMD38[6]	ac	[31:0] stuff bits	R1b	SECURE_ERASE	Protected Area Access Command: Erase a specified region of the Protected Area of SD Memory Card. Refer Security Specification Version 2.00 for more details.
ACMD41[6]	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[6]	ac	[31:1] stuff bits [0] set_cd	R1	SET_CLR_CARD_DETECT	Connect [1]/Disconnect [0] the 50KΩ pull-up resistor on CD/DAT3 of SD card.
ACMD43[6]	adtc	[31:24]Unit_Count: [23:16] MKB_ID: [15:0]Unit_Offset:	R1	GET_MKB	Reads Media Key Block from the System Area of SD Memory Card. -Unit_Count specifies the Number of units to read. (Here, a unit=512 byte (fixed).) - MKB_ID specifies the application unique number. - Unit_Offset specifies the start address(offset) to read. Refer Security Specification Version 2.00 for more details.
ACMD44[6]	adtc	[31:0] stuff bits	R1	GET_MID	Reads Media ID from the system area of SD memory card. Refer Security Specification Version 2.00 for more details.
ACMD45[6]	adtc	[31:0] stuff bits	R1	SET_CER_RN1	AKE Command: Writes random number RN1 as challenge1 in AKE process. Refer Security Specification Version 2.00 for more details.
ACMD46[6]	adtc	[31:0] stuff bits	R1	GET_CER_RN2	AKE Command:

Table continues on the next page...

Table 18-14. Commands for /SD/SDIO cards and eMMC device (continued)

CMD INDEX	Type	Argument	Resp	Abbreviation	Description
					Reads random number RN2 as challenge2 in AKE process. Refer security specification version 2.00 for more details.
ACMD47[6]	adtc	[31:0] stuff bits	R1	SET_CER_RES2	AKE Command: Writes RES2 as response2 to RN2 in AKE process Refer Security Specification Version 2.00 for more details.
ACMD48[6]	adtc	[31:0] stuff bits	R1	GET_CER_RES1	AKE Command: Reads RES1 as response1 to RN1 in AKE process. Refer Security Specification Version 2.00 for more details.
ACMD49[6]	ac	[31:0] stuff bits	R1b	CHANGE_SECURE_AREA	Protected Area Access Command: Change size of the protected area. Refer Security Specification Version 2.00 for more details.
ACMD51[6]	adtc	[31:0] stuff bits	R1	SEND_SCR	Reads the SD configuration register (SCR).

1. CMD3 differs for eMMC devices and SD cards. For eMMC device, 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 eMMC devices and high speed SD cards. Command SWITCH_FUNC is for high speed SD cards.
3. Command SWITCH is for high speed eMMC devices. 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. CMD8 for SD stands for SEND_IF_COND.
5. CMD8 for eMMC stands for SEND_EXT_CSD.
6. ACMDs should 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).

Table 18-15. 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 bit in the Value field.
10	Clear bits	The bits in the pointed byte are cleared, according to the 1 bit in the Value field.
11	Write Byte	The Value field is written into the pointed byte.

18.9 Software restrictions

This section covers software restrictions.

18.9.1 Software polling procedure

For polling read or write, once the software begins a buffer read or write, it must access the buffer data port register (DATPORT) exactly the number of times as watermark level value set in the watermark level register (WML).

However, if the block size is not same as 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 read operations, if the RD_WML is 4, indicating the watermark level is 16 bytes, block size is 40 bytes, and the block count is 2, then the access times to data buffer port register for the burst sequence in the whole transfer process must be 4, 4, 2 (for first block); 4, 4, 2 (for second block).

18.9.2 Suspend operation

In order to suspend the data transfer, the software must inform eSDHC 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 eSDHC that the transfer is suspended.

If software needs to resume the suspended transfer, it should read the value in BLKCNT register to save the remained number of blocks before sending the normal command marked as suspend, otherwise on sending such "suspend" command, eSDHC will regard the current transfer a aborted and change the BLKCNT register to its original value, instead of keeping the remaining number of blocks.

18.9.3 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 eSDHC buffer.

18.9.4 Multi-block read

For pre-defined multi-block read operation, that is, the number of blocks to read has been defined by previous CMD23 for eMMC, or pre-defined number of blocks in CMD53 for SDIO/SDCombo, or SD security read commands, or whatever multi-block read without abort command at card side; soft reset for data(SYSCTL[RSTD]) is required after the transfer is complete to drive the internal state machine to idle mode.

18.9.5 ADMA address

For ADMA1/ADMA2 operation, the ADMA system address register (ADSADDR) and address defined in the address field of the descriptor table should be 4-byte aligned.

18.9.6 Allowed operations after stop at block gap

Only one of these operations is allowed after stop at block gap event:

- Continue data transfer by setting SYSCTL[CREQ]
- Issue suspend command
- Abort data transfer by issued abort command

No other command can be issued until one of the operations listed above is performed.

18.9.7 SDIO card interrupt during soft reset

The host driver should disable SDIO card interrupts in IRQSTAT[CINT] before issuing soft reset for data or all (generally used for error recovery) in the system control register (SYSCTL), and enable it after error recovery sequence has been completed.

18.9.8 Soft reset for data not allowed when SD clock is disabled

Soft reset for data and CMD (SYSCTL[RSTD]/SYSCTL[RSTC]) should not be issued when SD clock is disabled; that is, when SYSCTL[SDCLKEN] is cleared.

Instead, the host driver may issue soft reset for all (SYSCTL[RSTA]).

18.9.9 Data transfer with Auto CMD12 Enable

When Auto CMD12 is enabled for data transfer, it generates IRQSTAT[TC] for data transfer completion but does not generate TC for Auto CMD12(command with busy).

Host Driver needs make sure that card has reached to “trans“ state before issuing any new data command. CMD13(SEND_STATUS) could be sent to check the card status.

Chapter 19
FlexTimer Module (FTM)

19.1 The FlexTimer module as implemented on the chip

This section provides details about how the FlexTimer module is implemented on the chip.

19.1.1 LS1012A FlexTimer module integration

The following table describes the FlexTimer module integration into this chip:

Table 19-1. FlexTimer module integration

Module	Module Base address
FlexTimer Module 1	29D_0000
FlexTimer Module 2	29E_0000

The remainder of this chapter refers to a single FlexTimer module. Notes are included to indicate variations for multiple instantiations.

19.1.2 LS1012A FlexTimer signals

The following table lists the SoC signal names and their corresponding FlexTimer module signal names used in this chapter:

Table 19-2. LS1012A FlexTimer signals

LS1012A signal name	FlexTimer module signal
FTM_EXTCLK	EXTCLK
FTMn_CHn	CHn
Not used	FAULTj

Table continues on the next page...

Table 19-2. LS1012A FlexTimer signals (continued)

LS1012A signal name	FlexTimer module signal
Not used	PHA
Not used	PHB

19.1.3 LS1012A FlexTimer module special consideration

The FlexTimer module implements the following parameter settings in the chip:

Table 19-3. LS1012A FlexTimer parameter settings

FlexTimer parameters	LS1012A parameter value	
	FTM1	FTM2
Number of channels available at device I/O level	4	4
Quadrature decoding support	Yes	Yes
EXTCLK support	Yes (single EXTCLK is shared by both FTM controllers)	Yes
Fixed frequency clock support	32 KHz RTC	32 KHz RTC
System clock support	DDR clock/8	DDR clock/8
Stop mode support	Yes. Refers to the LPM20 low power mode of the chip.	

The table below shows the FlexTimer chaining for 32-bit counter. See the FTM chain configuration register (FTM_CHAIN_CONFIG) for more details.

Table 19-4. FlexTimer chaining for 32-bit counter

FlexTimer-A	FlexTimer-B	Control bit	Control bit default value
FTM1	FTM2	FTM_CHAIN_CONFIG[FTM_CHN1], FTM_CHAIN_CONFIG[FTM_CHN2]	0 = Chaining disabled

It is possible to chain two FlexTimer controllers to get a bigger 32-bit counter. This is achieved by:

- Connecting CH7 output of FlexTimer-B to PHA input of FlexTimer-A.
- Tieing PHB input of FlexTimer-A to one.
- Programming FlexTimer-A to be in quad-mode.

FlexTimer-A has [31:16] of the 32-bit counter while FlexTimer-B has [15:0] of the 32-bit counter. The above table also shows how the chaining is implemented for FlexTimer-A and FlexTimer-B.

19.2 Introduction

The FlexTimer module (FTM) is a two-to-eight channel timer that supports input capture, output compare, and the generation of PWM signals to control electric motor and power management applications. The FTM time reference is a 16-bit counter that can be used as an unsigned or signed counter.

NOTE

The number of channels supported can vary for each instance of the FTM module on a chip. See the chip-specific FTM information to see how many channels are supported for each module instance. For example, if a module instance supports only six channels, references to channel numbers 6 and 7 do not apply for that instance.

19.2.1 FlexTimer philosophy

The FlexTimer is built upon a simple timer, the Timer PWM Module – TPM, used for many years on our HCS08 family of 8-bit microcontrollers. The FlexTimer extends the functionality to meet the demands of motor control, digital lighting solutions, and power conversion, while providing low cost and backwards compatibility with the TPM module.

Several key enhancements are made:

- Signed up counter
- Deadtime insertion hardware
- Fault control inputs
- Enhanced triggering functionality
- Initialization and polarity control

All of the features common with the TPM have fully backwards compatible register assignments. The FlexTimer can also use code on the same core platform without change to perform the same functions.

Motor control and power conversion features have been added through a dedicated set of registers and defaults turn off all new features. The new features, such as hardware deadtime insertion, polarity, fault control, and output forcing and masking, greatly reduce loading on the execution software and are usually each controlled by a group of registers.

FlexTimer input triggers can be from comparators, ADC, or other submodules to initiate timer functions automatically. These triggers can be linked in a variety of ways during integration of the sub modules so please note the options available for used FlexTimer configuration.

More than one FlexTimers may be synchronized to provide a larger timer with their counters incrementing in unison, assuming the initialization, the input clocks, the initial and final counting values are the same in each FlexTimer.

All main user access registers are buffered to ease the load on the executing software. A number of trigger options exist to determine which registers are updated with this user defined data.

19.2.2 Features

The FTM features include:

- FTM source clock is selectable.
    - The source clock can be the system clock, the fixed frequency clock, or an external clock
    - Fixed frequency clock is an additional clock input to allow the selection of an on chip clock source other than the system clock
    - Selecting external clock connects FTM clock to a chip level input pin therefore allowing to synchronize the FTM counter with an off chip clock source
- Prescaler divide-by 1, 2, 4, 8, 16, 32, 64, or 128
- 16-bit counter
    - It can be a free-running counter or a counter with initial and final value
    - The counting can be up or up-down
- Each channel can be configured for input capture, output compare, or edge-aligned PWM mode
- In Input Capture mode:
    - The capture can occur on rising edges, falling edges or both edges
    - An input filter can be selected for some channels

- In Output Compare mode the output signal can be set, cleared, or toggled on match
- All channels can be configured for center-aligned PWM mode
- Each pair of channels can be combined to generate a PWM signal with independent control of both edges of PWM signal
- The FTM channels can operate as pairs with equal outputs, pairs with complementary outputs, or independent channels with independent outputs
- The deadtime insertion is available for each complementary pair
- Generation of match triggers
- Initialization trigger
- Software control of PWM outputs
- Up to 4 fault inputs for global fault control
- The polarity of each channel is configurable
- The generation of an interrupt per channel
- The generation of an interrupt when the counter overflows
- The generation of an interrupt when the fault condition is detected
- Synchronized loading of write buffered FTM registers
- Write protection for critical registers
- Backwards compatible with TPM
- Testing of input captures for a stuck at zero and one conditions
- Dual edge capture for pulse and period width measurement
- Quadrature decoder with input filters, relative position counting, and interrupt on position count or capture of position count on external event

19.2.3 Modes of operation

When the chip is in an active mode, the FTM temporarily suspends all counting until the chip returns to normal user operating mode. During Stop mode, all FTM input clocks are stopped, so the FTM is effectively disabled until clocks resume. During Wait mode, the

FTM continues to operate normally. If the FTM does not need to produce a real time reference or provide the interrupt sources needed to wake the chip from Wait mode, the power can then be saved by disabling FTM functions before entering Wait mode.

19.2.4 Block diagram

The FTM uses one input/output (I/O) pin per channel, CHn (FTM channel (n)) where n is the channel number (0–7).

The following figure shows the FTM structure. The central component of the FTM is the 16-bit counter with programmable initial and final values and its counting can be up or up-down.

NOTE

The number of channels supported can vary for each instance of the FTM module on a chip. See the chip-specific FTM information to see how many channels are supported for each module instance. For example, if a module instance supports only six channels, references to channel numbers 6 and 7 do not apply for that instance.

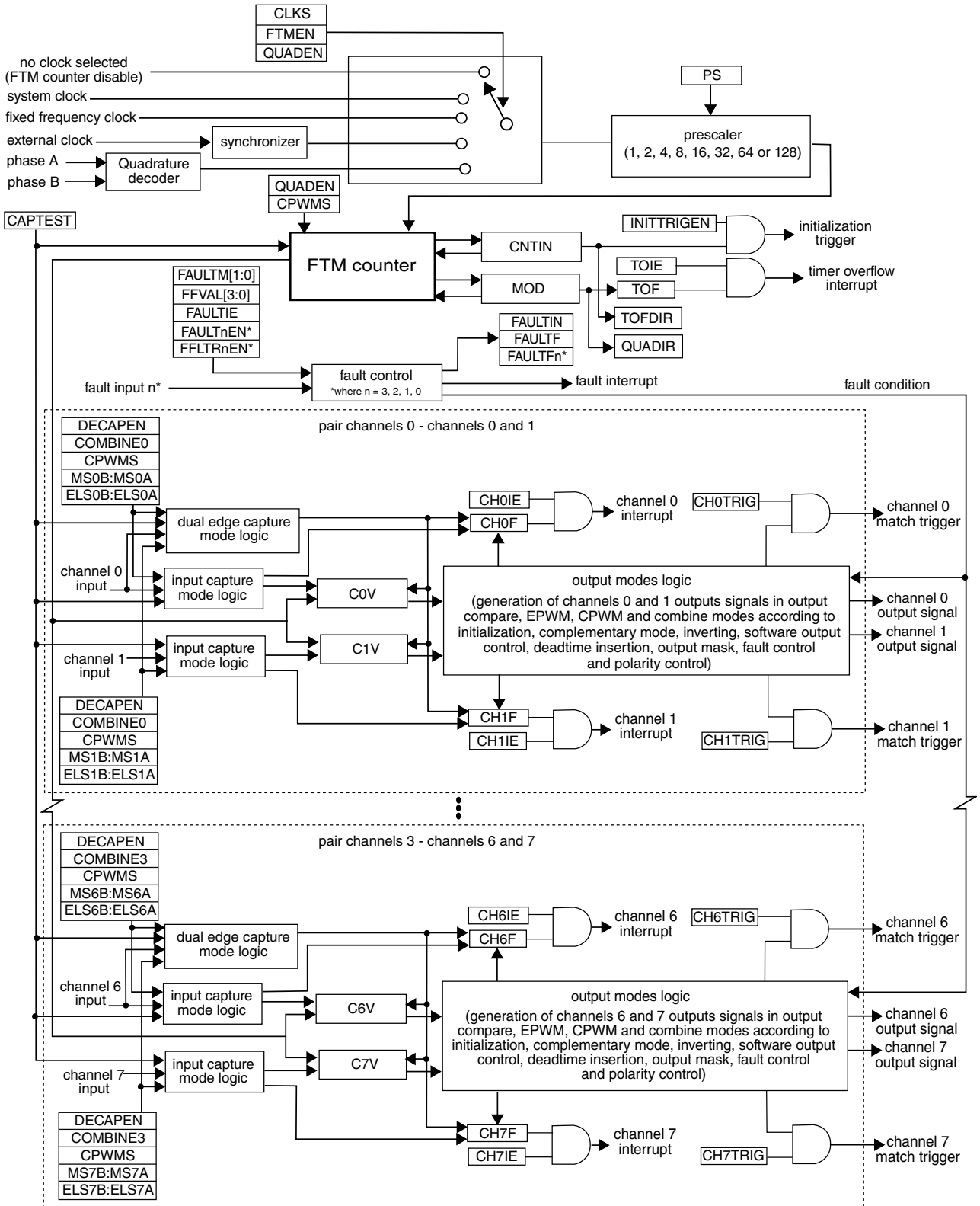


Figure 19-1. FTM block diagram

19.3 FTM signal descriptions

Table 19-5 shows the user-accessible signals for the FTM.

Table 19-5. FTM signal descriptions

Signal	Description	I/O	Function
EXTCLK	External clock. FTM external clock can be selected to drive the FTM counter.	I	The external clock input signal is used as the FTM counter clock if selected by CLKS[1:0] bits in the SC register. This clock signal must not exceed 1/4 of system clock frequency. The FTM counter prescaler selection and settings are also used when an external clock is selected.
CHn	FTM channel (n), where n can be 7-0	I/O	Each FTM channel can be configured to operate either as input or output. The direction associated with each channel, input or output, is selected according to the mode assigned for that channel.
FAULTj	Fault input (j), where j can be 3-0	I	The fault input signals are used to control the CHn channel output state. If a fault is detected, the FAULTj signal is asserted and the channel output is put in a safe state. The behavior of the fault logic is defined by the FAULTM[1:0] control bits in the MODE register and FAULTEN bit in the COMBINEm register. Note that each FAULTj input may affect all channels selectively since FAULTM[1:0] and FAULTEN control bits are defined for each pair of channels. Because there are several FAULTj inputs, maximum of 4 for the FTM module, each one of these inputs is activated by the FAULTjEN bit in the FLTCTRL register.
PHA	Quadrature decoder phase A input. Input pin associated with quadrature decoder phase A.	I	The quadrature decoder phase A input is used as the Quadrature Decoder mode is selected. The phase A input signal is one of the signals that control the FTM counter increment or decrement in the Quadrature Decoder mode.
PHB	Quadrature decoder phase B input. Input pin associated with quadrature decoder phase B.	I	The quadrature decoder phase B input is used as the Quadrature Decoder mode is selected. The phase B input signal is one of the signals that control the FTM counter increment or decrement in the Quadrature Decoder mode.

19.4 Memory map and register definition

19.4.1 Memory map

This section presents a high-level summary of the FTM registers and how they are mapped.

The registers and bits of an unavailable function in the FTM remain in the memory map and in the reset value, but they have no active function.

Note

Do not write in the region from the CNTIN register through the PWMLOAD register when FTMEN = 0.

NOTE

The number of channels supported can vary for each instance of the FTM module on a chip. See the chip-specific FTM information to see how many channels are supported for each module instance. For example, if a module instance supports only six channels, references to channel numbers 6 and 7 do not apply for that instance.

19.4.2 Register descriptions

Accesses to reserved addresses result in transfer errors. Registers for absent channels are considered reserved.

FTM memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
29D_0000	Status And Control (FTM1_SC)	32	R/W	0000_0000h	19.4.3/866
29D_0004	Counter (FTM1_CNT)	32	R/W	0000_0000h	19.4.4/868
29D_0008	Modulo (FTM1_MOD)	32	R/W	0000_0000h	19.4.5/868
29D_000C	Channel (n) Status And Control (FTM1_C0SC)	32	R/W	0000_0000h	19.4.6/869
29D_0010	Channel (n) Value (FTM1_C0V)	32	R/W	0000_0000h	19.4.7/871
29D_0014	Channel (n) Status And Control (FTM1_C1SC)	32	R/W	0000_0000h	19.4.6/869
29D_0018	Channel (n) Value (FTM1_C1V)	32	R/W	0000_0000h	19.4.7/871
29D_001C	Channel (n) Status And Control (FTM1_C2SC)	32	R/W	0000_0000h	19.4.6/869
29D_0020	Channel (n) Value (FTM1_C2V)	32	R/W	0000_0000h	19.4.7/871
29D_0024	Channel (n) Status And Control (FTM1_C3SC)	32	R/W	0000_0000h	19.4.6/869
29D_0028	Channel (n) Value (FTM1_C3V)	32	R/W	0000_0000h	19.4.7/871
29D_002C	Channel (n) Status And Control (FTM1_C4SC)	32	R/W	0000_0000h	19.4.6/869
29D_0030	Channel (n) Value (FTM1_C4V)	32	R/W	0000_0000h	19.4.7/871
29D_0034	Channel (n) Status And Control (FTM1_C5SC)	32	R/W	0000_0000h	19.4.6/869
29D_0038	Channel (n) Value (FTM1_C5V)	32	R/W	0000_0000h	19.4.7/871
29D_003C	Channel (n) Status And Control (FTM1_C6SC)	32	R/W	0000_0000h	19.4.6/869
29D_0040	Channel (n) Value (FTM1_C6V)	32	R/W	0000_0000h	19.4.7/871
29D_0044	Channel (n) Status And Control (FTM1_C7SC)	32	R/W	0000_0000h	19.4.6/869
29D_0048	Channel (n) Value (FTM1_C7V)	32	R/W	0000_0000h	19.4.7/871
29D_004C	Counter Initial Value (FTM1_CNTIN)	32	R/W	0000_0000h	19.4.8/872

Table continues on the next page...

FTM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
29D_0050	Capture And Compare Status (FTM1_STATUS)	32	R/W	0000_0000h	19.4.9/873
29D_0054	Features Mode Selection (FTM1_MODE)	32	R/W	0000_0004h	19.4.10/ 875
29D_0058	Synchronization (FTM1_SYNC)	32	R/W	0000_0000h	19.4.11/ 876
29D_005C	Initial State For Channels Output (FTM1_OUTINIT)	32	R/W	0000_0000h	19.4.12/ 879
29D_0060	Output Mask (FTM1_OUTMASK)	32	R/W	0000_0000h	19.4.13/ 880
29D_0064	Function For Linked Channels (FTM1_COMBINE)	32	R/W	0000_0000h	19.4.14/ 882
29D_0068	Deadtime Insertion Control (FTM1_DEADTIME)	32	R/W	0000_0000h	19.4.15/ 887
29D_006C	FTM External Trigger (FTM1_EXTTRIG)	32	R/W	0000_0000h	19.4.16/ 888
29D_0070	Channels Polarity (FTM1_POL)	32	R/W	0000_0000h	19.4.17/ 890
29D_0074	Fault Mode Status (FTM1_FMS)	32	R/W	0000_0000h	19.4.18/ 892
29D_0078	Input Capture Filter Control (FTM1_FILTER)	32	R/W	0000_0000h	19.4.19/ 894
29D_007C	Fault Control (FTM1_FLTCTRL)	32	R/W	0000_0000h	19.4.20/ 895
29D_0080	Quadrature Decoder Control And Status (FTM1_QDCTRL)	32	R/W	0000_0000h	19.4.21/ 897
29D_0084	Configuration (FTM1_CONF)	32	R/W	0000_0000h	19.4.22/ 899
29D_0088	FTM Fault Input Polarity (FTM1_FLTPOL)	32	R/W	0000_0000h	19.4.23/ 900
29D_008C	Synchronization Configuration (FTM1_SYNCONF)	32	R/W	0000_0000h	19.4.24/ 902
29D_0090	FTM Inverting Control (FTM1_INVCTRL)	32	R/W	0000_0000h	19.4.25/ 904
29D_0094	FTM Software Output Control (FTM1_SWOCTRL)	32	R/W	0000_0000h	19.4.26/ 905
29D_0098	FTM PWM Load (FTM1_PWMLOAD)	32	R/W	0000_0000h	19.4.27/ 907
29E_0000	Status And Control (FTM2_SC)	32	R/W	0000_0000h	19.4.3/866
29E_0004	Counter (FTM2_CNT)	32	R/W	0000_0000h	19.4.4/868
29E_0008	Modulo (FTM2_MOD)	32	R/W	0000_0000h	19.4.5/868
29E_000C	Channel (n) Status And Control (FTM2_C0SC)	32	R/W	0000_0000h	19.4.6/869
29E_0010	Channel (n) Value (FTM2_C0V)	32	R/W	0000_0000h	19.4.7/871
29E_0014	Channel (n) Status And Control (FTM2_C1SC)	32	R/W	0000_0000h	19.4.6/869

Table continues on the next page...

FTM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
29E_0018	Channel (n) Value (FTM2_C1V)	32	R/W	0000_0000h	19.4.7/871
29E_001C	Channel (n) Status And Control (FTM2_C2SC)	32	R/W	0000_0000h	19.4.6/869
29E_0020	Channel (n) Value (FTM2_C2V)	32	R/W	0000_0000h	19.4.7/871
29E_0024	Channel (n) Status And Control (FTM2_C3SC)	32	R/W	0000_0000h	19.4.6/869
29E_0028	Channel (n) Value (FTM2_C3V)	32	R/W	0000_0000h	19.4.7/871
29E_002C	Channel (n) Status And Control (FTM2_C4SC)	32	R/W	0000_0000h	19.4.6/869
29E_0030	Channel (n) Value (FTM2_C4V)	32	R/W	0000_0000h	19.4.7/871
29E_0034	Channel (n) Status And Control (FTM2_C5SC)	32	R/W	0000_0000h	19.4.6/869
29E_0038	Channel (n) Value (FTM2_C5V)	32	R/W	0000_0000h	19.4.7/871
29E_003C	Channel (n) Status And Control (FTM2_C6SC)	32	R/W	0000_0000h	19.4.6/869
29E_0040	Channel (n) Value (FTM2_C6V)	32	R/W	0000_0000h	19.4.7/871
29E_0044	Channel (n) Status And Control (FTM2_C7SC)	32	R/W	0000_0000h	19.4.6/869
29E_0048	Channel (n) Value (FTM2_C7V)	32	R/W	0000_0000h	19.4.7/871
29E_004C	Counter Initial Value (FTM2_CNTIN)	32	R/W	0000_0000h	19.4.8/872
29E_0050	Capture And Compare Status (FTM2_STATUS)	32	R/W	0000_0000h	19.4.9/873
29E_0054	Features Mode Selection (FTM2_MODE)	32	R/W	0000_0004h	19.4.10/ 875
29E_0058	Synchronization (FTM2_SYNC)	32	R/W	0000_0000h	19.4.11/ 876
29E_005C	Initial State For Channels Output (FTM2_OUTINIT)	32	R/W	0000_0000h	19.4.12/ 879
29E_0060	Output Mask (FTM2_OUTMASK)	32	R/W	0000_0000h	19.4.13/ 880
29E_0064	Function For Linked Channels (FTM2_COMBINE)	32	R/W	0000_0000h	19.4.14/ 882
29E_0068	Deadtime Insertion Control (FTM2_DEADTIME)	32	R/W	0000_0000h	19.4.15/ 887
29E_006C	FTM External Trigger (FTM2_EXTTRIG)	32	R/W	0000_0000h	19.4.16/ 888
29E_0070	Channels Polarity (FTM2_POL)	32	R/W	0000_0000h	19.4.17/ 890
29E_0074	Fault Mode Status (FTM2_FMS)	32	R/W	0000_0000h	19.4.18/ 892
29E_0078	Input Capture Filter Control (FTM2_FILTER)	32	R/W	0000_0000h	19.4.19/ 894
29E_007C	Fault Control (FTM2_FLTCTRL)	32	R/W	0000_0000h	19.4.20/ 895
29E_0080	Quadrature Decoder Control And Status (FTM2_QDCTRL)	32	R/W	0000_0000h	19.4.21/ 897
29E_0084	Configuration (FTM2_CONF)	32	R/W	0000_0000h	19.4.22/ 899

Table continues on the next page...

FTM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
29E_0088	FTM Fault Input Polarity (FTM2_FLTPOL)	32	R/W	0000_0000h	19.4.23/ 900
29E_008C	Synchronization Configuration (FTM2_SYNCONF)	32	R/W	0000_0000h	19.4.24/ 902
29E_0090	FTM Inverting Control (FTM2_INVCTRL)	32	R/W	0000_0000h	19.4.25/ 904
29E_0094	FTM Software Output Control (FTM2_SWOCTRL)	32	R/W	0000_0000h	19.4.26/ 905
29E_0098	FTM PWM Load (FTM2_PWMLOAD)	32	R/W	0000_0000h	19.4.27/ 907

19.4.3 Status And Control (FTMx_SC)

SC contains the overflow status flag and control bits used to configure the interrupt enable, FTM configuration, clock source, and prescaler factor. These controls relate to all channels within this module.

Address: Base address + 0h offset

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

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

FTMx_SC field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 TOF	Timer Overflow Flag Set by hardware when the FTM counter passes the value in the MOD register. The TOF bit is cleared by reading the SC register while TOF is set and then writing a 0 to TOF bit. Writing a 1 to TOF has no effect. If another FTM overflow occurs between the read and write operations, the write operation has no effect; therefore, TOF remains set indicating an overflow has occurred. In this case, a TOF interrupt request is not lost due to the clearing sequence for a previous TOF. 0 FTM counter has not overflowed. 1 FTM counter has overflowed.
25 TOIE	Timer Overflow Interrupt Enable Enables FTM overflow interrupts. 0 Disable TOF interrupts. Use software polling. 1 Enable TOF interrupts. An interrupt is generated when TOF equals one.
26 CPWMS	Center-Aligned PWM Select Selects CPWM mode. This mode configures the FTM to operate in Up-Down Counting mode. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 FTM counter operates in Up Counting mode. 1 FTM counter operates in Up-Down Counting mode.
27–28 CLKS	Clock Source Selection Selects FTM counter clock sources. This field is write protected. It can be written only when MODE[WPDIS] = 1. 00 No clock selected. This in effect disables the FTM counter. 01 System clock 10 Fixed frequency clock 11 External clock
29–31 PS	Prescale Factor Selection Selects one of 8 division factors for the clock source selected by CLKS. The new prescaler factor affects the clock source on the next system clock cycle after the new value is updated into the register bits. This field is write protected. It can be written only when MODE[WPDIS] = 1. 000 Divide by 1 001 Divide by 2 010 Divide by 4 011 Divide by 8 100 Divide by 16 101 Divide by 32 110 Divide by 64 111 Divide by 128

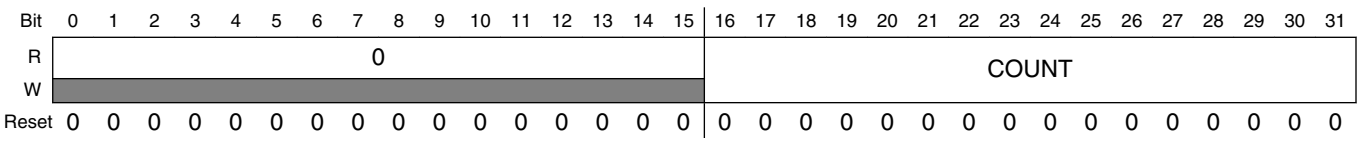
19.4.4 Counter (FTMx_CNT)

The CNT register contains the FTM counter value.

Reset clears the CNT register. Writing any value to COUNT updates the counter with its initial value, CNTIN.

When is active, the FTM counter is frozen. This is the value that you may read.

Address: Base address + 4h offset



FTMx_CNT field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 COUNT	Counter Value

19.4.5 Modulo (FTMx_MOD)

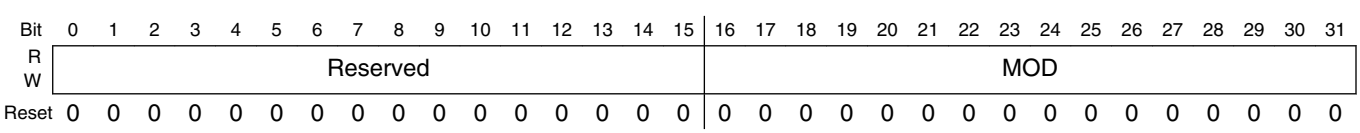
The Modulo register contains the modulo value for the FTM counter. After the FTM counter reaches the modulo value, the overflow flag (TOF) becomes set at the next clock, and the next value of FTM counter depends on the selected counting method; see Counter.

Writing to the MOD register latches the value into a buffer. The MOD register is updated with the value of its write buffer according to Registers updated from write buffers.

If FTMEN = 0, this write coherency mechanism may be manually reset by writing to the SC register whether is active or not.

Initialize the FTM counter, by writing to CNT, before writing to the MOD register to avoid confusion about when the first counter overflow will occur.

Address: Base address + 8h offset



FTMx_MOD field descriptions

Field	Description
0–15 Reserved	This field is reserved.
16–31 MOD	Modulo Value

19.4.6 Channel (n) Status And Control (FTM*x*_C*n*SC)

CnSC contains the channel-interrupt-status flag and control bits used to configure the interrupt enable, channel configuration, and pin function.

Table 19-6. Mode, edge, and level selection

DECAPEN	COMBINE	CPWMS	MSnB:MSnA	ELSnB:ELSnA	Mode	Configuration
X	X	X	XX	00	Pin not used for FTM—revert the channel pin to general purpose I/O or other peripheral control	
0	0	0	00	01	Input Capture	Capture on Rising Edge Only
				10		Capture on Falling Edge Only
				11		Capture on Rising or Falling Edge
			01	01	Output Compare	Toggle Output on match
				10		Clear Output on match
				11		Set Output on match
			1X	10	Edge-Aligned PWM	High-true pulses (clear Output on match)
				X1		Low-true pulses (set Output on match)
		1	XX	10	Center-Aligned PWM	High-true pulses (clear Output on match-up)
				X1		Low-true pulses (set Output on match-up)

Table continues on the next page...

Table 19-6. Mode, edge, and level selection (continued)

DECAPEN	COMBINE	CPWMS	MSnB:MSnA	ELSnB:ELSnA	Mode	Configuration
	1	0	XX	10	Combine PWM	High-true pulses (set on channel (n) match, and clear on channel (n+1) match)
				X1		Low-true pulses (clear on channel (n) match, and set on channel (n +1) match)
1	0	0	X0	See the following table (Table 19-7).	Dual Edge Capture	One-Shot Capture mode
			X1			Continuous Capture mode

Table 19-7. Dual Edge Capture mode — edge polarity selection

ELSnB	ELSnA	Channel Port Enable	Detected Edges
0	0	Disabled	No edge
0	1	Enabled	Rising edge
1	0	Enabled	Falling edge
1	1	Enabled	Rising and falling edges

Address: Base address + Ch offset + (8d × i), where i=0d to 7d

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

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								CHF	CHIE	MSB	MSA	ELSB	ELSA	0	DMA
W									0							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FTMx_C*n*SC field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 CHF	Channel Flag Set by hardware when an event occurs on the channel. CHF is cleared by reading the CSC register while CHnF is set and then writing a 0 to the CHF bit. Writing a 1 to CHF has no effect.

Table continues on the next page...

FTMx_C*n*SC field descriptions (continued)

Field	Description
	If another event occurs between the read and write operations, the write operation has no effect; therefore, CHF remains set indicating an event has occurred. In this case a CHF interrupt request is not lost due to the clearing sequence for a previous CHF. 0 No channel event has occurred. 1 A channel event has occurred.
25 CHIE	Channel Interrupt Enable Enables channel interrupts. 0 Disable channel interrupts. Use software polling. 1 Enable channel interrupts.
26 MSB	Channel Mode Select Used for further selections in the channel logic. Its functionality is dependent on the channel mode. See Table 19-6 . This field is write protected. It can be written only when MODE[WPDIS] = 1.
27 MSA	Channel Mode Select Used for further selections in the channel logic. Its functionality is dependent on the channel mode. See Table 19-6 . This field is write protected. It can be written only when MODE[WPDIS] = 1.
28 ELSB	Edge or Level Select The functionality of ELSB and ELSA depends on the channel mode. See Table 19-6 . This field is write protected. It can be written only when MODE[WPDIS] = 1.
29 ELSA	Edge or Level Select The functionality of ELSB and ELSA depends on the channel mode. See Table 19-6 . This field is write protected. It can be written only when MODE[WPDIS] = 1.
30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 DMA	DMA Enable Enables DMA transfers for the channel. 0 Disable DMA transfers. 1 Enable DMA transfers.

19.4.7 Channel (n) Value (FTMx_C*n*V)

These registers contain the captured FTM counter value for the input modes or the match value for the output modes.

In Input Capture, Capture Test, and Dual Edge Capture modes, any write to a CnV register is ignored.

In output modes, writing to a CnV register latches the value into a buffer. A CnV register is updated with the value of its write buffer according to Registers updated from write buffers.

If FTMEN = 0, this write coherency mechanism may be manually reset by writing to the CnSC register whether mode is active or not.

Address: Base address + 10h offset + (8d × i), where i=0d to 7d

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

FTMx_C*n*V field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16–31 VAL	Channel Value Captured FTM counter value of the input modes or the match value for the output modes

19.4.8 Counter Initial Value (FTMx_CNTIN)

The Counter Initial Value register contains the initial value for the FTM counter.

Writing to the CNTIN register latches the value into a buffer. The CNTIN register is updated with the value of its write buffer according to Registers updated from write buffers.

When the FTM clock is initially selected, by writing a non-zero value to the CLKS bits, the FTM counter starts with the value 0x0000. To avoid this behavior, before the first write to select the FTM clock, write the new value to the the CNTIN register and then initialize the FTM counter by writing any value to the CNT register.

Address: Base address + 4Ch offset

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

FTMx_CNTIN field descriptions

Field	Description
0–15 Reserved	This field is reserved.
16–31 INIT	Initial Value Of The FTM Counter

19.4.9 Capture And Compare Status (FTMx_STATUS)

The STATUS register contains a copy of the status flag CHnF bit in CnSC for each FTM channel for software convenience.

Each CHnF bit in STATUS is a mirror of CHnF bit in CnSC. All CHnF bits can be checked using only one read of STATUS. All CHnF bits can be cleared by reading STATUS followed by writing 0x00 to STATUS.

Hardware sets the individual channel flags when an event occurs on the channel. CHnF is cleared by reading STATUS while CHnF is set and then writing a 0 to the CHnF bit. Writing a 1 to CHnF has no effect.

If another event occurs between the read and write operations, the write operation has no effect; therefore, CHnF remains set indicating an event has occurred. In this case, a CHnF interrupt request is not lost due to the clearing sequence for a previous CHnF.

Address: Base address + 50h offset

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

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								CH7F	CH6F	CH5F	CH4F	CH3F	CH2F	CH1F	CH0F
W									0	0	0	0	0	0	0	0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FTMx_STATUS field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

FTMx_STATUS field descriptions (continued)

Field	Description
24 CH7F	Channel 7 Flag See the register description. 0 No channel event has occurred. 1 A channel event has occurred.
25 CH6F	Channel 6 Flag See the register description. 0 No channel event has occurred. 1 A channel event has occurred.
26 CH5F	Channel 5 Flag See the register description. 0 No channel event has occurred. 1 A channel event has occurred.
27 CH4F	Channel 4 Flag See the register description. 0 No channel event has occurred. 1 A channel event has occurred.
28 CH3F	Channel 3 Flag See the register description. 0 No channel event has occurred. 1 A channel event has occurred.
29 CH2F	Channel 2 Flag See the register description. 0 No channel event has occurred. 1 A channel event has occurred.
30 CH1F	Channel 1 Flag See the register description. 0 No channel event has occurred. 1 A channel event has occurred.
31 CH0F	Channel 0 Flag See the register description. 0 No channel event has occurred. 1 A channel event has occurred.

19.4.10 Features Mode Selection (FTMx_MODE)

This register contains the global enable bit for FTM-specific features and the control bits used to configure:

- Fault control mode and interrupt
- Capture Test mode
- PWM synchronization
- Write protection
- Channel output initialization

These controls relate to all channels within this module.

Address: Base address + 54h offset

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

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								FAULTIE	FAULTM		CAPTEST	PWMSYNC	WPDIS	INIT	FTMEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

FTMx_MODE field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 FAULTIE	Fault Interrupt Enable Enables the generation of an interrupt when a fault is detected by FTM and the FTM fault control is enabled. 0 Fault control interrupt is disabled. 1 Fault control interrupt is enabled.
25–26 FAULTM	Fault Control Mode Defines the FTM fault control mode. This field is write protected. It can be written only when MODE[WPDIS] = 1. 00 Fault control is disabled for all channels. 01 Fault control is enabled for even channels only (channels 0, 2, 4, and 6), and the selected mode is the manual fault clearing.

Table continues on the next page...

FTMx_MODE field descriptions (continued)

Field	Description
	10 Fault control is enabled for all channels, and the selected mode is the manual fault clearing. 11 Fault control is enabled for all channels, and the selected mode is the automatic fault clearing.
27 CAPTEST	Capture Test Mode Enable Enables the capture test mode. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 Capture test mode is disabled. 1 Capture test mode is enabled.
28 PWMSYNC	PWM Synchronization Mode Selects which triggers can be used by MOD, CnV, OUTMASK, and FTM counter synchronization. See PWM synchronization. The PWMSYNC bit configures the synchronization when SYNCMODE is 0. 0 No restrictions. Software and hardware triggers can be used by MOD, CnV, OUTMASK, and FTM counter synchronization. 1 Software trigger can only be used by MOD and CnV synchronization, and hardware triggers can only be used by OUTMASK and FTM counter synchronization.
29 WPDIS	Write Protection Disable When write protection is enabled (WPDIS = 0), write protected bits cannot be written. When write protection is disabled (WPDIS = 1), write protected bits can be written. The WPDIS bit is the negation of the WPEN bit. WPDIS is cleared when 1 is written to WPEN. WPDIS is set when WPEN bit is read as a 1 and then 1 is written to WPDIS. Writing 0 to WPDIS has no effect. 0 Write protection is enabled. 1 Write protection is disabled.
30 INIT	Initialize The Channels Output When a 1 is written to INIT bit the channels output is initialized according to the state of their corresponding bit in the OUTINIT register. Writing a 0 to INIT bit has no effect. The INIT bit is always read as 0.
31 FTMEN	FTM Enable This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 TPM compatibility. Free running counter and synchronization compatible with TPM. 1 Free running counter and synchronization are different from TPM behavior.

19.4.11 Synchronization (FTMx_SYNC)

This register configures the PWM synchronization.

A synchronization event can perform the synchronized update of MOD, CV, and OUTMASK registers with the value of their write buffer and the FTM counter initialization.

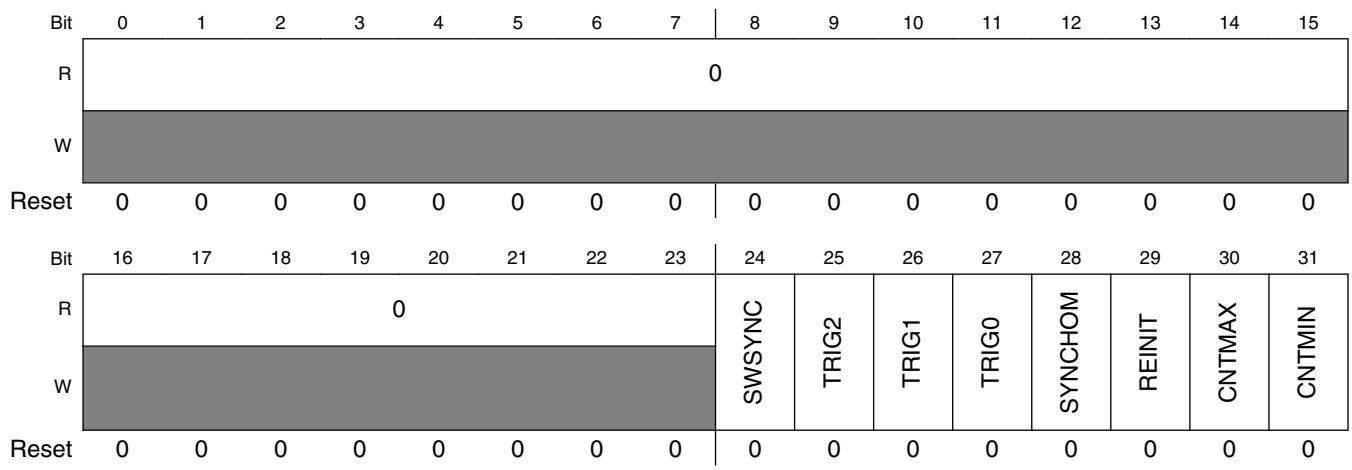
NOTE

The software trigger, SWSYNC bit, and hardware triggers TRIG0, TRIG1, and TRIG2 bits have a potential conflict if used together when SYNCMODE = 0. Use only hardware or software triggers but not both at the same time, otherwise unpredictable behavior is likely to happen.

The selection of the loading point, CNTMAX and CNTMIN bits, is intended to provide the update of MOD, CNTIN, and CnV registers across all enabled channels simultaneously. The use of the loading point selection together with SYNCMODE = 0 and hardware trigger selection, TRIG0, TRIG1, or TRIG2 bits, is likely to result in unpredictable behavior.

The synchronization event selection also depends on the PWMSYNC (MODE register) and SYNCMODE (SYNCONF register) bits. See PWM synchronization.

Address: Base address + 58h offset



FTMx_SYNC field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 SWSYNC	PWM Synchronization Software Trigger Selects the software trigger as the PWM synchronization trigger. The software trigger happens when a 1 is written to SWSYNC bit. 0 Software trigger is not selected. 1 Software trigger is selected.
25 TRIG2	PWM Synchronization Hardware Trigger 2

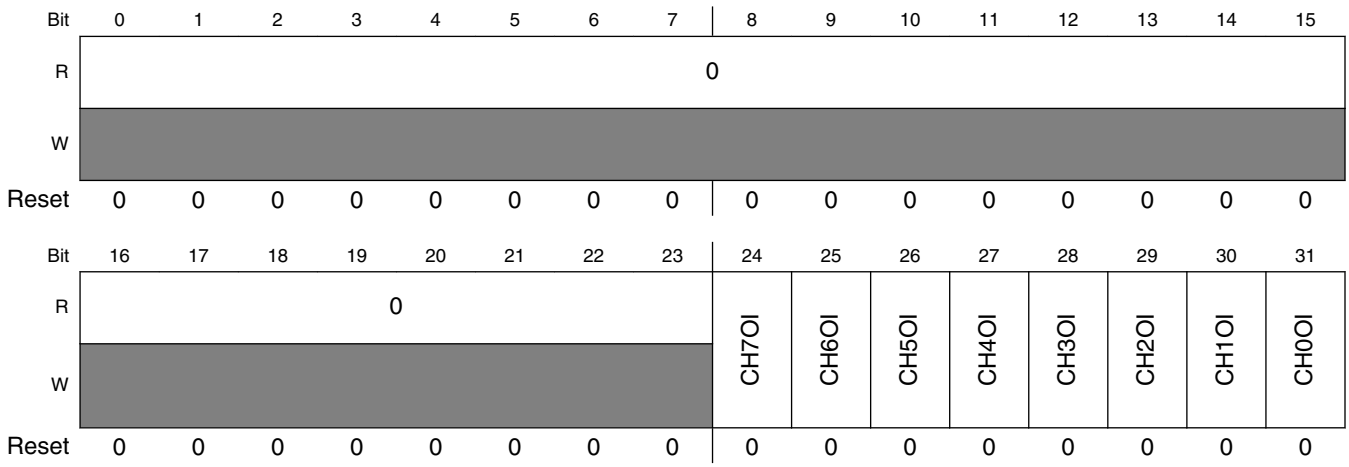
Table continues on the next page...

FTMx_SYNC field descriptions (continued)

Field	Description
	Enables hardware trigger 2 to the PWM synchronization. Hardware trigger 2 happens when a rising edge is detected at the trigger 2 input signal. 0 Trigger is disabled. 1 Trigger is enabled.
26 TRIG1	PWM Synchronization Hardware Trigger 1 Enables hardware trigger 1 to the PWM synchronization. Hardware trigger 1 happens when a rising edge is detected at the trigger 1 input signal. 0 Trigger is disabled. 1 Trigger is enabled.
27 TRIG0	PWM Synchronization Hardware Trigger 0 Enables hardware trigger 0 to the PWM synchronization. Hardware trigger 0 occurs when a rising edge is detected at the trigger 0 input signal. 0 Trigger is disabled. 1 Trigger is enabled.
28 SYNCHOM	Output Mask Synchronization Selects when the OUTMASK register is updated with the value of its buffer. 0 OUTMASK register is updated with the value of its buffer in all rising edges of the system clock. 1 OUTMASK register is updated with the value of its buffer only by the PWM synchronization.
29 REINIT	FTM Counter Reinitialization By Synchronization (FTM counter synchronization) Determines if the FTM counter is reinitialized when the selected trigger for the synchronization is detected. The REINIT bit configures the synchronization when SYNCMODE is zero. 0 FTM counter continues to count normally. 1 FTM counter is updated with its initial value when the selected trigger is detected.
30 CNTMAX	Maximum Loading Point Enable Selects the maximum loading point to PWM synchronization. See Boundary cycle and loading points. If CNTMAX is 1, the selected loading point is when the FTM counter reaches its maximum value (MOD register). 0 The maximum loading point is disabled. 1 The maximum loading point is enabled.
31 CNTMIN	Minimum Loading Point Enable Selects the minimum loading point to PWM synchronization. See Boundary cycle and loading points. If CNTMIN is one, the selected loading point is when the FTM counter reaches its minimum value (CNTIN register). 0 The minimum loading point is disabled. 1 The minimum loading point is enabled.

19.4.12 Initial State For Channels Output (FTMx_OUTINIT)

Address: Base address + 5Ch offset



FTMx_OUTINIT field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 CH7OI	Channel 7 Output Initialization Value Selects the value that is forced into the channel output when the initialization occurs. 0 The initialization value is 0. 1 The initialization value is 1.
25 CH6OI	Channel 6 Output Initialization Value Selects the value that is forced into the channel output when the initialization occurs. 0 The initialization value is 0. 1 The initialization value is 1.
26 CH5OI	Channel 5 Output Initialization Value Selects the value that is forced into the channel output when the initialization occurs. 0 The initialization value is 0. 1 The initialization value is 1.
27 CH4OI	Channel 4 Output Initialization Value Selects the value that is forced into the channel output when the initialization occurs. 0 The initialization value is 0. 1 The initialization value is 1.
28 CH3OI	Channel 3 Output Initialization Value Selects the value that is forced into the channel output when the initialization occurs.

Table continues on the next page...

FTMx_OUTINIT field descriptions (continued)

Field	Description
	0 The initialization value is 0. 1 The initialization value is 1.
29 CH2OI	Channel 2 Output Initialization Value Selects the value that is forced into the channel output when the initialization occurs. 0 The initialization value is 0. 1 The initialization value is 1.
30 CH1OI	Channel 1 Output Initialization Value Selects the value that is forced into the channel output when the initialization occurs. 0 The initialization value is 0. 1 The initialization value is 1.
31 CH0OI	Channel 0 Output Initialization Value Selects the value that is forced into the channel output when the initialization occurs. 0 The initialization value is 0. 1 The initialization value is 1.

19.4.13 Output Mask (FTMx_OUTMASK)

This register provides a mask for each FTM channel. The mask of a channel determines if its output responds, that is, it is masked or not, when a match occurs. This feature is used for BLDC control where the PWM signal is presented to an electric motor at specific times to provide electronic commutation.

Any write to the OUTMASK register, stores the value in its write buffer. The register is updated with the value of its write buffer according to PWM synchronization.

Address: Base address + 60h offset

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

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								CH7OM	CH6OM	CH5OM	CH4OM	CH3OM	CH2OM	CH1OM	CH0OM
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FTMx_OUTMASK field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 CH7OM	Channel 7 Output Mask Defines if the channel output is masked or unmasked. 0 Channel output is not masked. It continues to operate normally. 1 Channel output is masked. It is forced to its inactive state.
25 CH6OM	Channel 6 Output Mask Defines if the channel output is masked or unmasked. 0 Channel output is not masked. It continues to operate normally. 1 Channel output is masked. It is forced to its inactive state.
26 CH5OM	Channel 5 Output Mask Defines if the channel output is masked or unmasked. 0 Channel output is not masked. It continues to operate normally. 1 Channel output is masked. It is forced to its inactive state.
27 CH4OM	Channel 4 Output Mask Defines if the channel output is masked or unmasked. 0 Channel output is not masked. It continues to operate normally. 1 Channel output is masked. It is forced to its inactive state.
28 CH3OM	Channel 3 Output Mask Defines if the channel output is masked or unmasked. 0 Channel output is not masked. It continues to operate normally. 1 Channel output is masked. It is forced to its inactive state.
29 CH2OM	Channel 2 Output Mask Defines if the channel output is masked or unmasked. 0 Channel output is not masked. It continues to operate normally. 1 Channel output is masked. It is forced to its inactive state.
30 CH1OM	Channel 1 Output Mask Defines if the channel output is masked or unmasked. 0 Channel output is not masked. It continues to operate normally. 1 Channel output is masked. It is forced to its inactive state.
31 CH0OM	Channel 0 Output Mask Defines if the channel output is masked or unmasked. 0 Channel output is not masked. It continues to operate normally. 1 Channel output is masked. It is forced to its inactive state.

19.4.14 Function For Linked Channels (FTMx_COMBINE)

This register contains the control bits used to configure the fault control, synchronization, deadtime insertion, Dual Edge Capture mode, Complementary, and Combine mode for each pair of channels (n) and (n+1), where n equals 0, 2, 4, and 6.

Address: Base address + 64h offset

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	FAULTEN3	SYNCEN3	DTEN3	DECAP3	DECAPEN3	COMP3	COMBINE3	0	FAULTEN2	SYNCEN2	DTEN2	DECAP2	DECAPEN2	COMP2	COMBINE2
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	FAULTEN1	SYNCEN1	DTEN1	DECAP1	DECAPEN1	COMP1	COMBINE1	0	FAULTEN0	SYNCEN0	DTEN0	DECAP0	DECAPEN0	COMP0	COMBINE0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FTMx_COMBINE field descriptions

Field	Description
0 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 FAULTEN3	Fault Control Enable For n = 6 Enables the fault control in channels (n) and (n+1). This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The fault control in this pair of channels is disabled. 1 The fault control in this pair of channels is enabled.
2 SYNCEN3	Synchronization Enable For n = 6 Enables PWM synchronization of registers C(n)V and C(n+1)V. 0 The PWM synchronization in this pair of channels is disabled. 1 The PWM synchronization in this pair of channels is enabled.
3 DTEN3	Deadtime Enable For n = 6 Enables the deadtime insertion in the channels (n) and (n+1). This field is write protected. It can be written only when MODE[WPDIS] = 1.

Table continues on the next page...

FTMx_COMBINE field descriptions (continued)

Field	Description
	0 The deadtime insertion in this pair of channels is disabled. 1 The deadtime insertion in this pair of channels is enabled.
4 DECAP3	Dual Edge Capture Mode Captures For n = 6 Enables the capture of the FTM counter value according to the channel (n) input event and the configuration of the dual edge capture bits. This field applies only when DECAPEN = 1. DECAP bit is cleared automatically by hardware if dual edge capture – one-shot mode is selected and when the capture of channel (n+1) event is made. 0 The dual edge captures are inactive. 1 The dual edge captures are active.
5 DECAPEN3	Dual Edge Capture Mode Enable For n = 6 Enables the Dual Edge Capture mode in the channels (n) and (n+1). This bit reconfigures the function of MSnA, ELSnB:ELSnA and ELS(n+1)B:ELS(n+1)A bits in Dual Edge Capture mode according to Table 19-6. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The Dual Edge Capture mode in this pair of channels is disabled. 1 The Dual Edge Capture mode in this pair of channels is enabled.
6 COMP3	Complement Of Channel (n) for n = 6 Enables Complementary mode for the combined channels. In Complementary mode the channel (n+1) output is the inverse of the channel (n) output. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The channel (n+1) output is the same as the channel (n) output. 1 The channel (n+1) output is the complement of the channel (n) output.
7 COMBINE3	Combine Channels For n = 6 Enables the combine feature for channels (n) and (n+1). This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 Channels (n) and (n+1) are independent. 1 Channels (n) and (n+1) are combined.
8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
9 FAULTEN2	Fault Control Enable For n = 4 Enables the fault control in channels (n) and (n+1). This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The fault control in this pair of channels is disabled. 1 The fault control in this pair of channels is enabled.
10 SYNCEN2	Synchronization Enable For n = 4 Enables PWM synchronization of registers C(n)V and C(n+1)V.

Table continues on the next page...

FTMx_COMBINE field descriptions (continued)

Field	Description
	0 The PWM synchronization in this pair of channels is disabled. 1 The PWM synchronization in this pair of channels is enabled.
11 DTEN2	Deadtime Enable For n = 4 Enables the deadtime insertion in the channels (n) and (n+1). This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The deadtime insertion in this pair of channels is disabled. 1 The deadtime insertion in this pair of channels is enabled.
12 DECAP2	Dual Edge Capture Mode Captures For n = 4 Enables the capture of the FTM counter value according to the channel (n) input event and the configuration of the dual edge capture bits. This field applies only when DECAPEN = 1. DECAP bit is cleared automatically by hardware if dual edge capture – one-shot mode is selected and when the capture of channel (n+1) event is made. 0 The dual edge captures are inactive. 1 The dual edge captures are active.
13 DECAPEN2	Dual Edge Capture Mode Enable For n = 4 Enables the Dual Edge Capture mode in the channels (n) and (n+1). This bit reconfigures the function of MSnA, ELSnB:ELSnA and ELS(n+1)B:ELS(n+1)A bits in Dual Edge Capture mode according to Table 19-6. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The Dual Edge Capture mode in this pair of channels is disabled. 1 The Dual Edge Capture mode in this pair of channels is enabled.
14 COMP2	Complement Of Channel (n) For n = 4 Enables Complementary mode for the combined channels. In Complementary mode the channel (n+1) output is the inverse of the channel (n) output. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The channel (n+1) output is the same as the channel (n) output. 1 The channel (n+1) output is the complement of the channel (n) output.
15 COMBINE2	Combine Channels For n = 4 Enables the combine feature for channels (n) and (n+1). This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 Channels (n) and (n+1) are independent. 1 Channels (n) and (n+1) are combined.
16 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
17 FAULTEN1	Fault Control Enable For n = 2 Enables the fault control in channels (n) and (n+1). This field is write protected. It can be written only when MODE[WPDIS] = 1.

Table continues on the next page...

FTMx_COMBINE field descriptions (continued)

Field	Description
	0 The fault control in this pair of channels is disabled. 1 The fault control in this pair of channels is enabled.
18 SYNCEN1	Synchronization Enable For n = 2 Enables PWM synchronization of registers C(n)V and C(n+1)V. 0 The PWM synchronization in this pair of channels is disabled. 1 The PWM synchronization in this pair of channels is enabled.
19 DTEN1	Deadtime Enable For n = 2 Enables the deadtime insertion in the channels (n) and (n+1). This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The deadtime insertion in this pair of channels is disabled. 1 The deadtime insertion in this pair of channels is enabled.
20 DECAP1	Dual Edge Capture Mode Captures For n = 2 Enables the capture of the FTM counter value according to the channel (n) input event and the configuration of the dual edge capture bits. This field applies only when DECAPEN = 1. DECAP bit is cleared automatically by hardware if Dual Edge Capture – One-Shot mode is selected and when the capture of channel (n+1) event is made. 0 The dual edge captures are inactive. 1 The dual edge captures are active.
21 DECAPEN1	Dual Edge Capture Mode Enable For n = 2 Enables the Dual Edge Capture mode in the channels (n) and (n+1). This bit reconfigures the function of MSnA, ELSnB:ELSnA and ELS(n+1)B:ELS(n+1)A bits in Dual Edge Capture mode according to Table 19-6. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The Dual Edge Capture mode in this pair of channels is disabled. 1 The Dual Edge Capture mode in this pair of channels is enabled.
22 COMP1	Complement Of Channel (n) For n = 2 Enables Complementary mode for the combined channels. In Complementary mode the channel (n+1) output is the inverse of the channel (n) output. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The channel (n+1) output is the same as the channel (n) output. 1 The channel (n+1) output is the complement of the channel (n) output.
23 COMBINE1	Combine Channels For n = 2 Enables the combine feature for channels (n) and (n+1). This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 Channels (n) and (n+1) are independent. 1 Channels (n) and (n+1) are combined.

Table continues on the next page...

FTMx_COMBINE field descriptions (continued)

Field	Description
24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 FAULTEN0	Fault Control Enable For n = 0 Enables the fault control in channels (n) and (n+1). This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The fault control in this pair of channels is disabled. 1 The fault control in this pair of channels is enabled.
26 SYNCEN0	Synchronization Enable For n = 0 Enables PWM synchronization of registers C(n)V and C(n+1)V. 0 The PWM synchronization in this pair of channels is disabled. 1 The PWM synchronization in this pair of channels is enabled.
27 DTEN0	Deadtime Enable For n = 0 Enables the deadtime insertion in the channels (n) and (n+1). This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The deadtime insertion in this pair of channels is disabled. 1 The deadtime insertion in this pair of channels is enabled.
28 DECAP0	Dual Edge Capture Mode Captures For n = 0 Enables the capture of the FTM counter value according to the channel (n) input event and the configuration of the dual edge capture bits. This field applies only when DECAPEN = 1. DECAP bit is cleared automatically by hardware if dual edge capture – one-shot mode is selected and when the capture of channel (n+1) event is made. 0 The dual edge captures are inactive. 1 The dual edge captures are active.
29 DECAPEN0	Dual Edge Capture Mode Enable For n = 0 Enables the Dual Edge Capture mode in the channels (n) and (n+1). This bit reconfigures the function of MSnA, ELSnB:ELSnA and ELS(n+1)B:ELS(n+1)A bits in Dual Edge Capture mode according to Table 19-6. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The Dual Edge Capture mode in this pair of channels is disabled. 1 The Dual Edge Capture mode in this pair of channels is enabled.
30 COMP0	Complement Of Channel (n) For n = 0 Enables Complementary mode for the combined channels. In Complementary mode the channel (n+1) output is the inverse of the channel (n) output. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The channel (n+1) output is the same as the channel (n) output. 1 The channel (n+1) output is the complement of the channel (n) output.

Table continues on the next page...

FTMx_COMBINE field descriptions (continued)

Field	Description
31 COMBINE0	Combine Channels For n = 0 Enables the combine feature for channels (n) and (n+1). This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 Channels (n) and (n+1) are independent. 1 Channels (n) and (n+1) are combined.

19.4.15 Deadtime Insertion Control (FTMx_DEADTIME)

This register selects the deadtime prescaler factor and deadtime value. All FTM channels use this clock prescaler and this deadtime value for the deadtime insertion.

Address: Base address + 68h offset

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

FTMx_DEADTIME field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–25 DTPS	Deadtime Prescaler Value Selects the division factor of the system clock. This prescaled clock is used by the deadtime counter. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0x Divide the system clock by 1. 10 Divide the system clock by 4. 11 Divide the system clock by 16.
26–31 DTVAL	Deadtime Value Selects the deadtime insertion value for the deadtime counter. The deadtime counter is clocked by a scaled version of the system clock. See the description of DTPS. Deadtime insert value = (DTPS × DTVAL). DTVAL selects the number of deadtime counts inserted as follows: When DTVAL is 0, no counts are inserted. When DTVAL is 1, 1 count is inserted. When DTVAL is 2, 2 counts are inserted. This pattern continues up to a possible 63 counts. This field is write protected. It can be written only when MODE[WPDIS] = 1.

19.4.16 FTM External Trigger (FTMx_EXTTRIG)

This register:

- Indicates when a channel trigger was generated
- Enables the generation of a trigger when the FTM counter is equal to its initial value
- Selects which channels are used in the generation of the channel triggers

Several channels can be selected to generate multiple triggers in one PWM period. See Channel trigger output and Initialization trigger.

Channels 6 and 7 are not used to generate channel triggers.

Address: Base address + 6Ch offset

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

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved								TRIGF	INITTRIGEN	CH1TRIG	CH0TRIG	CH5TRIG	CH4TRIG	CH3TRIG	CH2TRIG
W									0							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FTMx_EXTTRIG field descriptions

Field	Description
0–23 Reserved	This field is reserved.
24 TRIGF	Channel Trigger Flag Set by hardware when a channel trigger is generated. Clear TRIGF by reading EXTTRIG while TRIGF is set and then writing a 0 to TRIGF. Writing a 1 to TRIGF has no effect. If another channel trigger is generated before the clearing sequence is completed, the sequence is reset so TRIGF remains set after the clear sequence is completed for the earlier TRIGF. 0 No channel trigger was generated. 1 A channel trigger was generated.
25 INITTRIGEN	Initialization Trigger Enable Enables the generation of the trigger when the FTM counter is equal to the CNTIN register. 0 The generation of initialization trigger is disabled. 1 The generation of initialization trigger is enabled.
26 CH1TRIG	Channel 1 Trigger Enable Enables the generation of the channel trigger when the FTM counter is equal to the CnV register. 0 The generation of the channel trigger is disabled. 1 The generation of the channel trigger is enabled.
27 CH0TRIG	Channel 0 Trigger Enable Enables the generation of the channel trigger when the FTM counter is equal to the CnV register. 0 The generation of the channel trigger is disabled. 1 The generation of the channel trigger is enabled.
28 CH5TRIG	Channel 5 Trigger Enable Enables the generation of the channel trigger when the FTM counter is equal to the CnV register. 0 The generation of the channel trigger is disabled. 1 The generation of the channel trigger is enabled.
29 CH4TRIG	Channel 4 Trigger Enable Enables the generation of the channel trigger when the FTM counter is equal to the CnV register. 0 The generation of the channel trigger is disabled. 1 The generation of the channel trigger is enabled.
30 CH3TRIG	Channel 3 Trigger Enable Enables the generation of the channel trigger when the FTM counter is equal to the CnV register. 0 The generation of the channel trigger is disabled. 1 The generation of the channel trigger is enabled.
31 CH2TRIG	Channel 2 Trigger Enable Enables the generation of the channel trigger when the FTM counter is equal to the CnV register.

Table continues on the next page...

FTMx_EXTTRIG field descriptions (continued)

Field	Description
	0 The generation of the channel trigger is disabled. 1 The generation of the channel trigger is enabled.

19.4.17 Channels Polarity (FTMx_POL)

This register defines the output polarity of the FTM channels.

NOTE

The safe value that is driven in a channel output when the fault control is enabled and a fault condition is detected is the inactive state of the channel. That is, the safe value of a channel is the value of its POL bit.

Address: Base address + 70h offset

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

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Reserved								POL7	POL6	POL5	POL4	POL3	POL2	POL1	POL0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FTMx_POL field descriptions

Field	Description
0–23 Reserved	This field is reserved.
24 POL7	Channel 7 Polarity Defines the polarity of the channel output. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The channel polarity is active high. 1 The channel polarity is active low.
25 POL6	Channel 6 Polarity Defines the polarity of the channel output. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The channel polarity is active high. 1 The channel polarity is active low.
26 POL5	Channel 5 Polarity Defines the polarity of the channel output. This field is write protected. It can be written only when MODE[WPDIS] = 1.

Table continues on the next page...

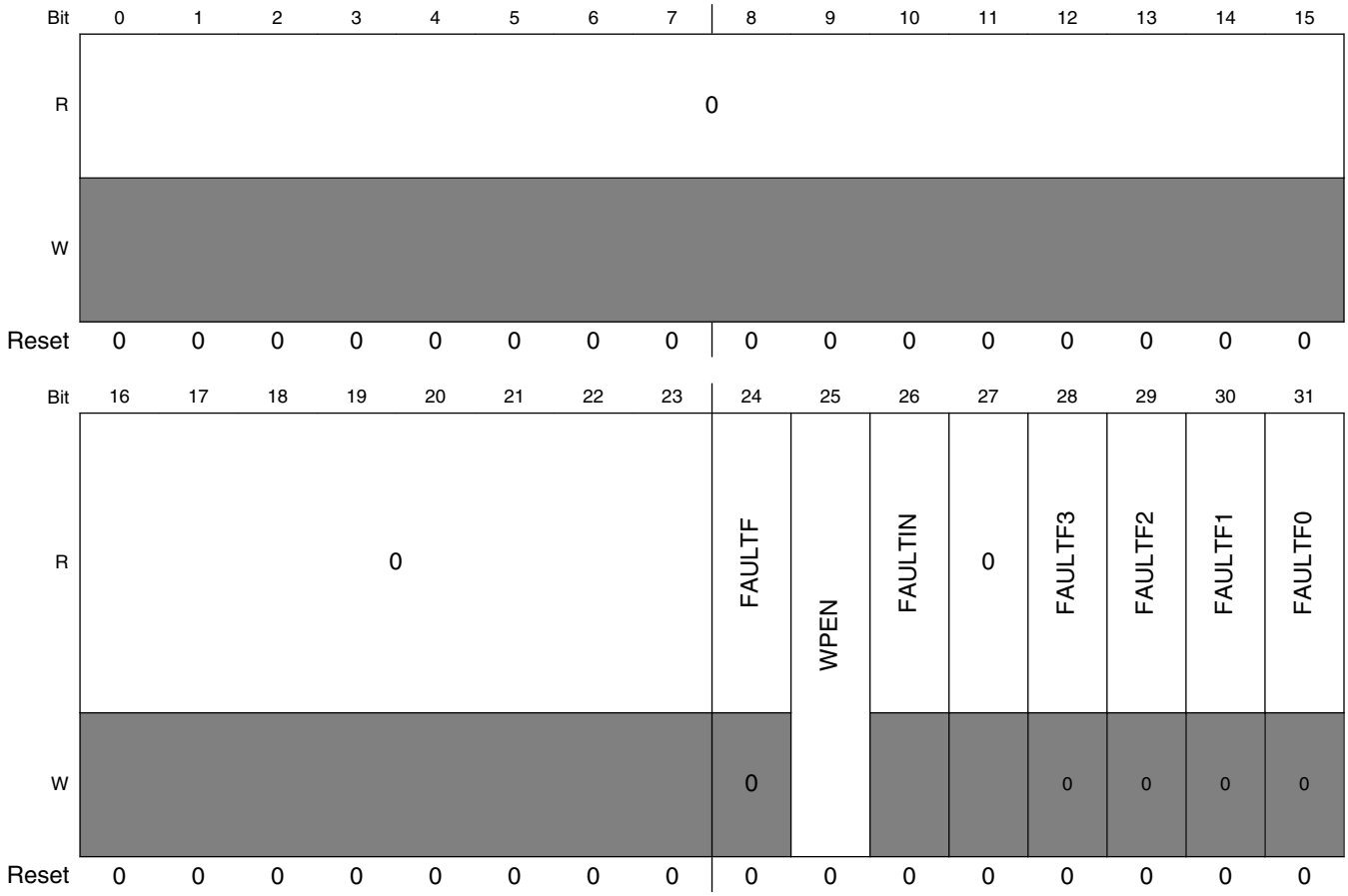
FTMx_POL field descriptions (continued)

Field	Description
	0 The channel polarity is active high. 1 The channel polarity is active low.
27 POL4	Channel 4 Polarity Defines the polarity of the channel output. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The channel polarity is active high. 1 The channel polarity is active low.
28 POL3	Channel 3 Polarity Defines the polarity of the channel output. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The channel polarity is active high. 1 The channel polarity is active low.
29 POL2	Channel 2 Polarity Defines the polarity of the channel output. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The channel polarity is active high. 1 The channel polarity is active low.
30 POL1	Channel 1 Polarity Defines the polarity of the channel output. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The channel polarity is active high. 1 The channel polarity is active low.
31 POL0	Channel 0 Polarity Defines the polarity of the channel output. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The channel polarity is active high. 1 The channel polarity is active low.

19.4.18 Fault Mode Status (FTMx_FMS)

This register contains the fault detection flags, write protection enable bit, and the logic OR of the enabled fault inputs.

Address: Base address + 74h offset



FTMx_FMS field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 FAULTF	Fault Detection Flag Represents the logic OR of the individual FAULTFj bits where j = 3, 2, 1, 0. Clear FAULTF by reading the FMS register while FAULTF is set and then writing a 0 to FAULTF while there is no existing fault condition at the enabled fault inputs. Writing a 1 to FAULTF has no effect. If another fault condition is detected in an enabled fault input before the clearing sequence is completed, the sequence is reset so FAULTF remains set after the clearing sequence is completed for the earlier fault condition. FAULTF is also cleared when FAULTFj bits are cleared individually. 0 No fault condition was detected. 1 A fault condition was detected.

Table continues on the next page...

FTMx_FMS field descriptions (continued)

Field	Description
25 WPEN	Write Protection Enable The WPEN bit is the negation of the WPDIS bit. WPEN is set when 1 is written to it. WPEN is cleared when WPEN bit is read as a 1 and then 1 is written to WPDIS. Writing 0 to WPEN has no effect. 0 Write protection is disabled. Write protected bits can be written. 1 Write protection is enabled. Write protected bits cannot be written.
26 FAULTIN	Fault Inputs Represents the logic OR of the enabled fault inputs after their filter (if their filter is enabled) when fault control is enabled. 0 The logic OR of the enabled fault inputs is 0. 1 The logic OR of the enabled fault inputs is 1.
27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 FAULTF3	Fault Detection Flag 3 Set by hardware when fault control is enabled, the corresponding fault input is enabled and a fault condition is detected at the fault input. Clear FAULTF3 by reading the FMS register while FAULTF3 is set and then writing a 0 to FAULTF3 while there is no existing fault condition at the corresponding fault input. Writing a 1 to FAULTF3 has no effect. FAULTF3 bit is also cleared when FAULTF bit is cleared. If another fault condition is detected at the corresponding fault input before the clearing sequence is completed, the sequence is reset so FAULTF3 remains set after the clearing sequence is completed for the earlier fault condition. 0 No fault condition was detected at the fault input. 1 A fault condition was detected at the fault input.
29 FAULTF2	Fault Detection Flag 2 Set by hardware when fault control is enabled, the corresponding fault input is enabled and a fault condition is detected at the fault input. Clear FAULTF2 by reading the FMS register while FAULTF2 is set and then writing a 0 to FAULTF2 while there is no existing fault condition at the corresponding fault input. Writing a 1 to FAULTF2 has no effect. FAULTF2 bit is also cleared when FAULTF bit is cleared. If another fault condition is detected at the corresponding fault input before the clearing sequence is completed, the sequence is reset so FAULTF2 remains set after the clearing sequence is completed for the earlier fault condition. 0 No fault condition was detected at the fault input. 1 A fault condition was detected at the fault input.
30 FAULTF1	Fault Detection Flag 1 Set by hardware when fault control is enabled, the corresponding fault input is enabled and a fault condition is detected at the fault input. Clear FAULTF1 by reading the FMS register while FAULTF1 is set and then writing a 0 to FAULTF1 while there is no existing fault condition at the corresponding fault input. Writing a 1 to FAULTF1 has no effect. FAULTF1 bit is also cleared when FAULTF bit is cleared.

Table continues on the next page...

FTMx_FMS field descriptions (continued)

Field	Description
	If another fault condition is detected at the corresponding fault input before the clearing sequence is completed, the sequence is reset so FAULTF1 remains set after the clearing sequence is completed for the earlier fault condition. 0 No fault condition was detected at the fault input. 1 A fault condition was detected at the fault input.
31 FAULTF0	Fault Detection Flag 0 Set by hardware when fault control is enabled, the corresponding fault input is enabled and a fault condition is detected at the fault input. Clear FAULTF0 by reading the FMS register while FAULTF0 is set and then writing a 0 to FAULTF0 while there is no existing fault condition at the corresponding fault input. Writing a 1 to FAULTF0 has no effect. FAULTF0 bit is also cleared when FAULTF bit is cleared. If another fault condition is detected at the corresponding fault input before the clearing sequence is completed, the sequence is reset so FAULTF0 remains set after the clearing sequence is completed for the earlier fault condition. 0 No fault condition was detected at the fault input. 1 A fault condition was detected at the fault input.

19.4.19 Input Capture Filter Control (FTMx_FILTER)

This register selects the filter value for the inputs of channels.

Channels 4, 5, 6 and 7 do not have an input filter.

NOTE

Writing to the FILTER register has immediate effect and must be done only when the channels 0, 1, 2, and 3 are not in input modes. Failure to do this could result in a missing valid signal.

Address: Base address + 78h offset

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Reserved																CH3FVAL				CH2FVAL				CH1FVAL				CH0FVAL			
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

FTMx_FILTER field descriptions

Field	Description
0–15 Reserved	This field is reserved.
16–19 CH3FVAL	Channel 3 Input Filter Selects the filter value for the channel input. The filter is disabled when the value is zero.

Table continues on the next page...

FTMx_FILTER field descriptions (continued)

Field	Description
20–23 CH2FVAL	Channel 2 Input Filter Selects the filter value for the channel input. The filter is disabled when the value is zero.
24–27 CH1FVAL	Channel 1 Input Filter Selects the filter value for the channel input. The filter is disabled when the value is zero.
28–31 CH0FVAL	Channel 0 Input Filter Selects the filter value for the channel input. The filter is disabled when the value is zero.

19.4.20 Fault Control (FTMx_FLTCTRL)

This register selects the filter value for the fault inputs, enables the fault inputs and the fault inputs filter.

Address: Base address + 7Ch offset

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

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0				FFVAL				FFLTR3EN	FFLTR2EN	FFLTR1EN	FFLTR0EN	FAULT3EN	FAULT2EN	FAULT1EN	FAULT0EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FTMx_FLTCTRL field descriptions

Field	Description
0–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–23 FFVAL	Fault Input Filter Selects the filter value for the fault inputs. The fault filter is disabled when the value is zero.

Table continues on the next page...

FTMx_FLTCTRL field descriptions (continued)

Field	Description
	NOTE: Writing to this field has immediate effect and must be done only when the fault control or all fault inputs are disabled. Failure to do this could result in a missing fault detection.
24 FFLTR3EN	Fault Input 3 Filter Enable Enables the filter for the fault input. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 Fault input filter is disabled. 1 Fault input filter is enabled.
25 FFLTR2EN	Fault Input 2 Filter Enable Enables the filter for the fault input. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 Fault input filter is disabled. 1 Fault input filter is enabled.
26 FFLTR1EN	Fault Input 1 Filter Enable Enables the filter for the fault input. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 Fault input filter is disabled. 1 Fault input filter is enabled.
27 FFLTR0EN	Fault Input 0 Filter Enable Enables the filter for the fault input. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 Fault input filter is disabled. 1 Fault input filter is enabled.
28 FAULT3EN	Fault Input 3 Enable Enables the fault input. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 Fault input is disabled. 1 Fault input is enabled.
29 FAULT2EN	Fault Input 2 Enable Enables the fault input. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 Fault input is disabled. 1 Fault input is enabled.
30 FAULT1EN	Fault Input 1 Enable Enables the fault input. This field is write protected. It can be written only when MODE[WPDIS] = 1.

Table continues on the next page...

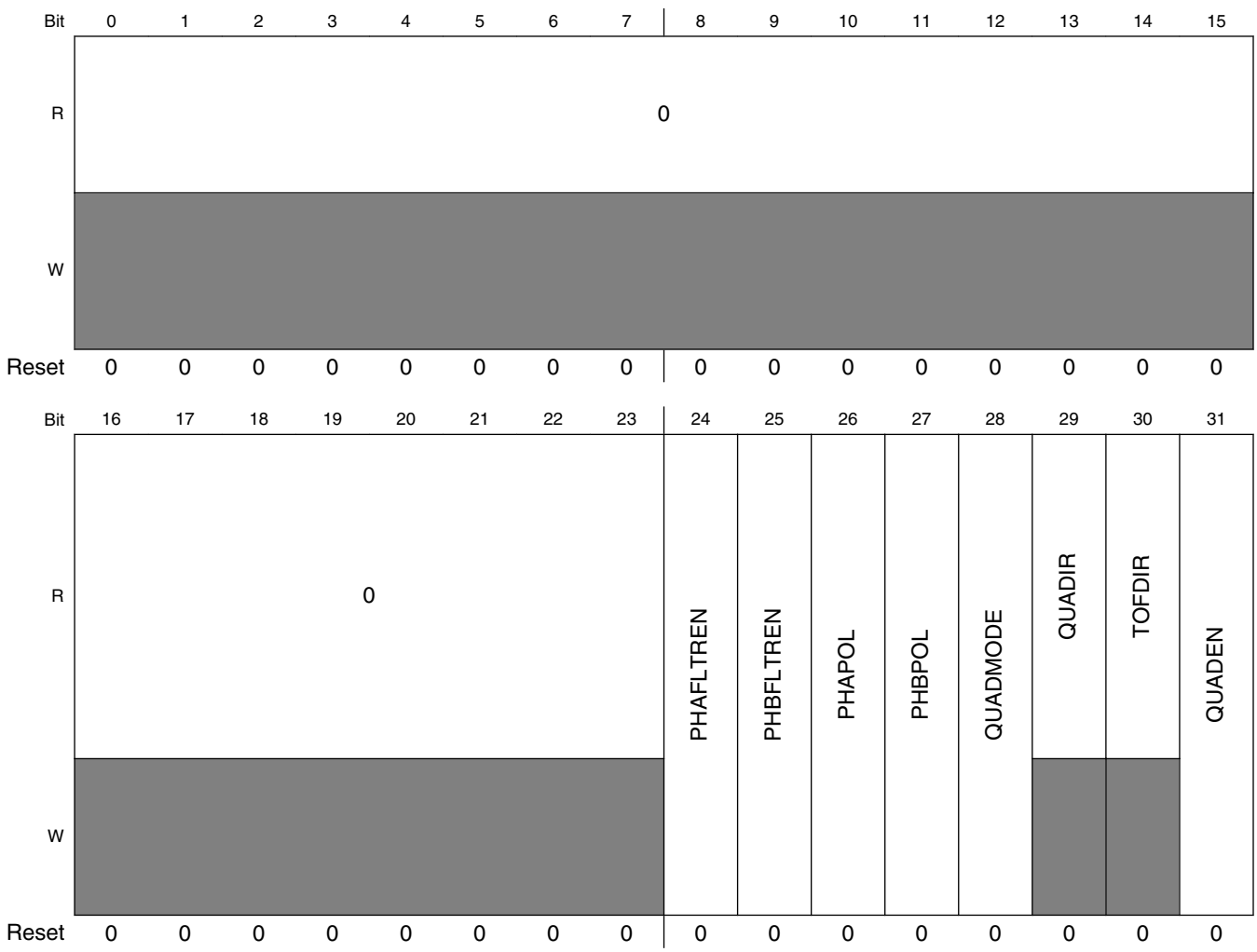
FTMx_FLTCTRL field descriptions (continued)

Field	Description
	0 Fault input is disabled. 1 Fault input is enabled.
31 FAULT0EN	Fault Input 0 Enable Enables the fault input. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 Fault input is disabled. 1 Fault input is enabled.

19.4.21 Quadrature Decoder Control And Status (FTMx_QDCTRL)

This register has the control and status bits for the Quadrature Decoder mode.

Address: Base address + 80h offset



FTMx_QDCTRL field descriptions

Field	Description
0–23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 PHAFLTREN	Phase A Input Filter Enable Enables the filter for the quadrature decoder phase A input. The filter value for the phase A input is defined by the CH0FVAL field of FILTER. The phase A filter is also disabled when CH0FVAL is zero. 0 Phase A input filter is disabled. 1 Phase A input filter is enabled.
25 PHBFLTREN	Phase B Input Filter Enable Enables the filter for the quadrature decoder phase B input. The filter value for the phase B input is defined by the CH1FVAL field of FILTER. The phase B filter is also disabled when CH1FVAL is zero. 0 Phase B input filter is disabled. 1 Phase B input filter is enabled.
26 PHAPOL	Phase A Input Polarity Selects the polarity for the quadrature decoder phase A input. 0 Normal polarity. Phase A input signal is not inverted before identifying the rising and falling edges of this signal. 1 Inverted polarity. Phase A input signal is inverted before identifying the rising and falling edges of this signal.
27 PHBPOL	Phase B Input Polarity Selects the polarity for the quadrature decoder phase B input. 0 Normal polarity. Phase B input signal is not inverted before identifying the rising and falling edges of this signal. 1 Inverted polarity. Phase B input signal is inverted before identifying the rising and falling edges of this signal.
28 QUADMODE	Quadrature Decoder Mode Selects the encoding mode used in the Quadrature Decoder mode. 0 Phase A and phase B encoding mode. 1 Count and direction encoding mode.
29 QUADIR	FTM Counter Direction In Quadrature Decoder Mode Indicates the counting direction. 0 Counting direction is decreasing (FTM counter decrement). 1 Counting direction is increasing (FTM counter increment).
30 TOFDIR	Timer Overflow Direction In Quadrature Decoder Mode Indicates if the TOF bit was set on the top or the bottom of counting. 0 TOF bit was set on the bottom of counting. There was an FTM counter decrement and FTM counter changes from its minimum value (CNTIN register) to its maximum value (MOD register). 1 TOF bit was set on the top of counting. There was an FTM counter increment and FTM counter changes from its maximum value (MOD register) to its minimum value (CNTIN register).

Table continues on the next page...

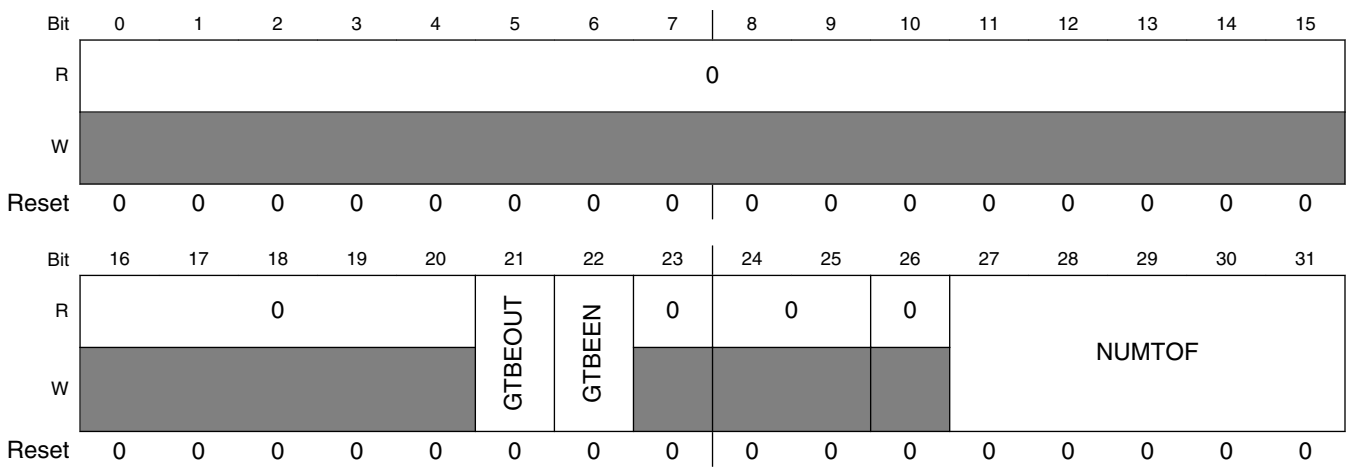
FTMx_QDCTRL field descriptions (continued)

Field	Description
31 QUADEN	Quadrature Decoder Mode Enable Enables the Quadrature Decoder mode. In this mode, the phase A and B input signals control the FTM counter direction. The Quadrature Decoder mode has precedence over the other modes. See Table 19-6. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 Quadrature Decoder mode is disabled. 1 Quadrature Decoder mode is enabled.

19.4.22 Configuration (FTMx_CONF)

This register selects the number of times that the FTM counter overflow should occur before the TOF bit to be set, the FTM behavior in modes, the use of an external global time base, and the global time base signal generation.

Address: Base address + 84h offset



FTMx_CONF field descriptions

Field	Description
0–20 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21 GTBEOUT	Global Time Base Output Enables the global time base signal generation to other FTMs. 0 A global time base signal generation is disabled. 1 A global time base signal generation is enabled.
22 GTBEEN	Global Time Base Enable

Table continues on the next page...

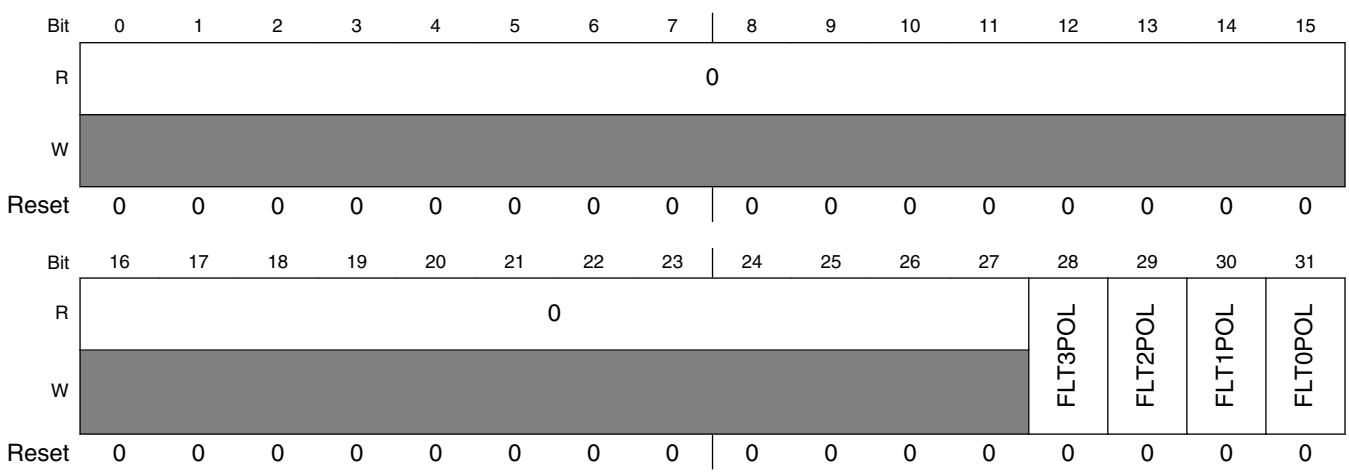
FTMx_CONF field descriptions (continued)

Field	Description
	Configures the FTM to use an external global time base signal that is generated by another FTM. 0 Use of an external global time base is disabled. 1 Use of an external global time base is enabled.
23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24–25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–31 NUMTOF	TOF Frequency Selects the ratio between the number of counter overflows to the number of times the TOF bit is set. NUMTOF = 0: The TOF bit is set for each counter overflow. NUMTOF = 1: The TOF bit is set for the first counter overflow but not for the next overflow. NUMTOF = 2: The TOF bit is set for the first counter overflow but not for the next 2 overflows. NUMTOF = 3: The TOF bit is set for the first counter overflow but not for the next 3 overflows. This pattern continues up to a maximum of 31.

19.4.23 FTM Fault Input Polarity (FTMx_FLTPOL)

This register defines the fault inputs polarity.

Address: Base address + 88h offset



FTMx_FLTPOL field descriptions

Field	Description
0–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

FTMx_FLTPOL field descriptions (continued)

Field	Description
28 FLT3POL	Fault Input 3 Polarity Defines the polarity of the fault input. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The fault input polarity is active high. A 1 at the fault input indicates a fault. 1 The fault input polarity is active low. A 0 at the fault input indicates a fault.
29 FLT2POL	Fault Input 2 Polarity Defines the polarity of the fault input. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The fault input polarity is active high. A 1 at the fault input indicates a fault. 1 The fault input polarity is active low. A 0 at the fault input indicates a fault.
30 FLT1POL	Fault Input 1 Polarity Defines the polarity of the fault input. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The fault input polarity is active high. A 1 at the fault input indicates a fault. 1 The fault input polarity is active low. A 0 at the fault input indicates a fault.
31 FLT0POL	Fault Input 0 Polarity Defines the polarity of the fault input. This field is write protected. It can be written only when MODE[WPDIS] = 1. 0 The fault input polarity is active high. A 1 at the fault input indicates a fault. 1 The fault input polarity is active low. A 0 at the fault input indicates a fault.

19.4.24 Synchronization Configuration (FTMx_SYNCONF)

This register selects the PWM synchronization configuration, SWOCTRL, INVCTRL and CNTIN registers synchronization, if FTM clears the TRIGj bit, where j = 0, 1, 2, when the hardware trigger j is detected.

Address: Base address + 8Ch offset

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0											HWSOC	HWINVC	HWOM	HWWRBUF	HWRSTCNT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0			SWSOC	SWINVC	SWOM	SWWRBUF	SWRSTCNT	SYNCMODE	0	SWOC	INVC	0	CNTINC	0	HWTRIGMODE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FTMx_SYNCONF field descriptions

Field	Description
0–10 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
11 HWSOC	Software output control synchronization is activated by a hardware trigger. 0 A hardware trigger does not activate the SWOCTRL register synchronization. 1 A hardware trigger activates the SWOCTRL register synchronization.
12 HWINVC	Inverting control synchronization is activated by a hardware trigger. 0 A hardware trigger does not activate the INVCTRL register synchronization. 1 A hardware trigger activates the INVCTRL register synchronization.
13 HWOM	Output mask synchronization is activated by a hardware trigger. 0 A hardware trigger does not activate the OUTMASK register synchronization. 1 A hardware trigger activates the OUTMASK register synchronization.
14 HWWRBUF	MOD, CNTIN, and CV registers synchronization is activated by a hardware trigger. 0 A hardware trigger does not activate MOD, CNTIN, and CV registers synchronization. 1 A hardware trigger activates MOD, CNTIN, and CV registers synchronization.
15 HWRSTCNT	FTM counter synchronization is activated by a hardware trigger. 0 A hardware trigger does not activate the FTM counter synchronization. 1 A hardware trigger activates the FTM counter synchronization.

Table continues on the next page...

FTMx_SYNCONF field descriptions (continued)

Field	Description
16–18 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
19 SWSOC	Software output control synchronization is activated by the software trigger. 0 The software trigger does not activate the SWOCTRL register synchronization. 1 The software trigger activates the SWOCTRL register synchronization.
20 SWINVC	Inverting control synchronization is activated by the software trigger. 0 The software trigger does not activate the INVCTRL register synchronization. 1 The software trigger activates the INVCTRL register synchronization.
21 SWOM	Output mask synchronization is activated by the software trigger. 0 The software trigger does not activate the OUTMASK register synchronization. 1 The software trigger activates the OUTMASK register synchronization.
22 SWWRBUF	MOD, CNTIN, and CV registers synchronization is activated by the software trigger. 0 The software trigger does not activate MOD, CNTIN, and CV registers synchronization. 1 The software trigger activates MOD, CNTIN, and CV registers synchronization.
23 SWRSTCNT	FTM counter synchronization is activated by the software trigger. 0 The software trigger does not activate the FTM counter synchronization. 1 The software trigger activates the FTM counter synchronization.
24 SYNCMODE	Synchronization Mode Selects the PWM Synchronization mode. 0 Legacy PWM synchronization is selected. 1 Enhanced PWM synchronization is selected.
25 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
26 SWOC	SWOCTRL Register Synchronization 0 SWOCTRL register is updated with its buffer value at all rising edges of system clock. 1 SWOCTRL register is updated with its buffer value by the PWM synchronization.
27 INVC	INVCTRL Register Synchronization 0 INVCTRL register is updated with its buffer value at all rising edges of system clock. 1 INVCTRL register is updated with its buffer value by the PWM synchronization.
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 CNTINC	CNTIN Register Synchronization 0 CNTIN register is updated with its buffer value at all rising edges of system clock. 1 CNTIN register is updated with its buffer value by the PWM synchronization.
30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
31 HWTRIGMODE	Hardware Trigger Mode

Table continues on the next page...

FTMx_SYNCONF field descriptions (continued)

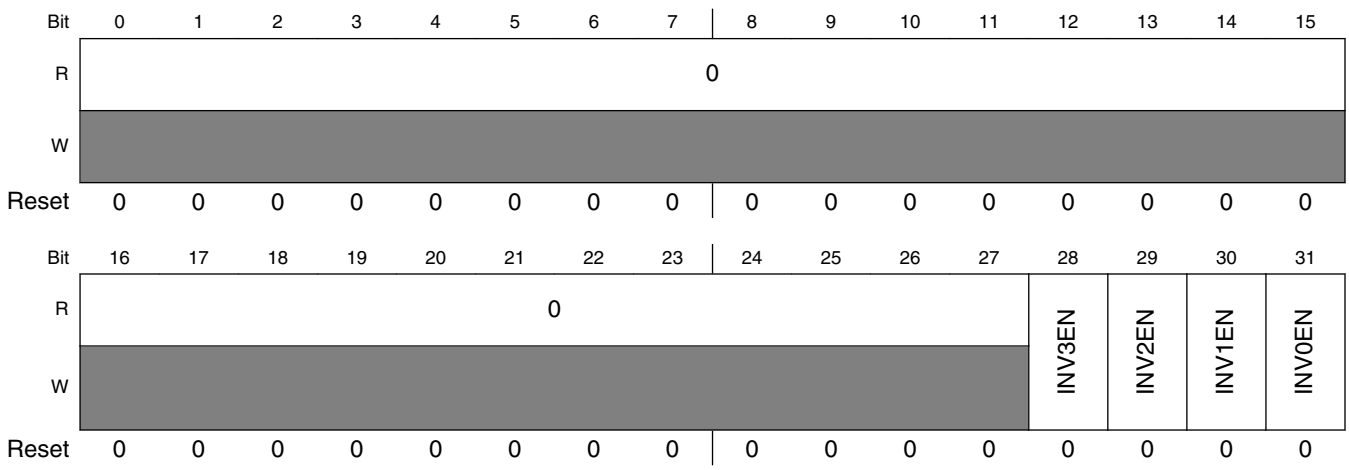
Field	Description
	0 FTM clears the TRIGj bit when the hardware trigger j is detected, where j = 0, 1,2. 1 FTM does not clear the TRIGj bit when the hardware trigger j is detected, where j = 0, 1,2.

19.4.25 FTM Inverting Control (FTMx_INVCTRL)

This register controls when the channel (n) output becomes the channel (n+1) output, and channel (n+1) output becomes the channel (n) output. Each INVmEN bit enables the inverting operation for the corresponding pair channels m.

This register has a write buffer. The INVmEN bit is updated by the INVCTRL register synchronization.

Address: Base address + 90h offset



FTMx_INVCTRL field descriptions

Field	Description
0–27 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 INV3EN	Pair Channels 3 Inverting Enable 0 Inverting is disabled. 1 Inverting is enabled.
29 INV2EN	Pair Channels 2 Inverting Enable 0 Inverting is disabled. 1 Inverting is enabled.
30 INV1EN	Pair Channels 1 Inverting Enable 0 Inverting is disabled. 1 Inverting is enabled.

Table continues on the next page...

FTMx_INVCTRL field descriptions (continued)

Field	Description
31 INV0EN	Pair Channels 0 Inverting Enable 0 Inverting is disabled. 1 Inverting is enabled.

19.4.26 FTM Software Output Control (FTMx_SWOCTRL)

This register enables software control of channel (n) output and defines the value forced to the channel (n) output:

- The CHnOC bits enable the control of the corresponding channel (n) output by software.
- The CHnOCV bits select the value that is forced at the corresponding channel (n) output.

This register has a write buffer. The fields are updated by the SWOCTRL register synchronization.

Address: Base address + 94h offset

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

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CH7OCV	CH6OCV	CH5OCV	CH4OCV	CH3OCV	CH2OCV	CH1OCV	CH0OCV	CH7OC	CH6OC	CH5OC	CH4OC	CH3OC	CH2OC	CH1OC	CH0OC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FTMx_SWOCTRL field descriptions

Field	Description
0–15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 CH7OCV	Channel 7 Software Output Control Value 0 The software output control forces 0 to the channel output. 1 The software output control forces 1 to the channel output.
17 CH6OCV	Channel 6 Software Output Control Value

Table continues on the next page...

FTMx_SWOCTRL field descriptions (continued)

Field	Description
	0 The software output control forces 0 to the channel output. 1 The software output control forces 1 to the channel output.
18 CH5OCV	Channel 5 Software Output Control Value 0 The software output control forces 0 to the channel output. 1 The software output control forces 1 to the channel output.
19 CH4OCV	Channel 4 Software Output Control Value 0 The software output control forces 0 to the channel output. 1 The software output control forces 1 to the channel output.
20 CH3OCV	Channel 3 Software Output Control Value 0 The software output control forces 0 to the channel output. 1 The software output control forces 1 to the channel output.
21 CH2OCV	Channel 2 Software Output Control Value 0 The software output control forces 0 to the channel output. 1 The software output control forces 1 to the channel output.
22 CH1OCV	Channel 1 Software Output Control Value 0 The software output control forces 0 to the channel output. 1 The software output control forces 1 to the channel output.
23 CH0OCV	Channel 0 Software Output Control Value 0 The software output control forces 0 to the channel output. 1 The software output control forces 1 to the channel output.
24 CH7OC	Channel 7 Software Output Control Enable 0 The channel output is not affected by software output control. 1 The channel output is affected by software output control.
25 CH6OC	Channel 6 Software Output Control Enable 0 The channel output is not affected by software output control. 1 The channel output is affected by software output control.
26 CH5OC	Channel 5 Software Output Control Enable 0 The channel output is not affected by software output control. 1 The channel output is affected by software output control.
27 CH4OC	Channel 4 Software Output Control Enable 0 The channel output is not affected by software output control. 1 The channel output is affected by software output control.
28 CH3OC	Channel 3 Software Output Control Enable 0 The channel output is not affected by software output control. 1 The channel output is affected by software output control.
29 CH2OC	Channel 2 Software Output Control Enable

Table continues on the next page...

FTMx_SWOCTRL field descriptions (continued)

Field	Description
	0 The channel output is not affected by software output control. 1 The channel output is affected by software output control.
30 CH1OC	Channel 1 Software Output Control Enable 0 The channel output is not affected by software output control. 1 The channel output is affected by software output control.
31 CH0OC	Channel 0 Software Output Control Enable 0 The channel output is not affected by software output control. 1 The channel output is affected by software output control.

19.4.27 FTM PWM Load (FTMx_PWMLOAD)

Enables the loading of the MOD, CNTIN, C(n)V, and C(n+1)V registers with the values of their write buffers when the FTM counter changes from the MOD register value to its next value or when a channel (j) match occurs. A match occurs for the channel (j) when FTM counter = C(j)V.

Address: Base address + 98h offset

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

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0						LDOK	0	CH7SEL	CH6SEL	CH5SEL	CH4SEL	CH3SEL	CH2SEL	CH1SEL	CH0SEL
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FTMx_PWMLOAD field descriptions

Field	Description
0–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
22 LDOK	Load Enable Enables the loading of the MOD, CNTIN, and CV registers with the values of their write buffers. 0 Loading updated values is disabled. 1 Loading updated values is enabled.

Table continues on the next page...

FTMx_PWMLOAD field descriptions (continued)

Field	Description
23 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
24 CH7SEL	Channel 7 Select 0 Do not include the channel in the matching process. 1 Include the channel in the matching process.
25 CH6SEL	Channel 6 Select 0 Do not include the channel in the matching process. 1 Include the channel in the matching process.
26 CH5SEL	Channel 5 Select 0 Do not include the channel in the matching process. 1 Include the channel in the matching process.
27 CH4SEL	Channel 4 Select 0 Do not include the channel in the matching process. 1 Include the channel in the matching process.
28 CH3SEL	Channel 3 Select 0 Do not include the channel in the matching process. 1 Include the channel in the matching process.
29 CH2SEL	Channel 2 Select 0 Do not include the channel in the matching process. 1 Include the channel in the matching process.
30 CH1SEL	Channel 1 Select 0 Do not include the channel in the matching process. 1 Include the channel in the matching process.
31 CH0SEL	Channel 0 Select 0 Do not include the channel in the matching process. 1 Include the channel in the matching process.

19.5 Functional description

The notation used in this document to represent the counters and the generation of the signals is shown in the following figure.

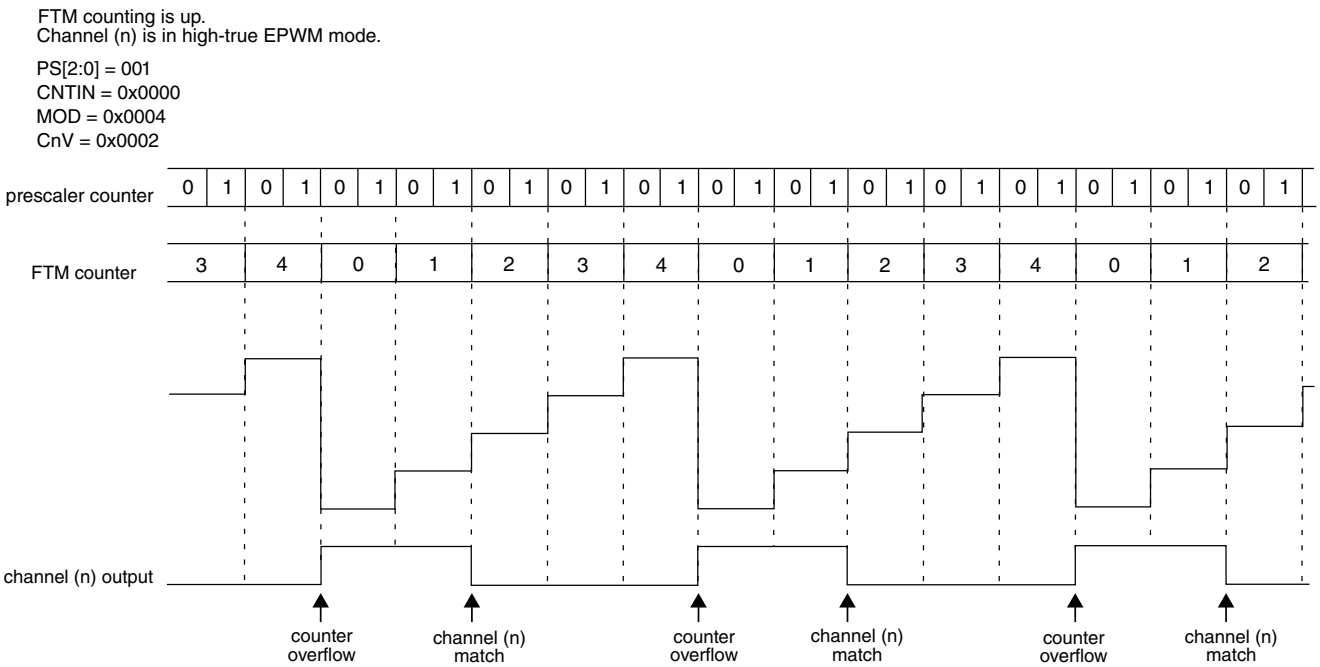


Figure 19-2. Notation used

19.5.1 Clock source

The FTM has only one clock domain: the system clock.

19.5.1.1 Counter clock source

The CLKS[1:0] bits in the SC register selects clock sources for the FTM counter or disables the FTM counter. After any chip reset, CLKS[1:0] = 0:0 so no clock source is selected.

The CLKS[1:0] bits may be read or written at any time. Disabling the FTM counter by writing 0:0 to the CLKS[1:0] bits does not affect the FTM counter value or other registers.

The fixed frequency clock is an alternative clock source for the FTM counter that allows the selection of a clock other than the system clock or an external clock. This clock input is defined by chip integration; see the chip-specific FTM information for further details. Due to FTM hardware implementation limitations, the frequency of the fixed frequency clock must not exceed 1/2 of the system clock frequency.

The external clock passes through a synchronizer clocked by the system clock to assure that counter transitions are properly aligned to system clock transitions.Therefore, to meet Nyquist criteria considering also jitter, the frequency of the external clock source must not exceed 1/4 of the system clock frequency.

19.5.2 Prescaler

The selected counter clock source passes through a prescaler that is a 7-bit counter. The value of the prescaler is selected by the PS[2:0] bits. The following figure shows an example of the prescaler counter and FTM counter.

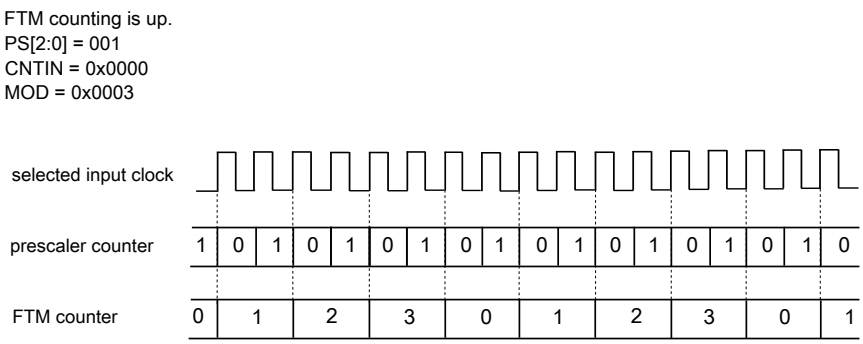


Figure 19-3. Example of the prescaler counter

19.5.3 Counter

The FTM has a 16-bit counter that is used by the channels either for input or output modes. The FTM counter clock is the selected clock divided by the prescaler.

The FTM counter has these modes of operation:

- Up counting
- Up-down counting
- Quadrature Decoder mode

19.5.3.1 Up counting

Up counting is selected when:

- QUADEN = 0, and
- CPWMS = 0

CNTIN defines the starting value of the count and MOD defines the final value of the count, see the following figure. The value of CNTIN is loaded into the FTM counter, and the counter increments until the value of MOD is reached, at which point the counter is reloaded with the value of CNTIN.

The FTM period when using up counting is (MOD – CNTIN + 0x0001) × period of the FTM counter clock.

The TOF bit is set when the FTM counter changes from MOD to CNTIN.

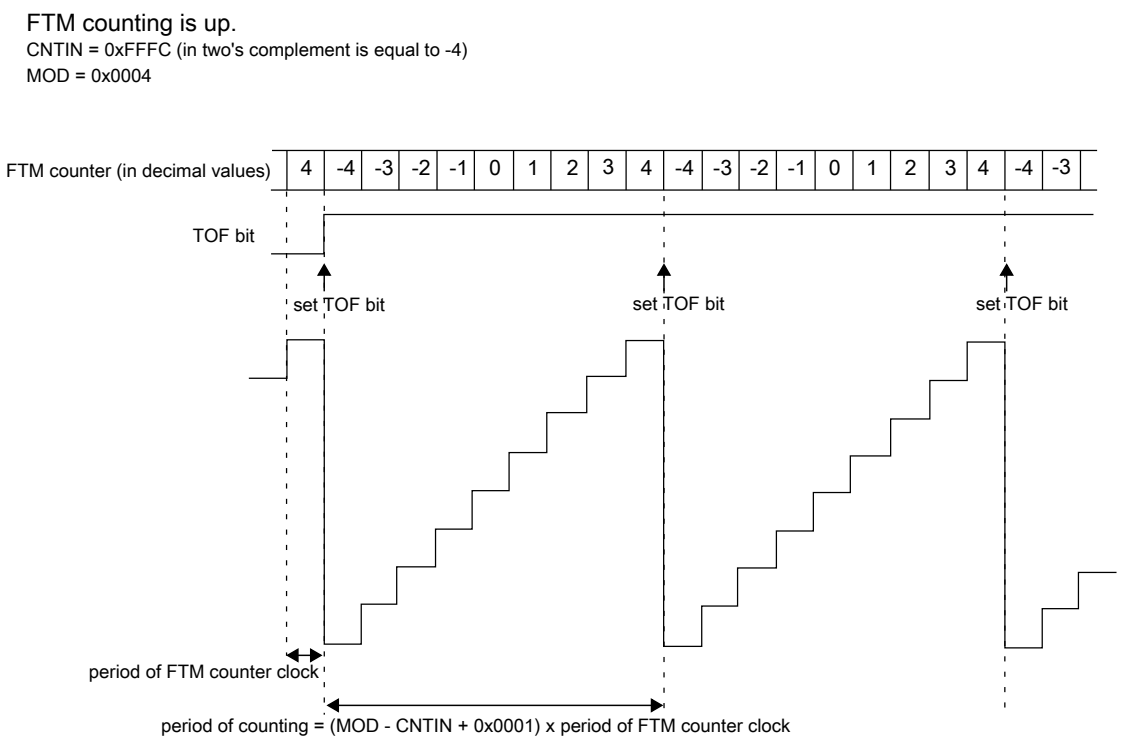


Figure 19-4. Example of FTM up and signed counting

Table 19-8. FTM counting based on CNTIN value

When	Then
CNTIN = 0x0000	The FTM counting is equivalent to TPM up counting, that is, up and unsigned counting. See the following figure.
CNTIN[15] = 1	The initial value of the FTM counter is a negative number in two's complement, so the FTM counting is up and signed.
CNTIN[15] = 0 and CNTIN ≠ 0x0000	The initial value of the FTM counter is a positive number, so the FTM counting is up and unsigned.

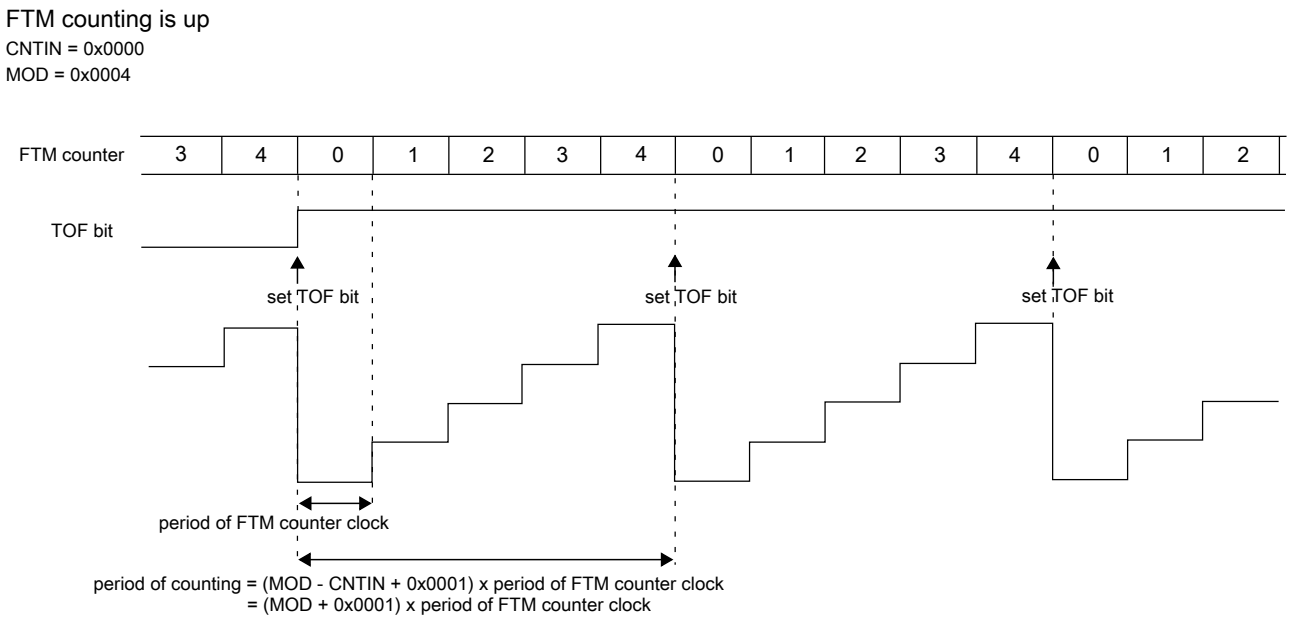


Figure 19-5. Example of FTM up counting with CNTIN = 0x0000

Note

- FTM operation is only valid when the value of the CNTIN register is less than the value of the MOD register, either in the unsigned counting or signed counting. It is the responsibility of the software to ensure that the values in the CNTIN and MOD registers meet this requirement. Any values of CNTIN and MOD that do not satisfy this criteria can result in unpredictable behavior.
- MOD = CNTIN is a redundant condition. In this case, the FTM counter is always equal to MOD and the TOF bit is set in each rising edge of the FTM counter clock.
- When MOD = 0x0000, CNTIN = 0x0000, for example after reset, and FTMEN = 1, the FTM counter remains stopped at 0x0000 until a non-zero value is written into the MOD or CNTIN registers.
- Setting CNTIN to be greater than the value of MOD is not recommended as this unusual setting may make the FTM operation difficult to comprehend. However, there is no restriction on this configuration, and an example is shown in the following figure.

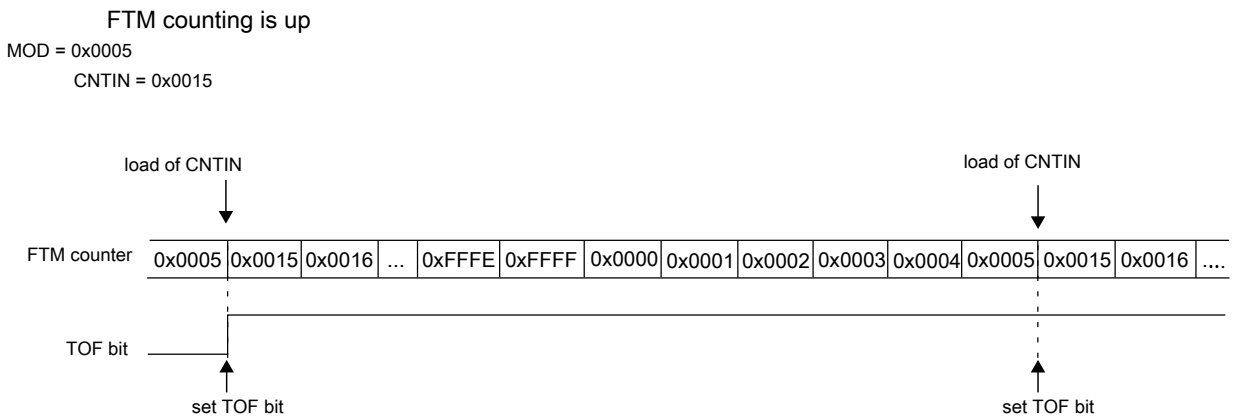


Figure 19-6. Example of up counting when the value of CNTIN is greater than the value of MOD

19.5.3.2 Up-down counting

Up-down counting is selected when:

- QUADEN = 0, and
- CPWMS = 1

CNTIN defines the starting value of the count and MOD defines the final value of the count. The value of CNTIN is loaded into the FTM counter, and the counter increments until the value of MOD is reached, at which point the counter is decremented until it returns to the value of CNTIN and the up-down counting restarts.

The FTM period when using up-down counting is 2 × (MOD – CNTIN) × period of the FTM counter clock.

The TOF bit is set when the FTM counter changes from MOD to (MOD – 1).

If (CNTIN = 0x0000), the FTM counting is equivalent to TPM up-down counting, that is, up-down and unsigned counting. See the following figure.

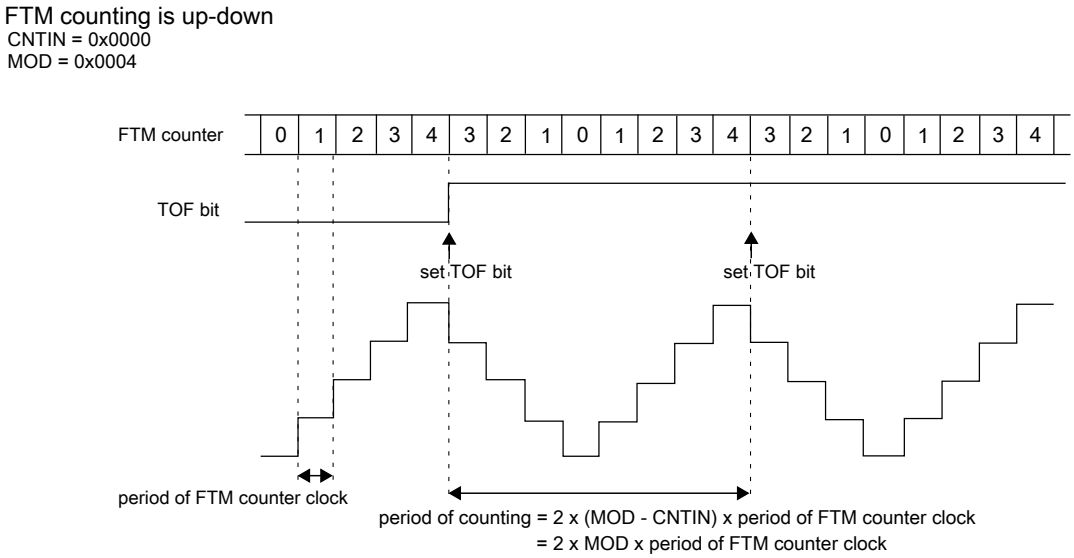


Figure 19-7. Example of up-down counting when CNTIN = 0x0000

Note

When CNTIN is different from zero in the up-down counting, a valid CPWM signal is generated:

- if CnV > CNTIN, or
- if CnV = 0 or if CnV[15] = 1. In this case, 0% CPWM is generated.

19.5.3.3 Free running counter

If (FTMEN = 0) and (MOD = 0x0000 or MOD = 0xFFFF), the FTM counter is a free running counter. In this case, the FTM counter runs free from 0x0000 through 0xFFFF and the TOF bit is set when the FTM counter changes from 0xFFFF to 0x0000. See the following figure.

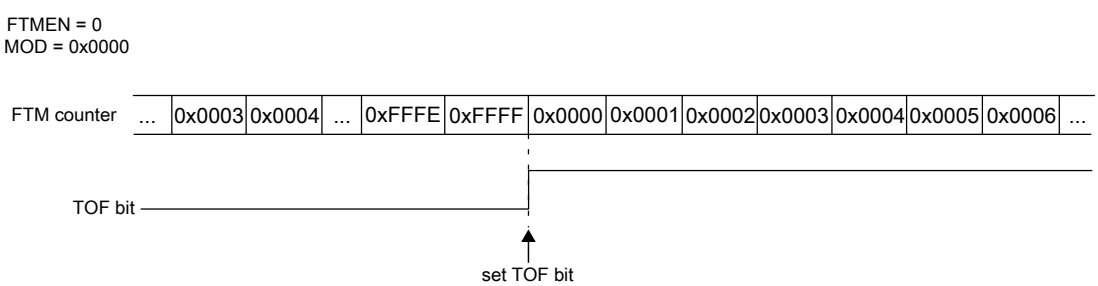


Figure 19-8. Example when the FTM counter is free running

The FTM counter is also a free running counter when:

- FTMEN = 1
- QUADEN = 0
- CPWMS = 0
- CNTIN = 0x0000, and
- MOD = 0xFFFF

19.5.3.4 Counter reset

Any one of the following cases resets the FTM counter to the value in the CNTIN register and the channels output to its initial value, except for channels in Output Compare mode.

- Any write to CNT.
- FTM counter synchronization.

19.5.3.5 When the TOF bit is set

The NUMTOF[4:0] bits define the number of times that the FTM counter overflow should occur before the TOF bit to be set. If NUMTOF[4:0] = 0x00, then the TOF bit is set at each FTM counter overflow.

Initialize the FTM counter, by writing to CNT, after writing to the NUMTOF[4:0] bits to avoid confusion about when the first counter overflow will occur.

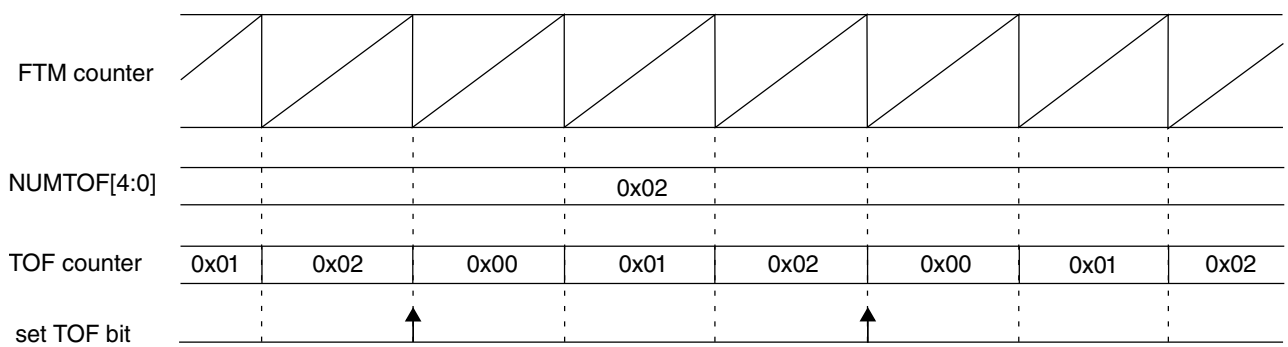


Figure 19-9. Periodic TOF when NUMTOF = 0x02

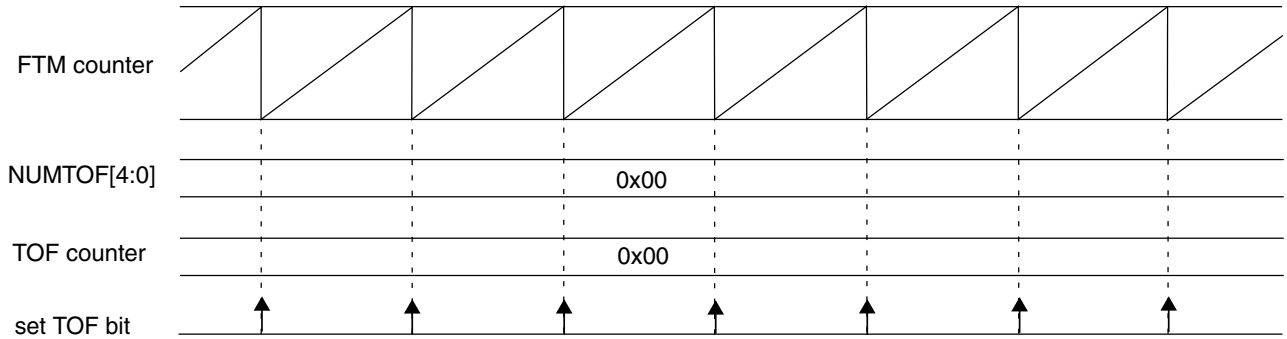


Figure 19-10. Periodic TOF when NUMTOF = 0x00

19.5.4 Input Capture mode

The Input Capture mode is selected when:

- DECAPEN = 0
- COMBINE = 0
- CPWMS = 0
- MSnB:MSnA = 0:0, and
- ELSnB:ELSnA ≠ 0:0

When a selected edge occurs on the channel input, the current value of the FTM counter is captured into the CnV register, at the same time the CHnF bit is set and the channel interrupt is generated if enabled by CHnIE = 1. See the following figure.

When a channel is configured for input capture, the FTMxCHn pin is an edge-sensitive input. ELSnB:ELSnA control bits determine which edge, falling or rising, triggers input-capture event. Note that the maximum frequency for the channel input signal to be detected correctly is system clock divided by 4, which is required to meet Nyquist criteria for signal sampling.

Writes to the CnV register is ignored in Input Capture mode.

While in , the input capture function works as configured. When a selected edge event occurs, the FTM counter value, which is frozen because of , is captured into the CnV register and the CHnF bit is set.

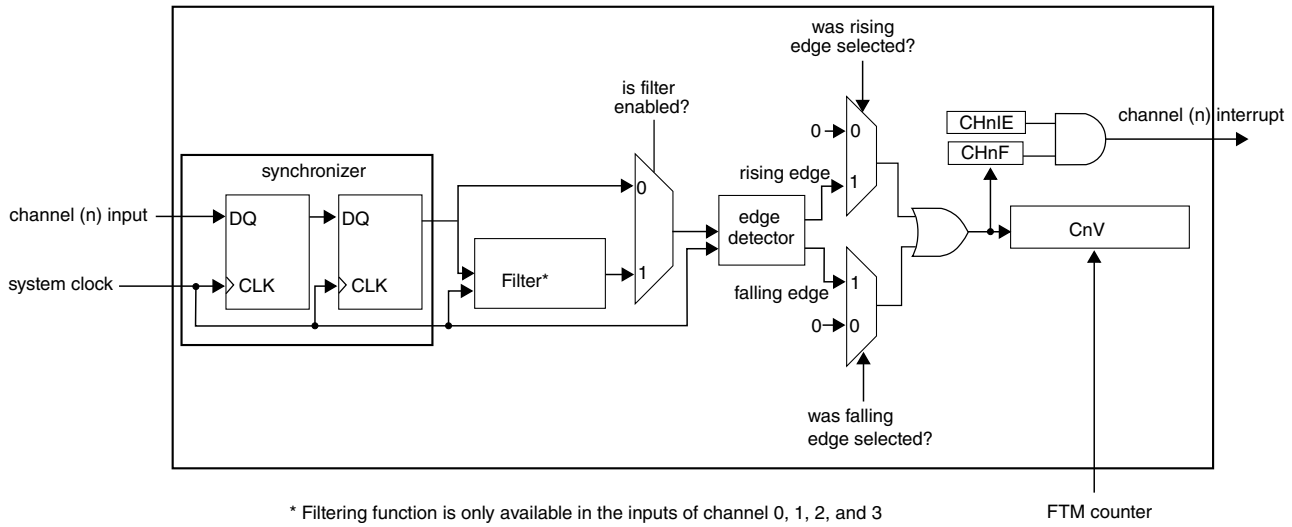


Figure 19-11. Input Capture mode

If the channel input does not have a filter enabled, then the input signal is always delayed 3 rising edges of the system clock, that is, two rising edges to the synchronizer plus one more rising edge to the edge detector. In other words, the CHnF bit is set on the third rising edge of the system clock after a valid edge occurs on the channel input.

19.5.4.1 Filter for Input Capture mode

The filter function is only available on channels 0, 1, 2, and 3.

First, the input signal is synchronized by the system clock. Following synchronization, the input signal enters the filter block. See the following figure.

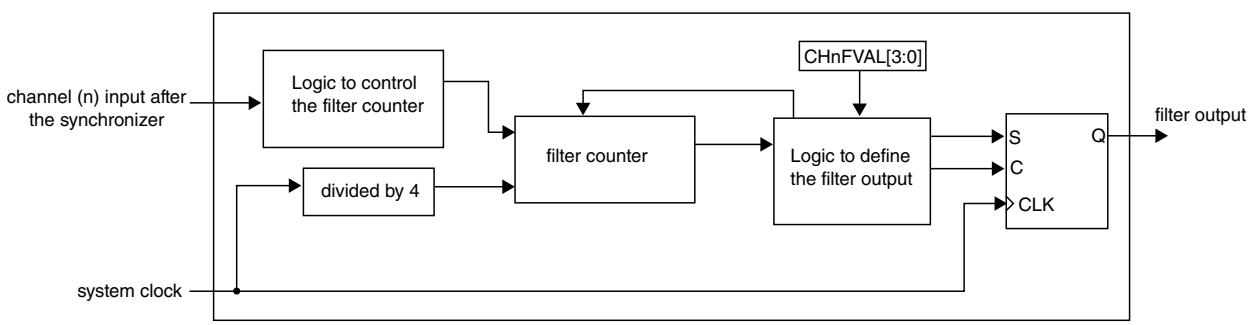


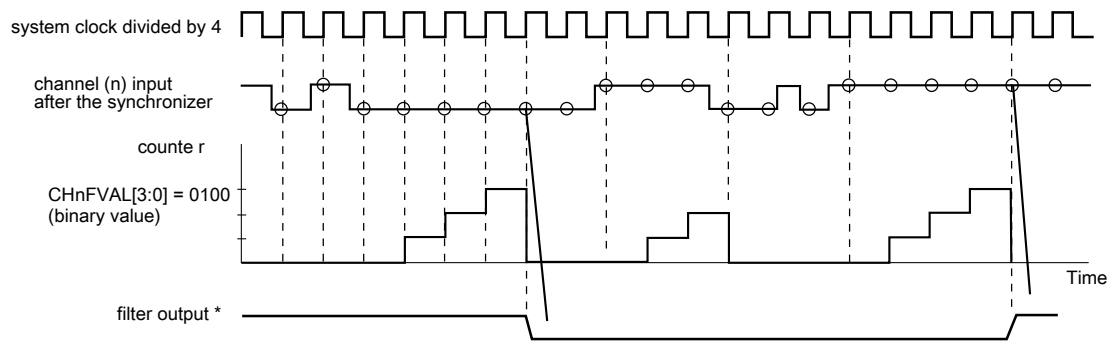
Figure 19-12. Channel input filter

When there is a state change in the input signal, the counter is reset and starts counting up. As long as the new state is stable on the input, the counter continues to increment. When the counter is equal to CHnFVAL[3:0], the state change of the input signal is validated. It is then transmitted as a pulse edge to the edge detector.

If the opposite edge appears on the input signal before it can be validated, the counter is reset. At the next input transition, the counter starts counting again. Any pulse that is shorter than the minimum value selected by CHnFVAL[3:0] (× 4 system clocks) is regarded as a glitch and is not passed on to the edge detector. A timing diagram of the input filter is shown in the following figure.

The filter function is disabled when CHnFVAL[3:0] bits are zero. In this case, the input signal is delayed 3 rising edges of the system clock. If (CHnFVAL[3:0] ≠ 0000), then the input signal is delayed by the minimum pulse width (CHnFVAL[3:0] × 4 system clocks) plus a further 4 rising edges of the system clock: two rising edges to the synchronizer, one rising edge to the filter output, plus one more to the edge detector. In other words, CHnF is set (4 + 4 × CHnFVAL[3:0]) system clock periods after a valid edge occurs on the channel input.

The clock for the counter in the channel input filter is the system clock divided by 4.



* Note: Filter output is delayed one system clock of filter counter logic output.

Figure 19-13. Channel input filter example

The figure below shows an example of input capture with filter enabled and the delay added by each part of the input capture logic. Note that the input signal is delayed only by the synchronizer and edge dector logic if the filter is disabled.

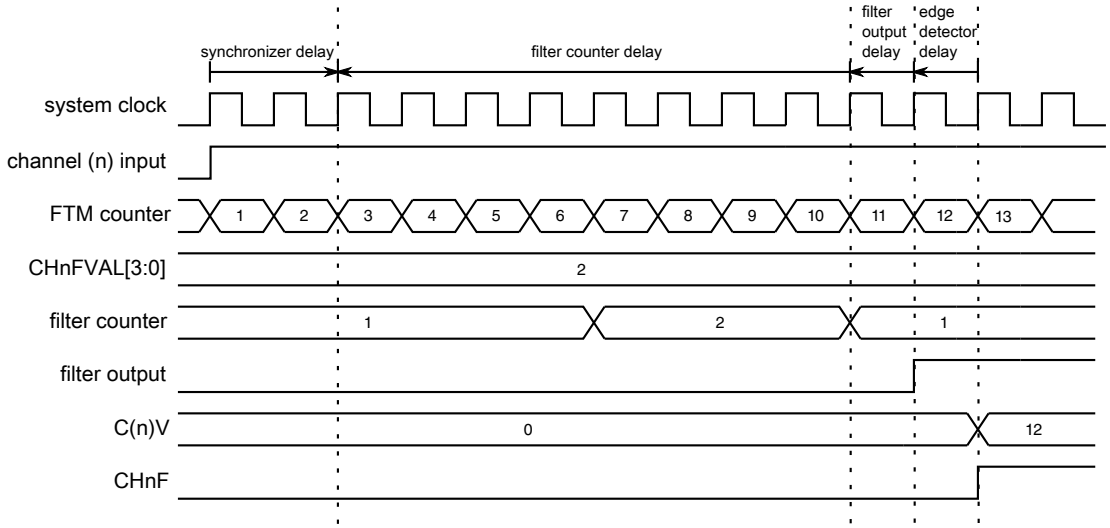


Figure 19-14. Input capture example

19.5.5 Output Compare mode

The Output Compare mode is selected when:

- DECAPEN = 0
- COMBINE = 0
- CPWMS = 0, and
- MSnB:MSnA = 0:1

In Output Compare mode, the FTM can generate timed pulses with programmable position, polarity, duration, and frequency. When the counter matches the value in the CnV register of an output compare channel, the channel (n) output can be set, cleared, or toggled.

When a channel is initially configured to Toggle mode, the previous value of the channel output is held until the first output compare event occurs.

The CHnF bit is set and the channel (n) interrupt is generated if CHnIE = 1 at the channel (n) match (FTM counter = CnV).

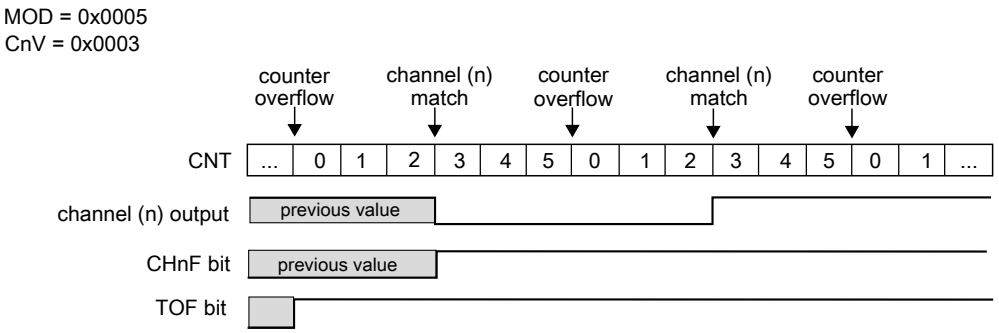


Figure 19-15. Example of the Output Compare mode when the match toggles the channel output

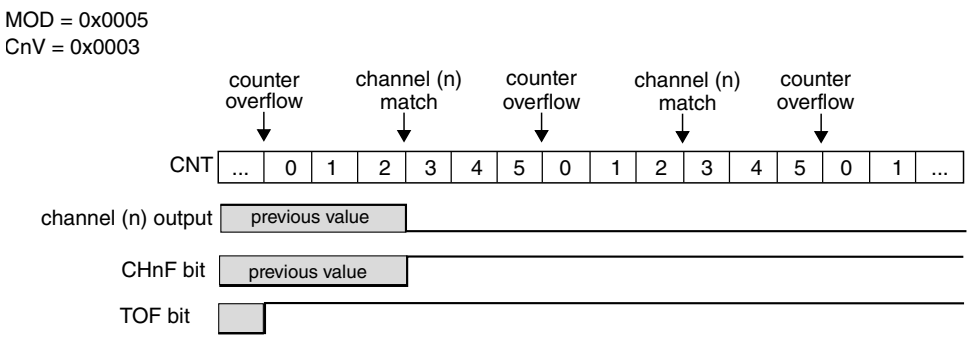


Figure 19-16. Example of the Output Compare mode when the match clears the channel output

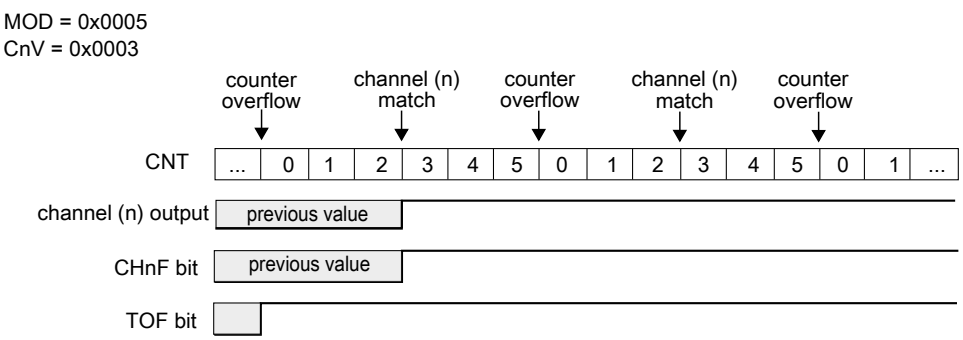


Figure 19-17. Example of the Output Compare mode when the match sets the channel output

If (ELSnB:ELSnA = 0:0) when the counter reaches the value in the CnV register, the CHnF bit is set and the channel (n) interrupt is generated if CHnIE = 1, however the channel (n) output is not modified and controlled by FTM.

19.5.6 Edge-Aligned PWM (EPWM) mode

The Edge-Aligned mode is selected when:

- QUADEN = 0

- DECAPEN = 0
- COMBINE = 0
- CPWMS = 0, and
- MSnB = 1

The EPWM period is determined by (MOD − CNTIN + 0x0001) and the pulse width (duty cycle) is determined by (CnV − CNTIN).

The CHnF bit is set and the channel (n) interrupt is generated if CHnIE = 1 at the channel (n) match (FTM counter = CnV), that is, at the end of the pulse width.

This type of PWM signal is called edge-aligned because the leading edges of all PWM signals are aligned with the beginning of the period, which is the same for all channels within an FTM.

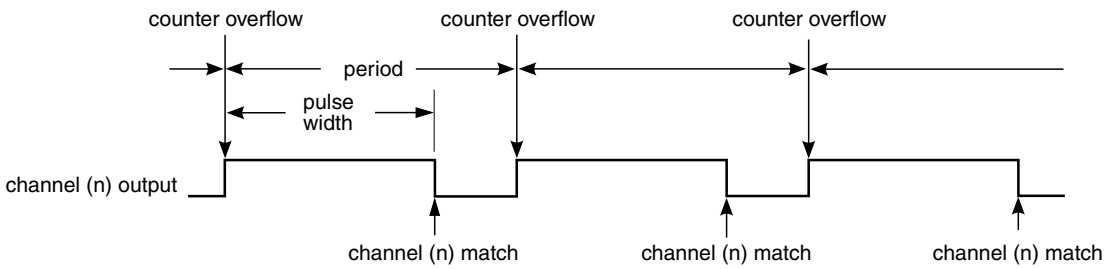


Figure 19-18. EPWM period and pulse width with ELSnB:ELSnA = 1:0

If (ELSnB:ELSnA = 0:0) when the counter reaches the value in the CnV register, the CHnF bit is set and the channel (n) interrupt is generated if CHnIE = 1, however the channel (n) output is not controlled by FTM.

If (ELSnB:ELSnA = 1:0), then the channel (n) output is forced high at the counter overflow when the CNTIN register value is loaded into the FTM counter, and it is forced low at the channel (n) match (FTM counter = CnV). See the following figure.

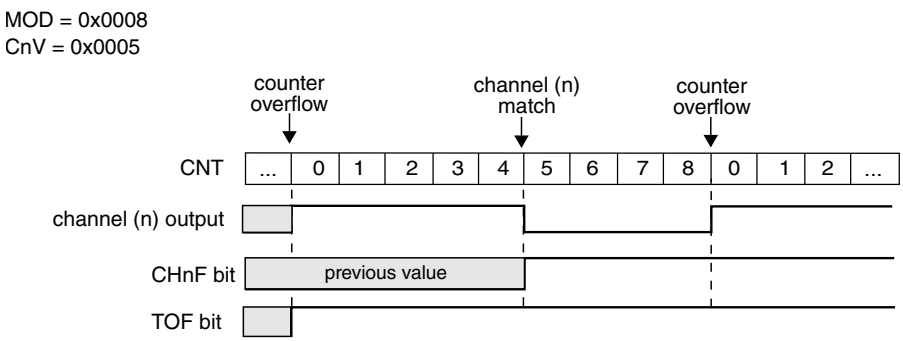


Figure 19-19. EPWM signal with ELSnB:ELSnA = 1:0

If (ELSnB:ELSnA = X:1), then the channel (n) output is forced low at the counter overflow when the CNTIN register value is loaded into the FTM counter, and it is forced high at the channel (n) match (FTM counter = CnV). See the following figure.

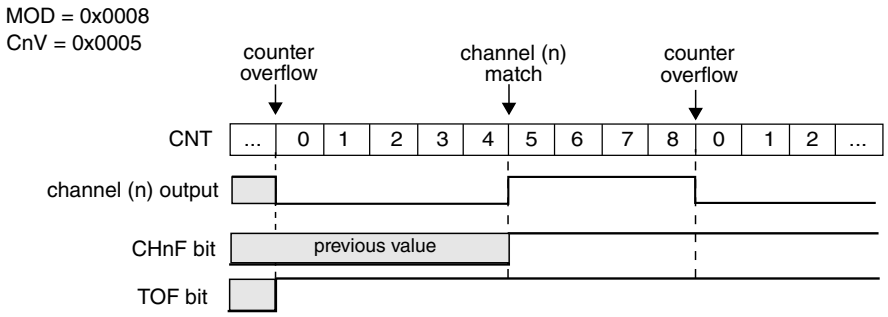


Figure 19-20. EPWM signal with ELSnB:ELSnA = X:1

If (CnV = 0x0000), then the channel (n) output is a 0% duty cycle EPWM signal and CHnF bit is not set even when there is the channel (n) match.

If (CnV > MOD), then the channel (n) output is a 100% duty cycle EPWM signal and CHnF bit is not set.Therefore, MOD must be less than 0xFFFF in order to get a 100% duty cycle EPWM signal.

Note

When CNTIN is different from zero the following EPWM signals can be generated:

- 0% EPWM signal if CnV = CNTIN,
- EPWM signal between 0% and 100% if CNTIN < CnV <= MOD,
- 100% EPWM signal when CNTIN > CnV or CnV > MOD.

19.5.7 Center-Aligned PWM (CPWM) mode

The Center-Aligned mode is selected when:

- QUADEN = 0
- DECAPEN = 0
- COMBINE = 0, and
- CPWMS = 1

The CPWM pulse width (duty cycle) is determined by 2 × (CnV − CNTIN) and the period is determined by 2 × (MOD − CNTIN). See the following figure. MOD must be kept in the range of 0x0001 to 0x7FFF because values outside this range can produce ambiguous results.

In the CPWM mode, the FTM counter counts up until it reaches MOD and then counts down until it reaches CNTIN.

The CHnF bit is set and channel (n) interrupt is generated (if CHnIE = 1) at the channel (n) match (FTM counter = CnV) when the FTM counting is down (at the begin of the pulse width) and when the FTM counting is up (at the end of the pulse width).

This type of PWM signal is called center-aligned because the pulse width centers for all channels are aligned with the value of CNTIN.

The other channel modes are not compatible with the up-down counter (CPWMS = 1). Therefore, all FTM channels must be used in CPWM mode when (CPWMS = 1).

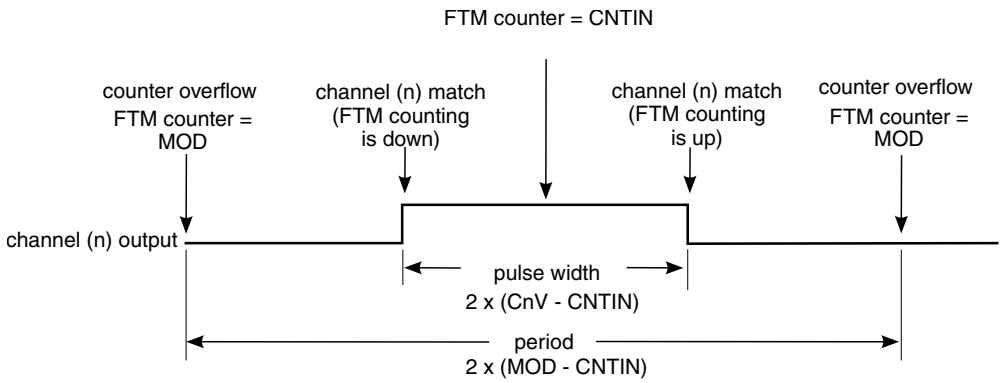


Figure 19-21. CPWM period and pulse width with ELSnB:ELSnA = 1:0

If (ELSnB:ELSnA = 0:0) when the FTM counter reaches the value in the CnV register, the CHnF bit is set and the channel (n) interrupt is generated (if CHnIE = 1), however the channel (n) output is not controlled by FTM.

If (ELSnB:ELSnA = 1:0), then the channel (n) output is forced high at the channel (n) match (FTM counter = CnV) when counting down, and it is forced low at the channel (n) match when counting up. See the following figure.

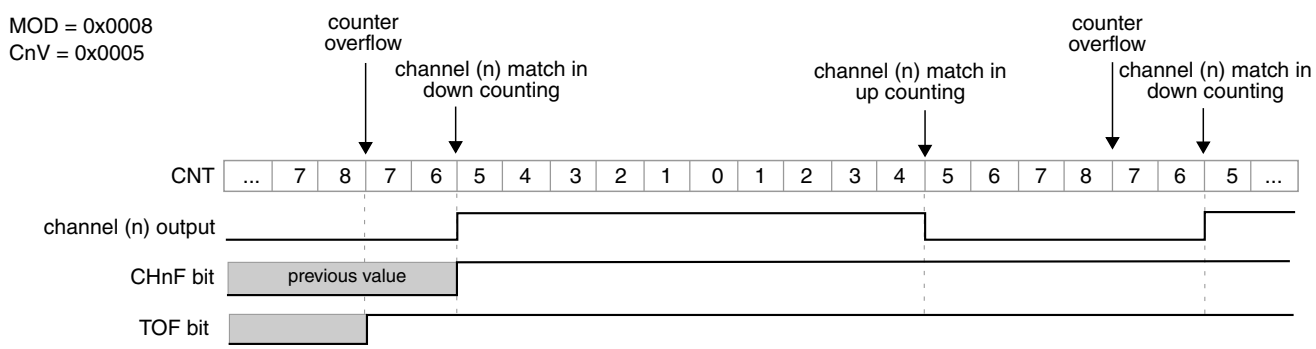


Figure 19-22. CPWM signal with ELSnB:ELSnA = 1:0

If (ELSnB:ELSnA = X:1), then the channel (n) output is forced low at the channel (n) match (FTM counter = CnV) when counting down, and it is forced high at the channel (n) match when counting up. See the following figure.

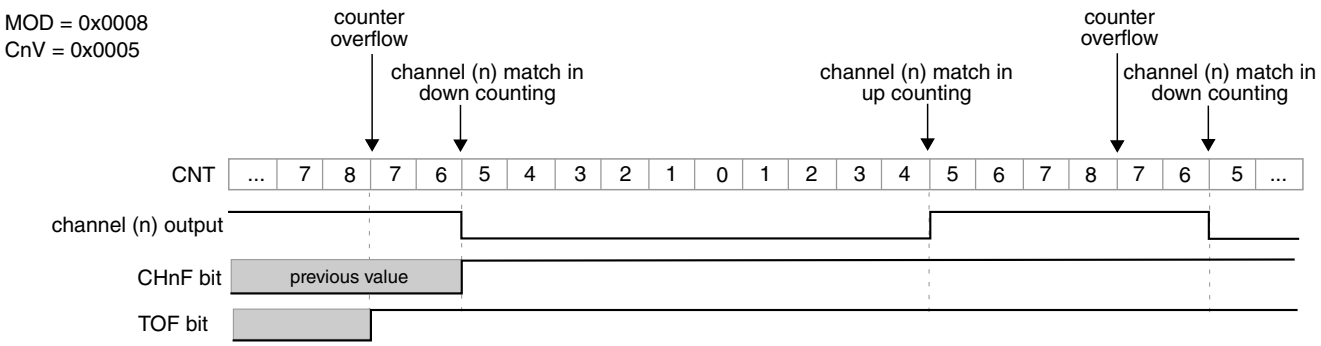


Figure 19-23. CPWM signal with ELSnB:ELSnA = X:1

If (CnV = 0x0000) or CnV is a negative value, that is (CnV[15] = 1), then the channel (n) output is a 0% duty cycle CPWM signal and CHnF bit is not set even when there is the channel (n) match.

If CnV is a positive value, that is (CnV[15] = 0), (CnV ≥ MOD), and (MOD ≠ 0x0000), then the channel (n) output is a 100% duty cycle CPWM signal and CHnF bit is not set even when there is the channel (n) match. This implies that the usable range of periods set by MOD is 0x0001 through 0x7FFE, 0x7FFF if you do not need to generate a 100% duty cycle CPWM signal. This is not a significant limitation because the resulting period is much longer than required for normal applications.

The CPWM mode must not be used when the FTM counter is a free running counter.

19.5.8 Combine mode

The Combine mode is selected when:

- QUADEN = 0
- DECAPEN = 0
- COMBINE = 1, and
- CPWMS = 0

In Combine mode, an even channel (n) and adjacent odd channel (n+1) are combined to generate a PWM signal in the channel (n) output.

In the Combine mode, the PWM period is determined by (MOD − CNTIN + 0x0001) and the PWM pulse width (duty cycle) is determined by (|C(n+1)V − C(n)V|).

The CHnF bit is set and the channel (n) interrupt is generated (if CHnIE = 1) at the channel (n) match (FTM counter = C(n)V). The CH(n+1)F bit is set and the channel (n +1) interrupt is generated, if CH(n+1)IE = 1, at the channel (n+1) match (FTM counter = C(n+1)V).

If (ELSnB:ELSnA = 1:0), then the channel (n) output is forced low at the beginning of the period (FTM counter = CNTIN) and at the channel (n+1) match (FTM counter = C(n +1)V). It is forced high at the channel (n) match (FTM counter = C(n)V). See the following figure.

If (ELSnB:ELSnA = X:1), then the channel (n) output is forced high at the beginning of the period (FTM counter = CNTIN) and at the channel (n+1) match (FTM counter = C(n +1)V). It is forced low at the channel (n) match (FTM counter = C(n)V). See the following figure.

In Combine mode, the ELS(n+1)B and ELS(n+1)A bits are not used in the generation of the channels (n) and (n+1) output. However, if (ELSnB:ELSnA = 0:0) then the channel (n) output is not controlled by FTM, and if (ELS(n+1)B:ELS(n+1)A = 0:0) then the channel (n+1) output is not controlled by FTM.

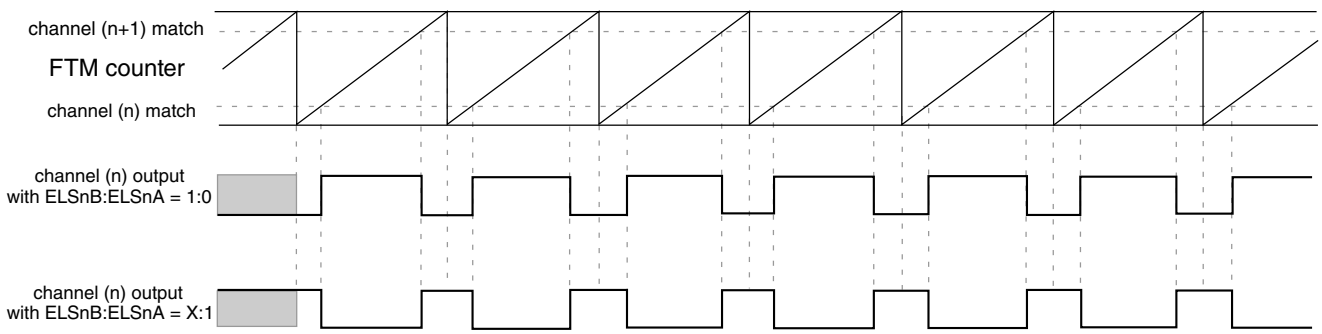


Figure 19-24. Combine mode

The following figures illustrate the PWM signals generation using Combine mode.

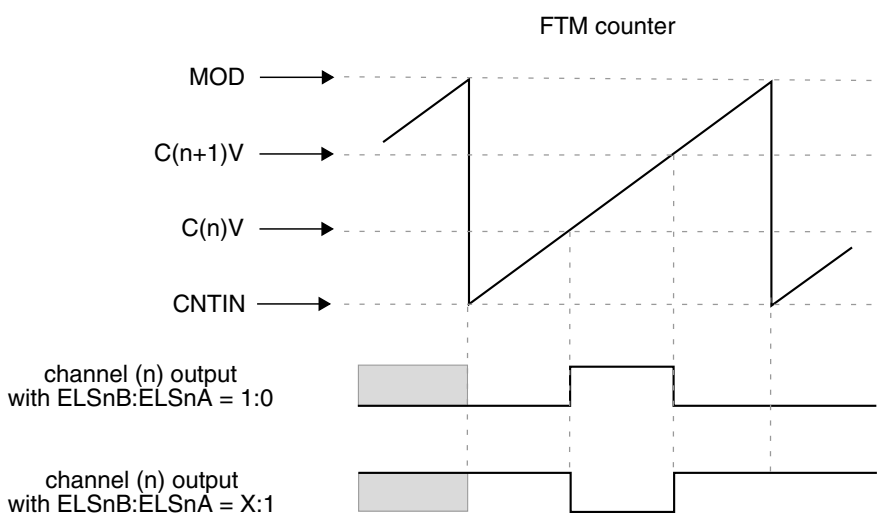


Figure 19-25. Channel (n) output if (CNTIN < C(n)V < MOD) and (CNTIN < C(n+1)V < MOD) and (C(n)V < C(n+1)V)

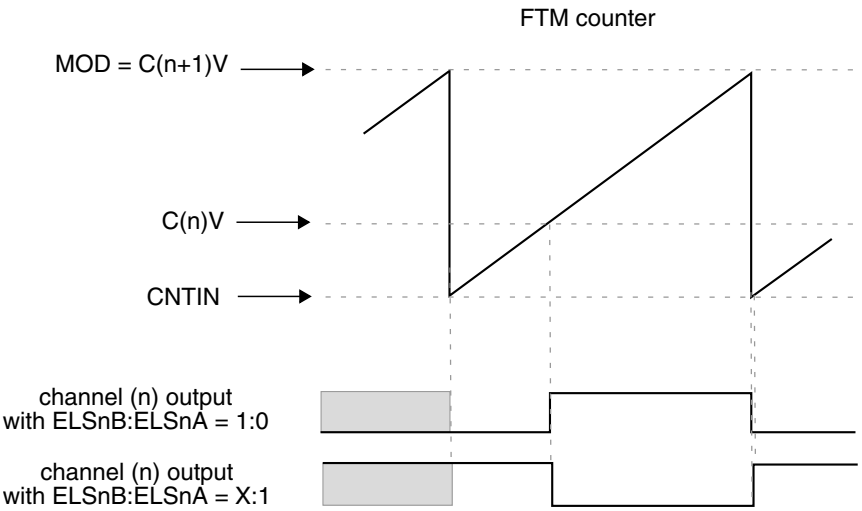


Figure 19-26. Channel (n) output if (CNTIN < C(n)V < MOD) and (C(n+1)V = MOD)

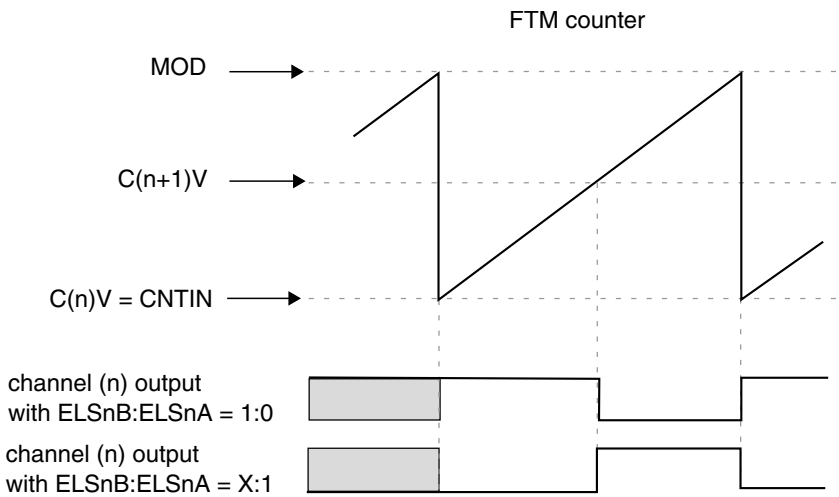


Figure 19-27. Channel (n) output if (C(n)V = CNTIN) and (CNTIN < C(n+1)V < MOD)

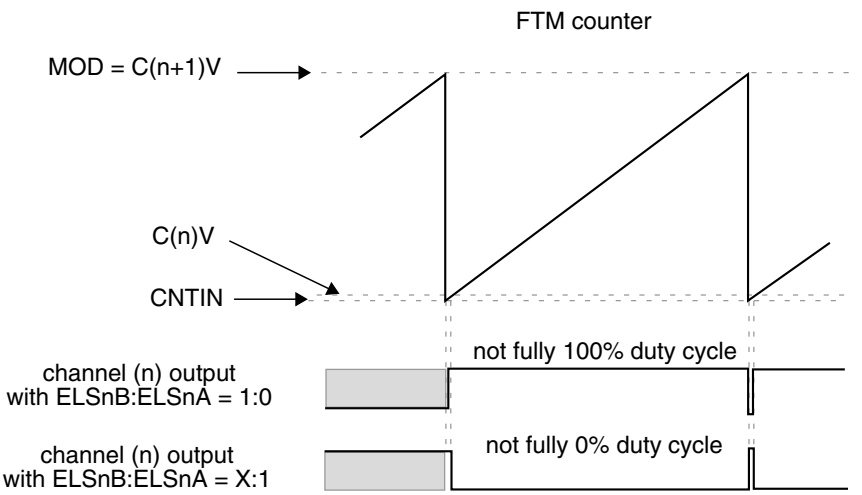


Figure 19-28. Channel (n) output if (CNTIN < C(n)V < MOD) and (C(n)V is Almost Equal to CNTIN) and (C(n+1)V = MOD)

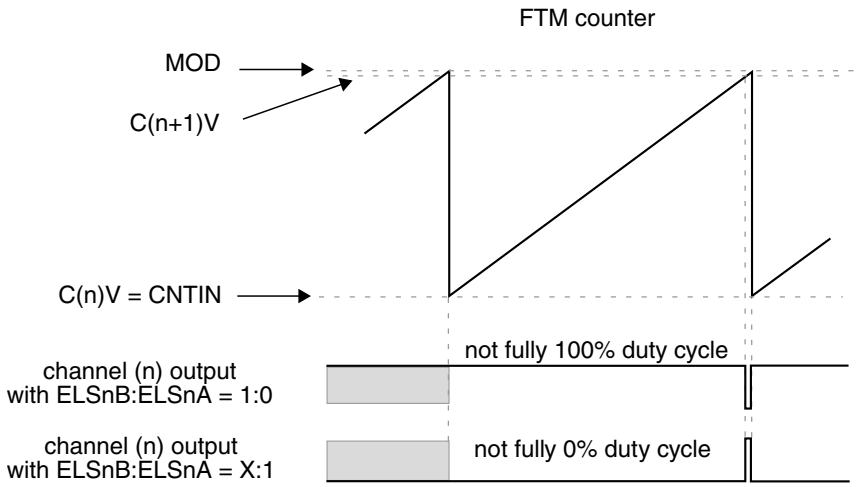


Figure 19-29. Channel (n) output if (C(n)V = CNTIN) and (CNTIN < C(n+1)V < MOD) and (C(n+1)V is Almost Equal to MOD)

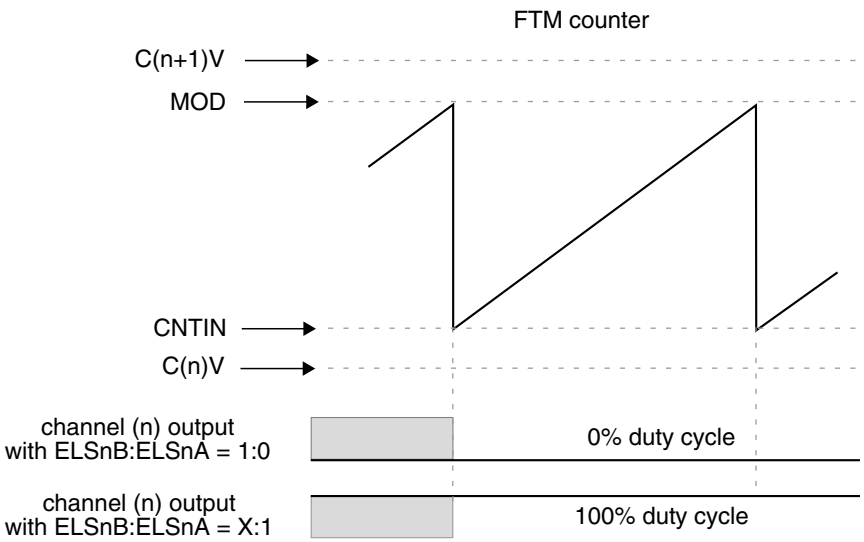


Figure 19-30. Channel (n) output if C(n)V and C(n+1)V are not between CNTIN and MOD

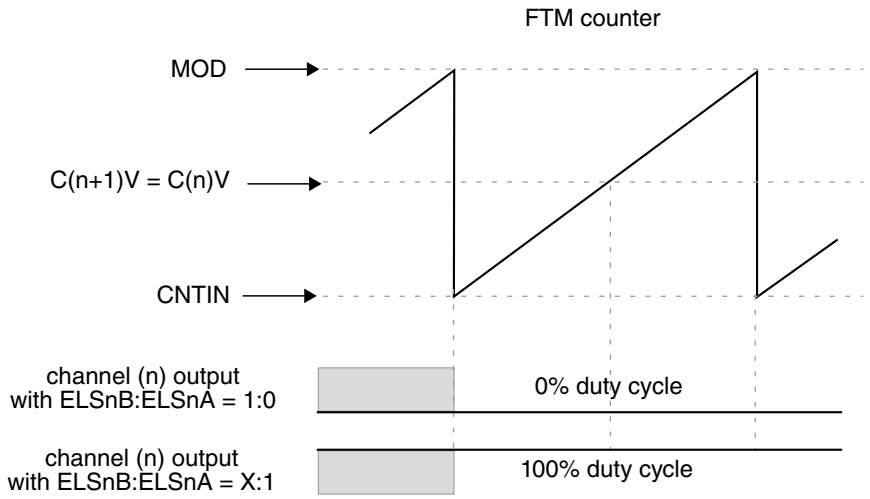


Figure 19-31. Channel (n) output if (CNTIN < C(n)V < MOD) and (CNTIN < C(n+1)V < MOD) and (C(n)V = C(n+1)V)

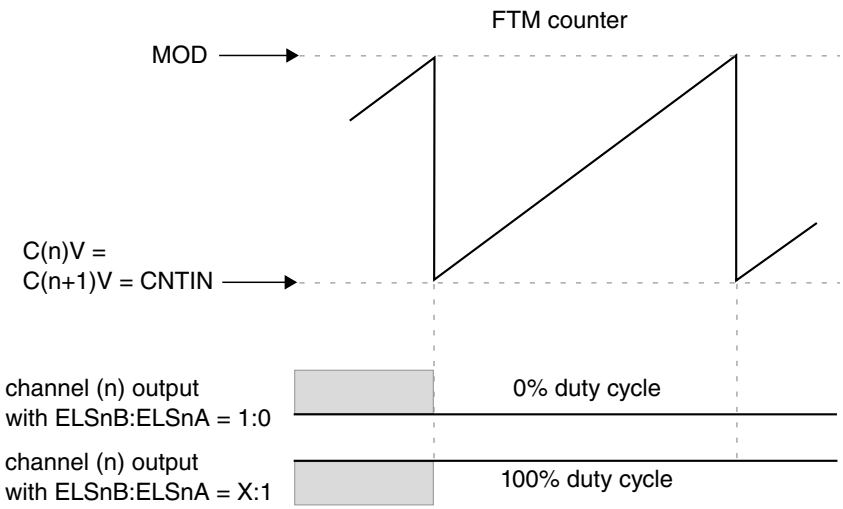


Figure 19-32. Channel (n) output if (C(n)V = C(n+1)V = CNTIN)

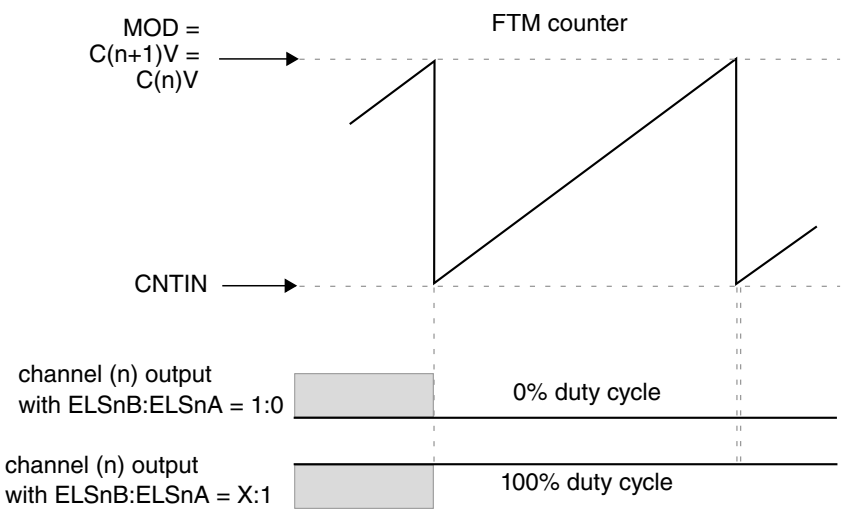


Figure 19-33. Channel (n) output if (C(n)V = C(n+1)V = MOD)

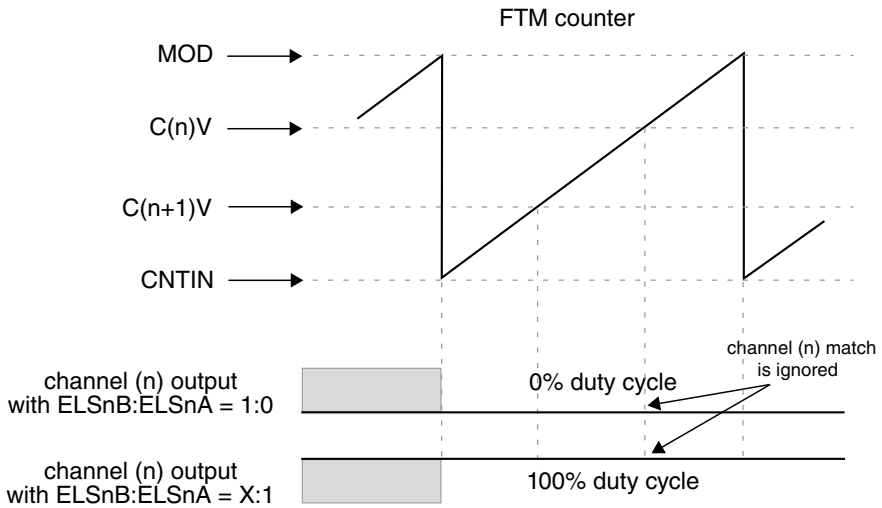


Figure 19-34. Channel (n) output if (CNTIN < C(n)V < MOD) and (CNTIN < C(n+1)V < MOD) and (C(n)V > C(n+1)V)

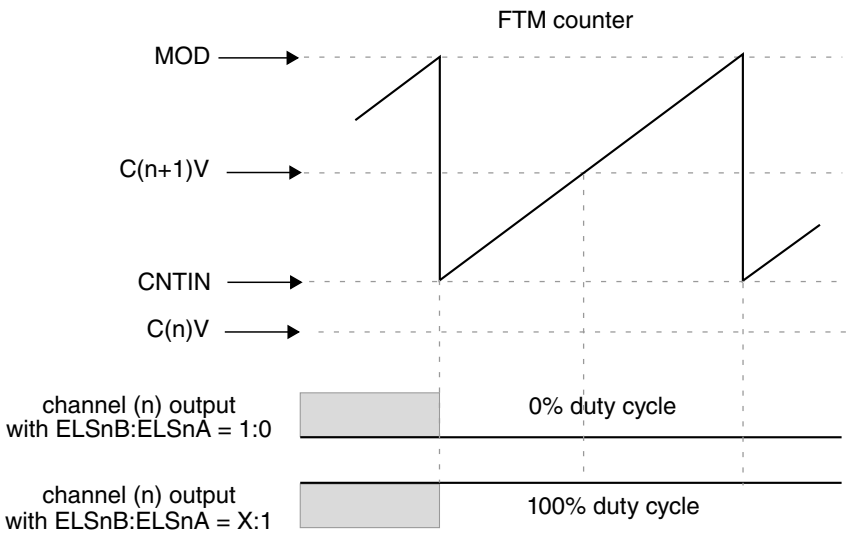


Figure 19-35. Channel (n) output if (C(n)V < CNTIN) and (CNTIN < C(n+1)V < MOD)

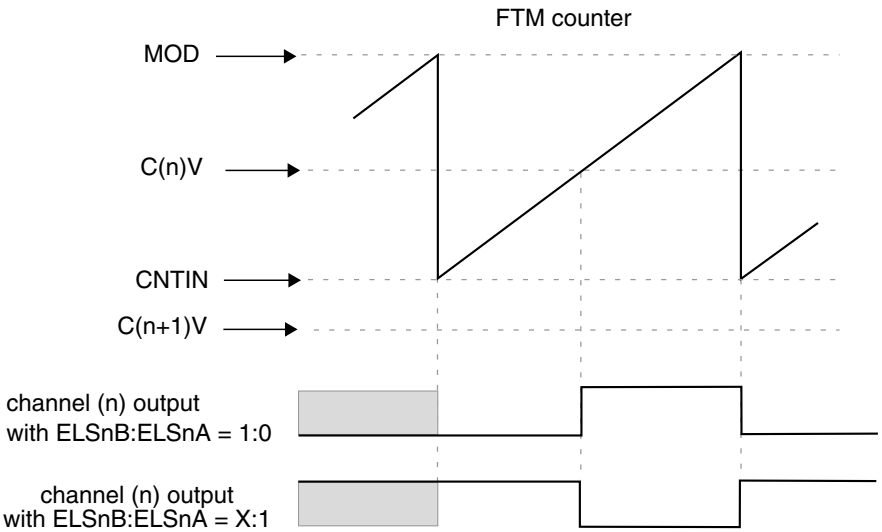


Figure 19-36. Channel (n) output if (C(n+1)V < CNTIN) and (CNTIN < C(n)V < MOD)

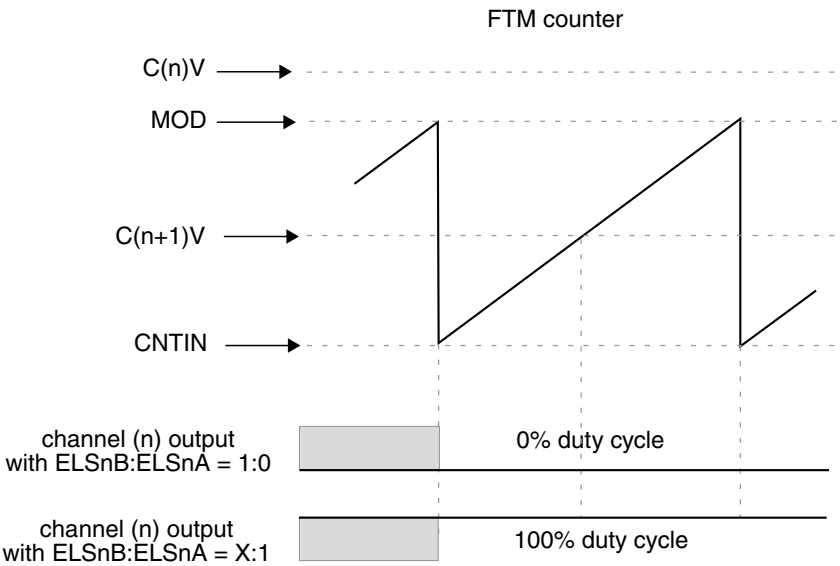


Figure 19-37. Channel (n) output if (C(n)V > MOD) and (CNTIN < C(n+1)V < MOD)

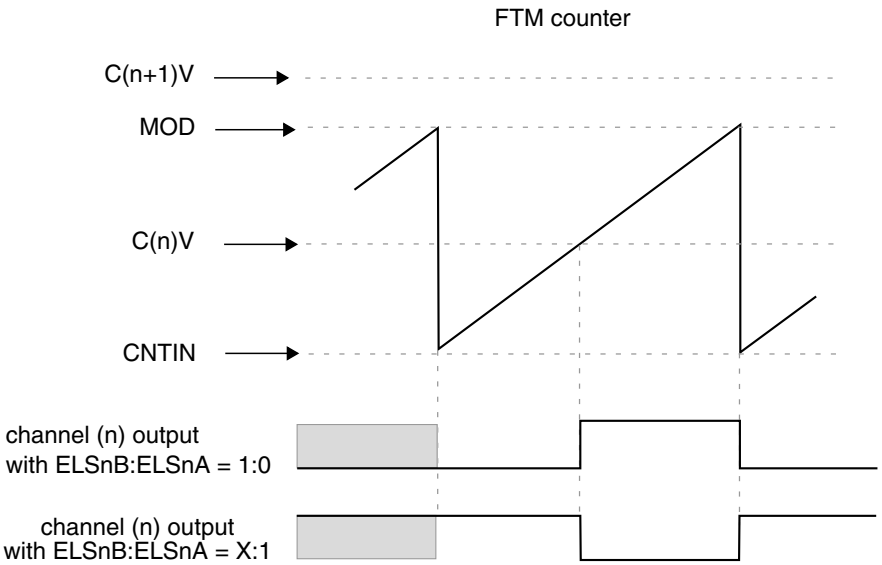


Figure 19-38. Channel (n) output if (C(n+1)V > MOD) and (CNTIN < C(n)V < MOD)

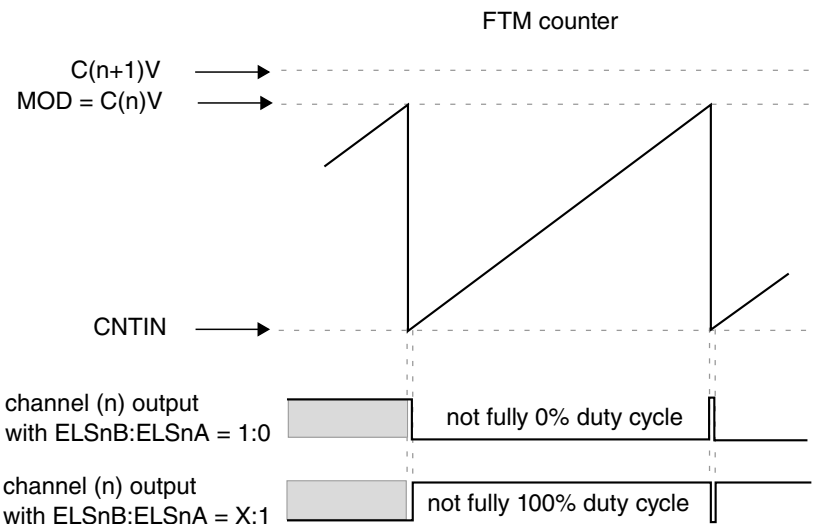


Figure 19-39. Channel (n) output if (C(n+1)V > MOD) and (CNTIN < C(n)V = MOD)

19.5.8.1 Asymmetrical PWM

In Combine mode, the control of the PWM signal first edge, when the channel (n) match occurs, that is, FTM counter = C(n)V, is independent of the control of the PWM signal second edge, when the channel (n+1) match occurs, that is, FTM counter = C(n+1)V. So, Combine mode allows the generation of asymmetrical PWM signals.

19.5.9 Complementary mode

The Complementary mode is selected when:

- QUADEN = 0
- DECAPEN = 0
- COMP = 1

In Complementary mode, the channel (n+1) output is the inverse of the channel (n) output.

So, the channel (n+1) output is the same as the channel (n) output when:

- QUADEN = 0
- DECAPEN = 0
- COMP = 0

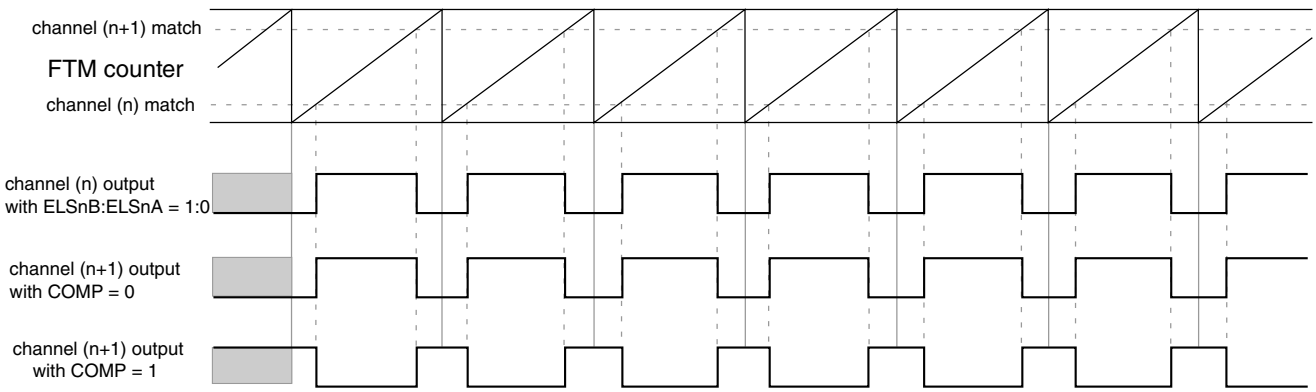


Figure 19-40. Channel (n+1) output in Complementary mode with (ELSnB:ELSnA = 1:0)

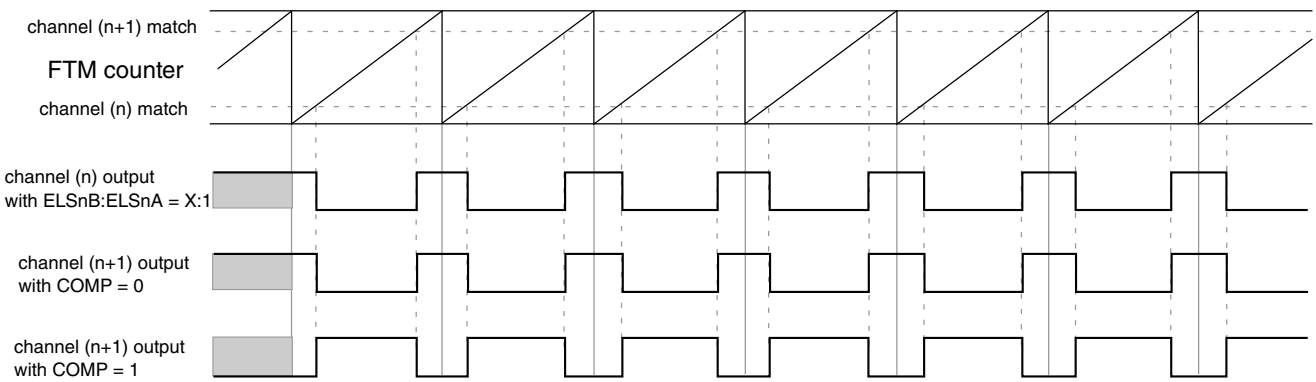


Figure 19-41. Channel (n+1) output in Complementary mode with (ELSnB:ELSnA = X:1)

NOTE

The complementary mode is not available in Output Compare mode.

19.5.10 Registers updated from write buffers

19.5.10.1 CNTIN register update

The following table describes when CNTIN register is updated:

Table 19-9. CNTIN register update

When	Then CNTIN register is updated
CLKS[1:0] = 0:0	When CNTIN register is written, independent of FTMEN bit.
• FTMEN = 0, or • CNTINC = 0	At the next system clock after CNTIN was written.
• FTMEN = 1, • SYNCMODE = 1, and • CNTINC = 1	By the CNTIN register synchronization.

19.5.10.2 MOD register update

The following table describes when MOD register is updated:

Table 19-10. MOD register update

When	Then MOD register is updated
CLKS[1:0] = 0:0	When MOD register is written, independent of FTMEN bit.
• CLKS[1:0] ≠ 0:0, and • FTMEN = 0	According to the CPWMS bit, that is: • If the selected mode is not CPWM then MOD register is updated after MOD register was written and the FTM counter changes from MOD to CNTIN. If the FTM counter is at free-running counter mode then this update occurs when the FTM counter changes from 0xFFFF to 0x0000. • If the selected mode is CPWM then MOD register is updated after MOD register was written and the FTM counter changes from MOD to (MOD – 0x0001).
• CLKS[1:0] ≠ 0:0, and • FTMEN = 1	By the MOD register synchronization.

19.5.10.3 CnV register update

The following table describes when CnV register is updated:

Table 19-11. CnV register update

When	Then CnV register is updated
CLKS[1:0] = 0:0	When CnV register is written, independent of FTMEN bit.
• CLKS[1:0] ≠ 0:0, and • FTMEN = 0	According to the selected mode, that is:

Table continues on the next page...

Table 19-11. CnV register update (continued)

When	Then CnV register is updated
	• If the selected mode is Output Compare, then CnV register is updated on the next FTM counter change, end of the prescaler counting, after CnV register was written. • If the selected mode is EPWM, then CnV register is updated after CnV register was written and the FTM counter changes from MOD to CNTIN. If the FTM counter is at free-running counter mode then this update occurs when the FTM counter changes from 0xFFFF to 0x0000. • If the selected mode is CPWM, then CnV register is updated after CnV register was written and the FTM counter changes from MOD to (MOD – 0x0001).
• CLKS[1:0] ≠ 0:0, and • FTMEN = 1	According to the selected mode, that is: • If the selected mode is output compare then CnV register is updated according to the SYNCEN bit. If (SYNCEN = 0) then CnV register is updated after CnV register was written at the next change of the FTM counter, the end of the prescaler counting. If (SYNCEN = 1) then CnV register is updated by the C(n)V and C(n+1)V register synchronization. • If the selected mode is not output compare and (SYNCEN = 1) then CnV register is updated by the C(n)V and C(n+1)V register synchronization.

19.5.11 PWM synchronization

The PWM synchronization provides an opportunity to update the MOD, CNTIN, CnV, OUTMASK, INVCTRL and SWOCTRL registers with their buffered value and force the FTM counter to the CNTIN register value.

Note

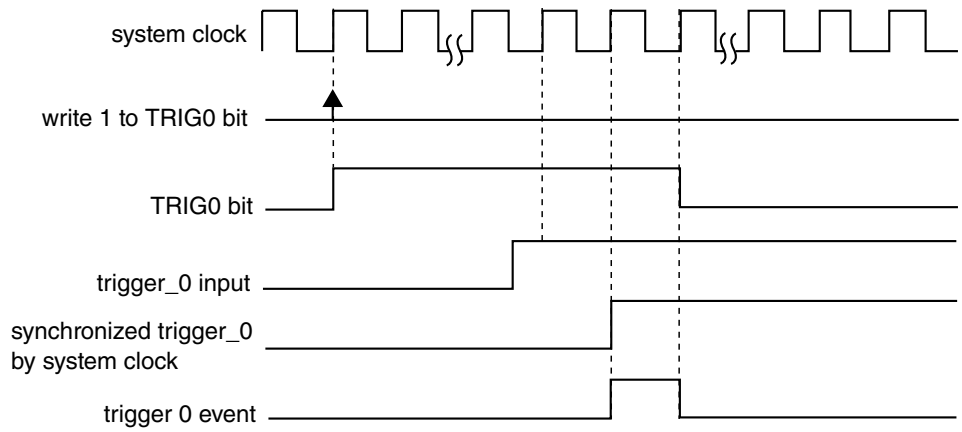
The legacy PWM synchronization (SYNCMODE = 0) is a subset of the enhanced PWM synchronization (SYNCMODE = 1). Thus, only the enhanced PWM synchronization must be used.

19.5.11.1 Hardware trigger

Three hardware trigger signal inputs of the FTM module are enabled when TRIGn = 1, where n = 0, 1 or 2 corresponding to each one of the input signals, respectively. The hardware trigger input n is synchronized by the system clock. The PWM synchronization with hardware trigger is initiated when a rising edge is detected at the enabled hardware trigger inputs.

If (HWTRIGMODE = 0) then the TRIGn bit is cleared when 0 is written to it or when the trigger n event is detected.

In this case, if two or more hardware triggers are enabled (for example, TRIG0 and TRIG1 = 1) and only trigger 1 event occurs, then only TRIG1 bit is cleared. If a trigger n event occurs together with a write setting TRIGn bit, then the synchronization is initiated, but TRIGn bit remains set due to the write operation.



Note
All hardware trigger inputs have the same behavior.

Figure 19-42. Hardware trigger event with HWTRIGMODE = 0

If HWTRIGMODE = 1, then the TRIGn bit is only cleared when 0 is written to it.

NOTE

The HWTRIGMODE bit must be 1 only with enhanced PWM synchronization (SYNCMODE = 1).

19.5.11.2 Software trigger

A software trigger event occurs when 1 is written to the SYNC[SWSYNC] bit. The SWSYNC bit is cleared when 0 is written to it or when the PWM synchronization, initiated by the software event, is completed.

If another software trigger event occurs (by writing another 1 to the SWSYNC bit) at the same time the PWM synchronization initiated by the previous software trigger event is ending, a new PWM synchronization is started and the SWSYNC bit remains equal to 1.

If SYNCMODE = 0 then the SWSYNC bit is also cleared by FTM according to PWMSYNC and REINIT bits. In this case if (PWMSYNC = 1) or (PWMSYNC = 0 and REINIT = 0) then SWSYNC bit is cleared at the next selected loading point after that the software trigger event occurred; see Boundary cycle and loading points and the following figure. If (PWMSYNC = 0) and (REINIT = 1) then SWSYNC bit is cleared when the software trigger event occurs.

If SYNCMODE = 1 then the SWSYNC bit is also cleared by FTM according to the SWRSTCNT bit. If SWRSTCNT = 0 then SWSYNC bit is cleared at the next selected loading point after that the software trigger event occurred; see the following figure. If SWRSTCNT = 1 then SWSYNC bit is cleared when the software trigger event occurs.

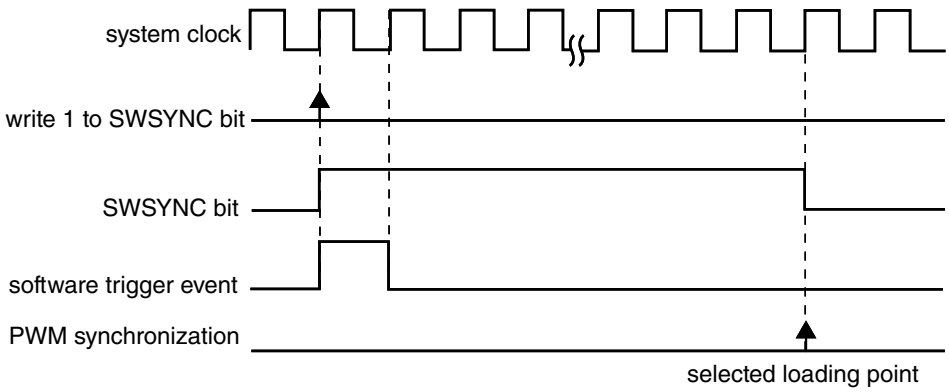


Figure 19-43. Software trigger event

19.5.11.3 Boundary cycle and loading points

The boundary cycle definition is important for the loading points for the registers MOD, CNTIN, and C(n)V.

In Up counting mode, the boundary cycle is defined as when the counter wraps to its initial value (CNTIN). If in Up-down counting mode, then the boundary cycle is defined as when the counter turns from down to up counting and when from up to down counting.

The following figure shows the boundary cycles and the loading points for the registers. In the Up Counting mode, the loading points are enabled if one of CNTMIN or CTMAX bits are 1. In the Up-Down Counting mode, the loading points are selected by CNTMIN and CNTMAX bits, as indicated in the figure. These loading points are safe places for register updates thus allowing a smooth transitions in PWM waveform generation.

For both counting modes, if neither CNTMIN nor CNTMAX are 1, then the boundary cycles are not used as loading points for registers updates. See the register synchronization descriptions in the following sections for details.

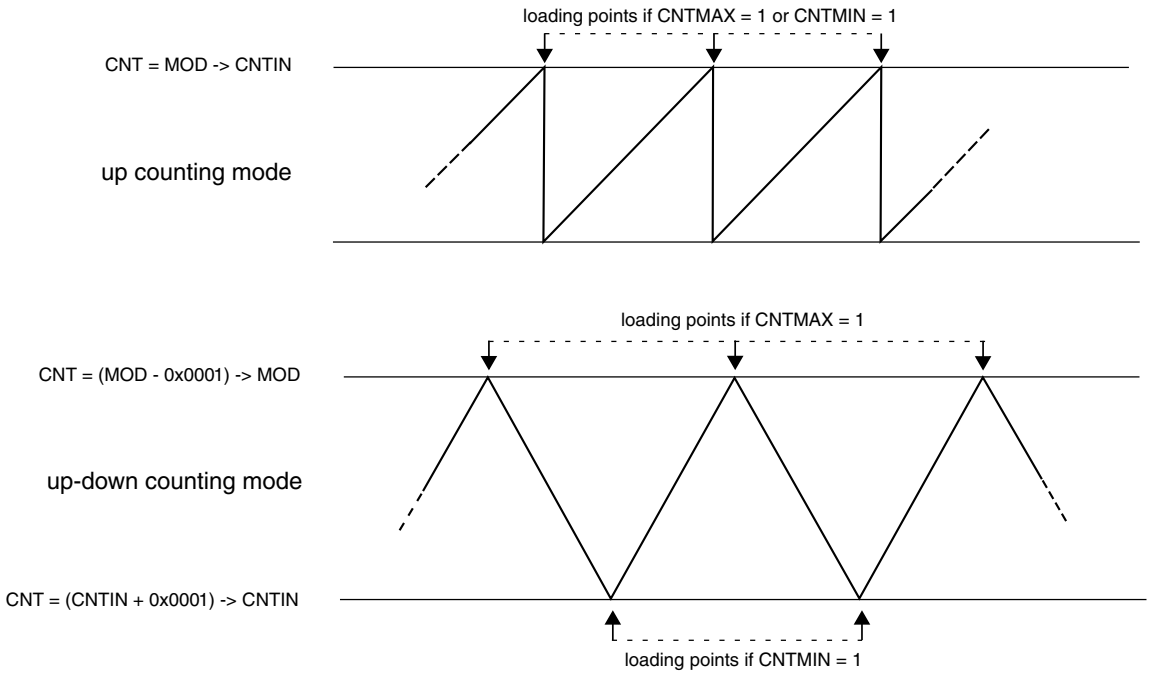


Figure 19-44. Boundary cycles and loading points

19.5.11.4 MOD register synchronization

The MOD register synchronization updates the MOD register with its buffer value. This synchronization is enabled if (FTMEN = 1).

The MOD register synchronization can be done by either the enhanced PWM synchronization (SYNCMODE = 1) or the legacy PWM synchronization (SYNCMODE = 0). However, it is expected that the MOD register be synchronized only by the enhanced PWM synchronization.

In the case of enhanced PWM synchronization, the MOD register synchronization depends on SWWRBUF, SWRSTCNT, HWWRBUF, and HWRSTCNT bits according to this flowchart:

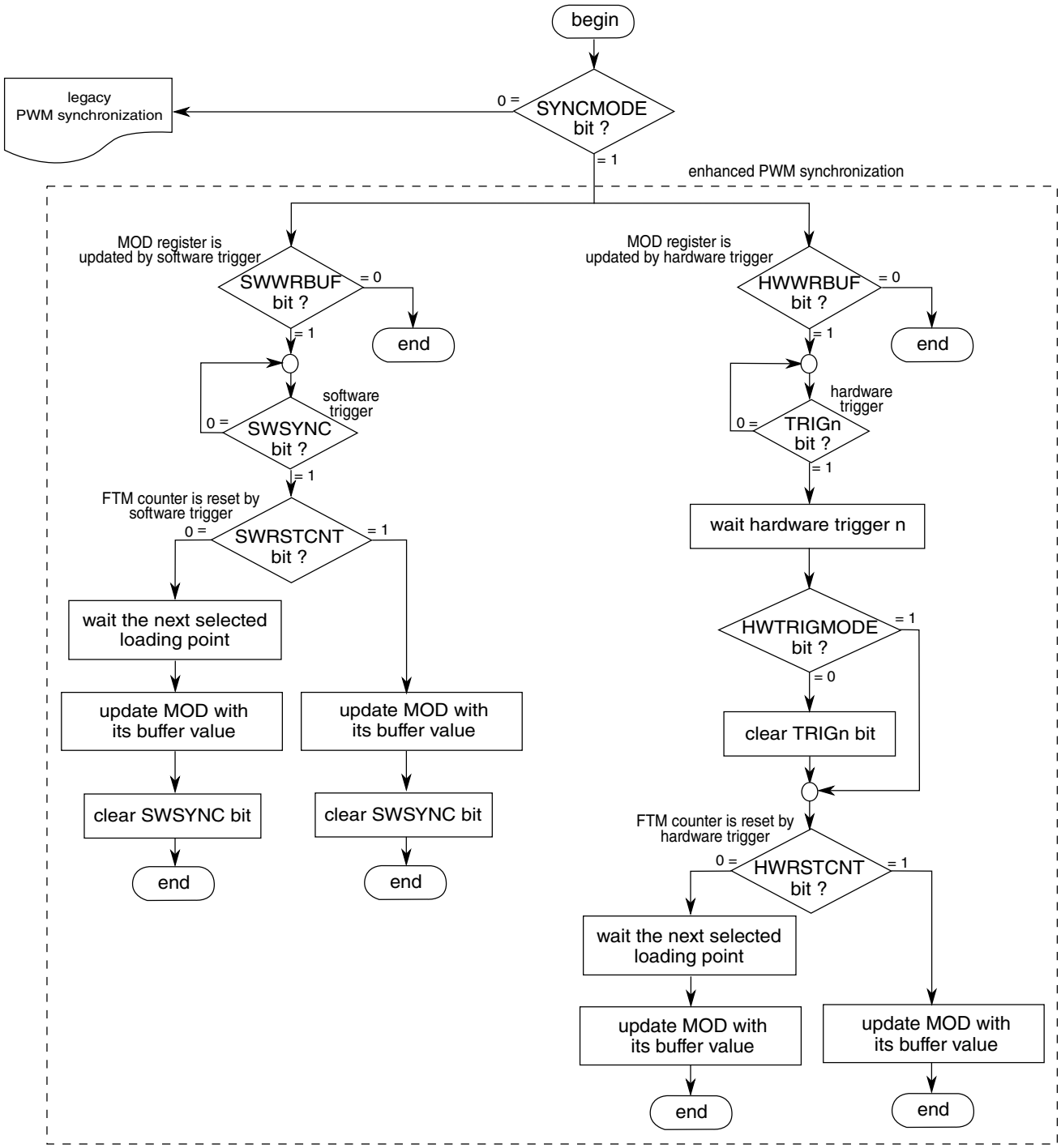


Figure 19-45. MOD register synchronization flowchart

In the case of legacy PWM synchronization, the MOD register synchronization depends on PWMSYNC and REINIT bits according to the following description.

If (SYNCMODE = 0), (PWMSYNC = 0), and (REINIT = 0), then this synchronization is made on the next selected loading point after an enabled trigger event takes place. If the trigger event was a software trigger, then the SWSYNC bit is cleared on the next selected

loading point. If the trigger event was a hardware trigger, then the trigger enable bit (TRIGn) is cleared according to Hardware trigger. Examples with software and hardware triggers follow.

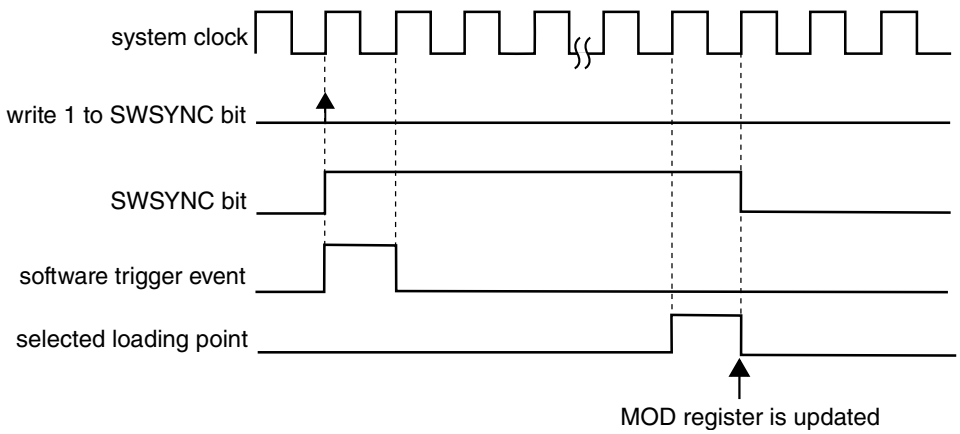


Figure 19-46. MOD synchronization with (SYNCMODE = 0), (PWMSYNC = 0), (REINIT = 0), and software trigger was used

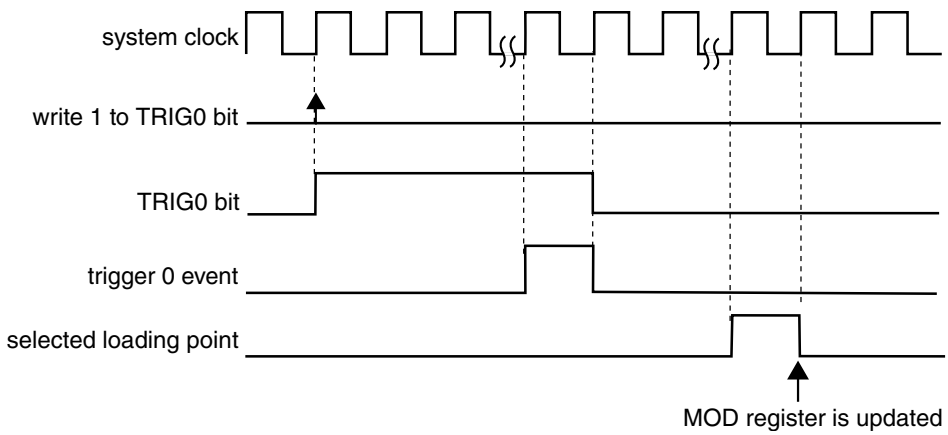


Figure 19-47. MOD synchronization with (SYNCMODE = 0), (HWTRIGMODE = 0), (PWMSYNC = 0), (REINIT = 0), and a hardware trigger was used

If (SYNCMODE = 0), (PWMSYNC = 0), and (REINIT = 1), then this synchronization is made on the next enabled trigger event. If the trigger event was a software trigger, then the SWSYNC bit is cleared according to the following example. If the trigger event was a hardware trigger, then the TRIGn bit is cleared according to Hardware trigger. Examples with software and hardware triggers follow.

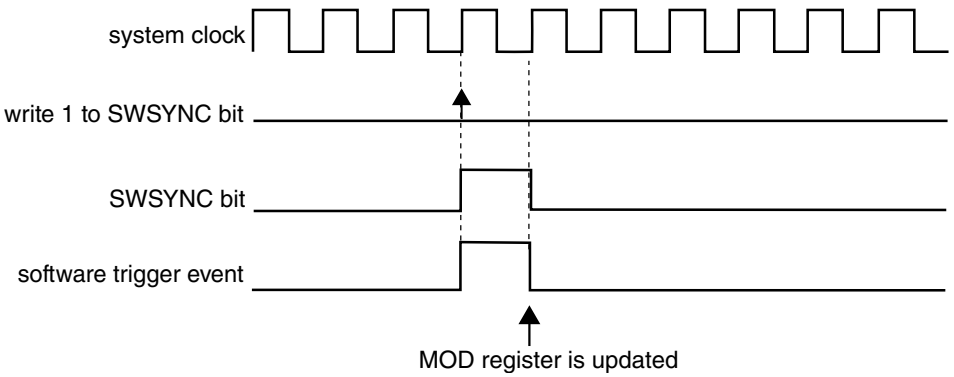


Figure 19-48. MOD synchronization with (SYNCMODE = 0), (PWMSYNC = 0), (REINIT = 1), and software trigger was used

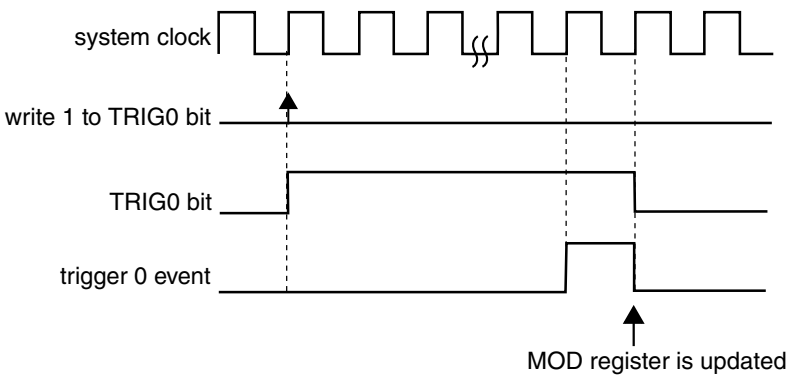


Figure 19-49. MOD synchronization with (SYNCMODE = 0), (HWTRIGMODE = 0), (PWMSYNC = 0), (REINIT = 1), and a hardware trigger was used

If (SYNCMODE = 0) and (PWMSYNC = 1), then this synchronization is made on the next selected loading point after the software trigger event takes place. The SWSYNC bit is cleared on the next selected loading point:

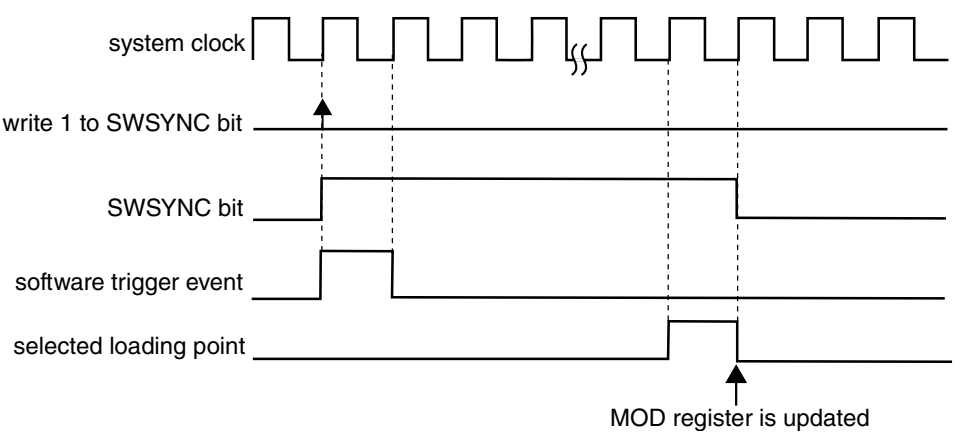


Figure 19-50. MOD synchronization with (SYNCMODE = 0) and (PWMSYNC = 1)

19.5.11.5 CNTIN register synchronization

The CNTIN register synchronization updates the CNTIN register with its buffer value.

This synchronization is enabled if (FTMEN = 1), (SYNCMODE = 1), and (CNTINC = 1). The CNTIN register synchronization can be done only by the enhanced PWM synchronization (SYNCMODE = 1). The synchronization mechanism is the same as the MOD register synchronization done by the enhanced PWM synchronization; see MOD register synchronization.

19.5.11.6 C(n)V and C(n+1)V register synchronization

The C(n)V and C(n+1)V registers synchronization updates the C(n)V and C(n+1)V registers with their buffer values.

This synchronization is enabled if (FTMEN = 1) and (SYNCEN = 1). The synchronization mechanism is the same as the MOD register synchronization. However, it is expected that the C(n)V and C(n+1)V registers be synchronized only by the enhanced PWM synchronization (SYNCMODE = 1).

19.5.11.7 OUTMASK register synchronization

The OUTMASK register synchronization updates the OUTMASK register with its buffer value.

The OUTMASK register can be updated at each rising edge of system clock (SYNCHOM = 0), by the enhanced PWM synchronization (SYNCHOM = 1 and SYNCMODE = 1) or by the legacy PWM synchronization (SYNCHOM = 1 and SYNCMODE = 0). However, it is expected that the OUTMASK register be synchronized only by the enhanced PWM synchronization.

In the case of enhanced PWM synchronization, the OUTMASK register synchronization depends on SWOM and HWOM bits. See the following flowchart:

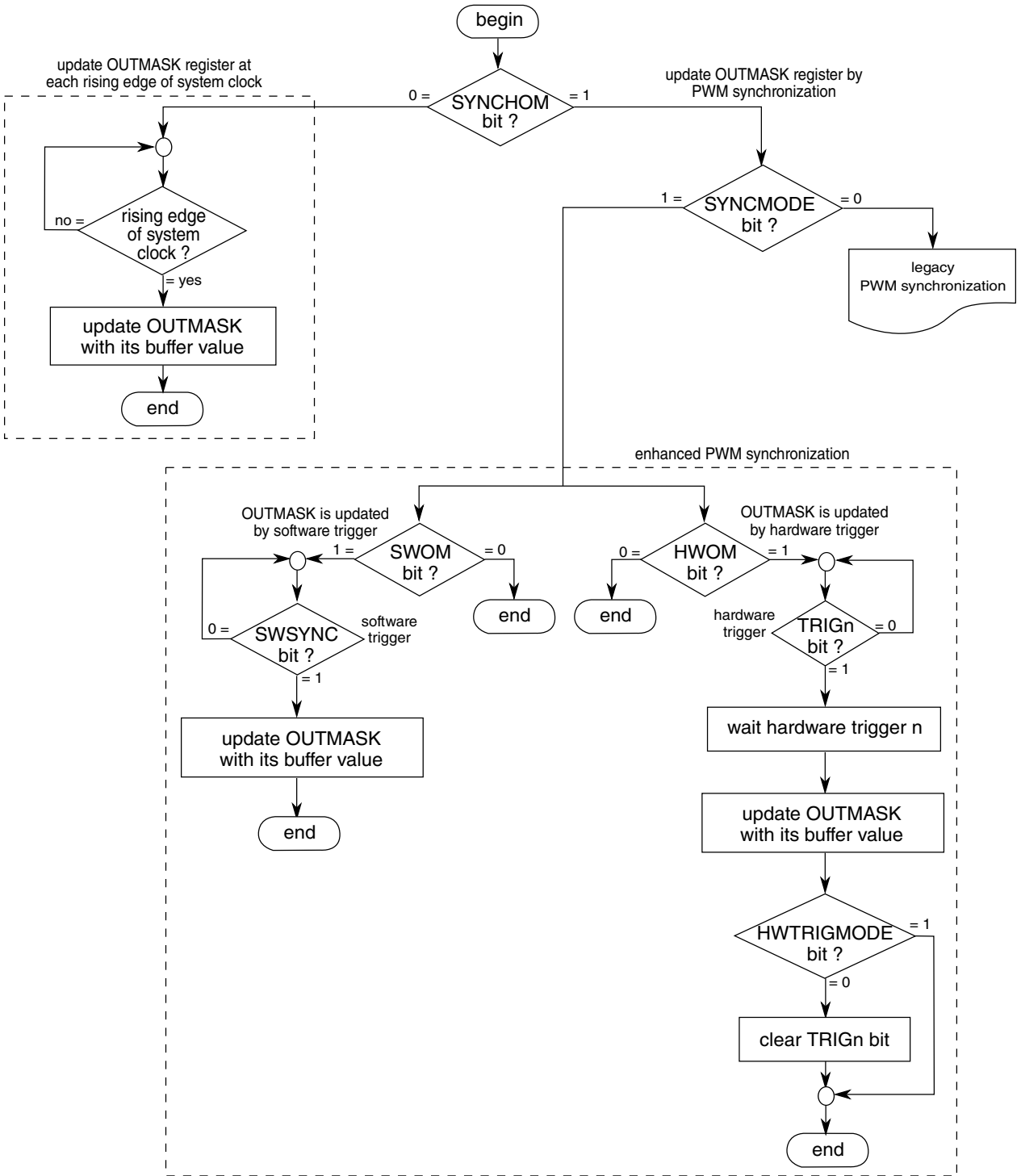


Figure 19-51. OUTMASK register synchronization flowchart

In the case of legacy PWM synchronization, the OUTMASK register synchronization depends on PWMSYNC bit according to the following description.

If (SYNCMODE = 0), (SYNCHOM = 1), and (PWMSYNC = 0), then this synchronization is done on the next enabled trigger event. If the trigger event was a software trigger, then the SWSYNC bit is cleared on the next selected loading point. If the trigger event was a hardware trigger, then the TRIGn bit is cleared according to Hardware trigger. Examples with software and hardware triggers follow.

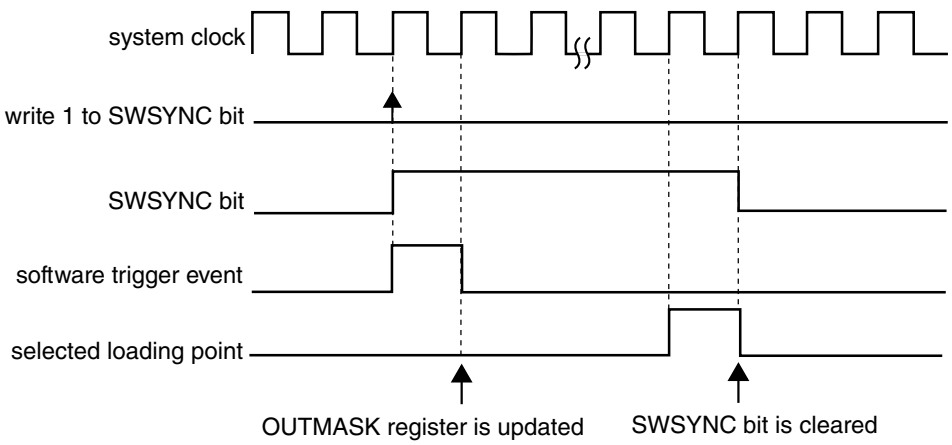


Figure 19-52. OUTMASK synchronization with (SYNCMODE = 0), (SYNCHOM = 1), (PWMSYNC = 0) and software trigger was used

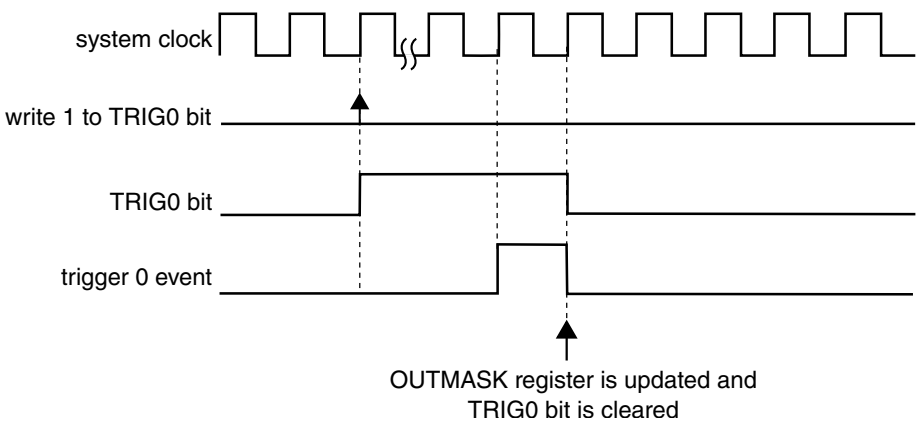


Figure 19-53. OUTMASK synchronization with (SYNCMODE = 0), (HWTRIGMODE = 0), (SYNCHOM = 1), (PWMSYNC = 0), and a hardware trigger was used

If (SYNCMODE = 0), (SYNCHOM = 1), and (PWMSYNC = 1), then this synchronization is made on the next enabled hardware trigger. The TRIGn bit is cleared according to Hardware trigger. An example with a hardware trigger follows.

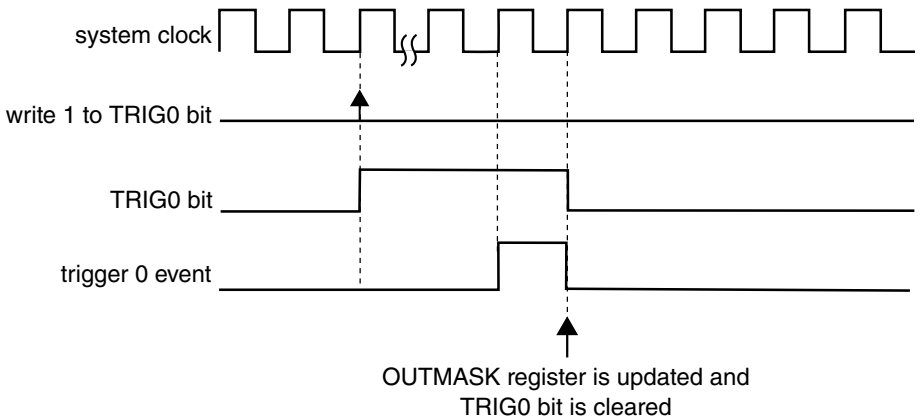


Figure 19-54. OUTMASK synchronization with (SYNCMODE = 0), (HWTRIGMODE = 0), (SYNCHOM = 1), (PWMSYNC = 1), and a hardware trigger was used

19.5.11.8 INVCTRL register synchronization

The INVCTRL register synchronization updates the INVCTRL register with its buffer value.

The INVCTRL register can be updated at each rising edge of system clock (INVC = 0) or by the enhanced PWM synchronization (INVC = 1 and SYNCMODE = 1) according to the following flowchart.

In the case of enhanced PWM synchronization, the INVCTRL register synchronization depends on SWINVC and HWINVC bits.

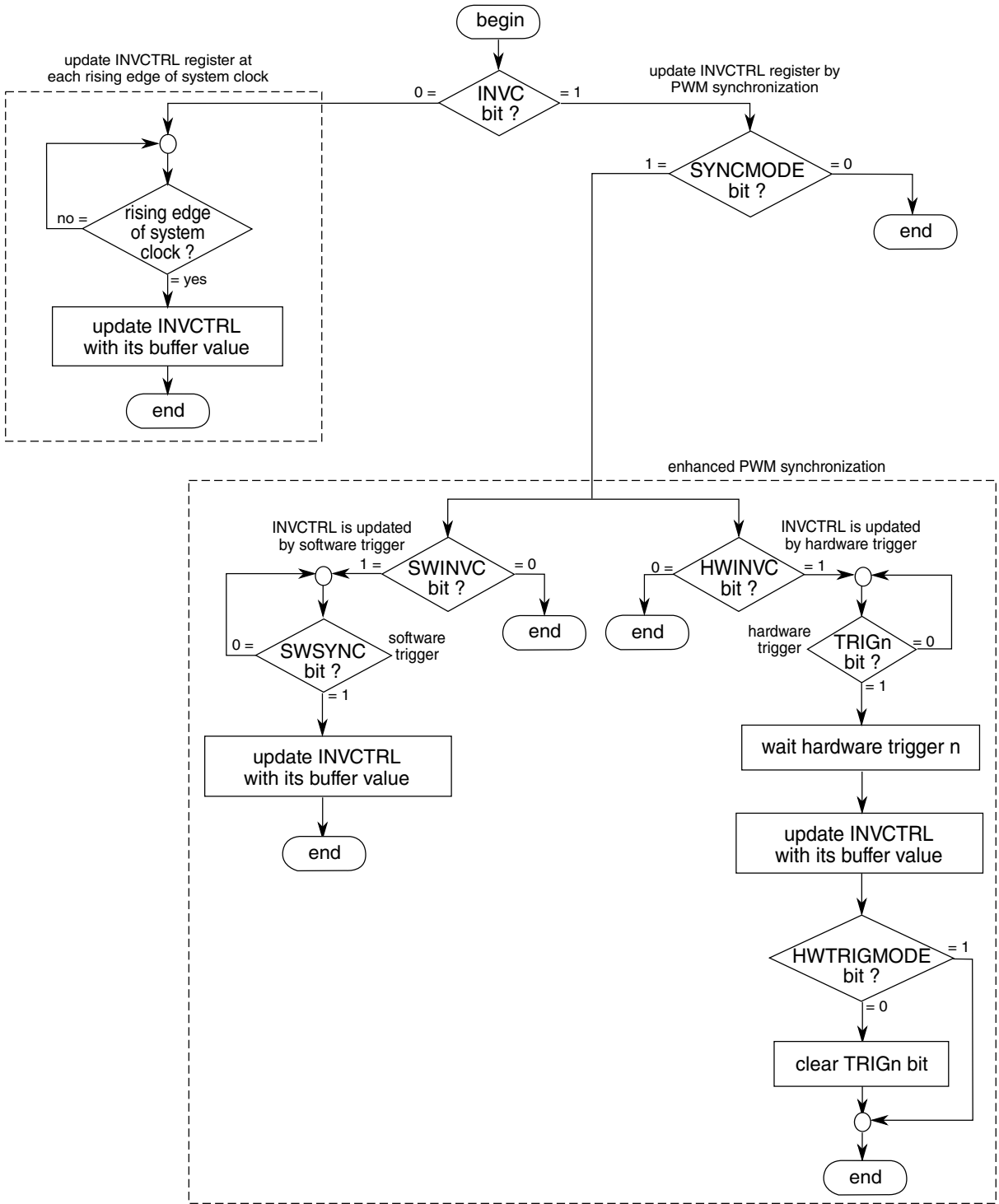


Figure 19-55. INVCTRL register synchronization flowchart

19.5.11.9 SWOCTRL register synchronization

The SWOCTRL register synchronization updates the SWOCTRL register with its buffer value.

The SWOCTRL register can be updated at each rising edge of system clock (SWOC = 0) or by the enhanced PWM synchronization (SWOC = 1 and SYNCMODE = 1) according to the following flowchart.

In the case of enhanced PWM synchronization, the SWOCTRL register synchronization depends on SWSOC and HWSOC bits.

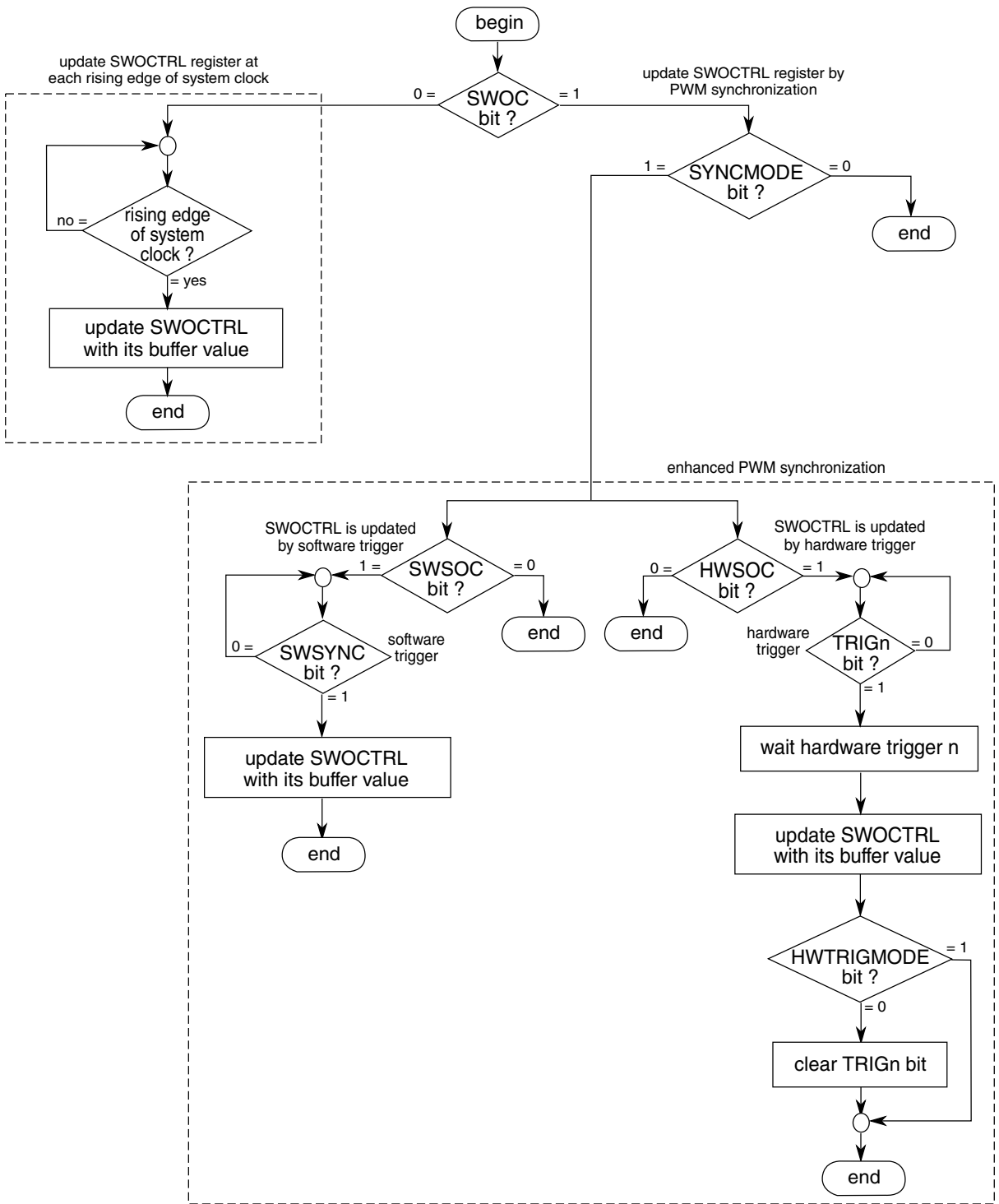


Figure 19-56. SWOCTRL register synchronization flowchart

19.5.11.10 FTM counter synchronization

The FTM counter synchronization is a mechanism that allows the FTM to restart the PWM generation at a certain point in the PWM period. The channels outputs are forced to their initial value, except for channels in Output Compare mode, and the FTM counter is forced to its initial counting value defined by CNTIN register.

The following figure shows the FTM counter synchronization. Note that after the synchronization event occurs, the channel (n) is set to its initial value and the channel (n +1) is not set to its initial value due to a specific timing of this figure in which the deadtime insertion prevents this channel output from transitioning to 1. If no deadtime insertion is selected, then the channel (n+1) transitions to logical value 1 immediately after the synchronization event occurs.

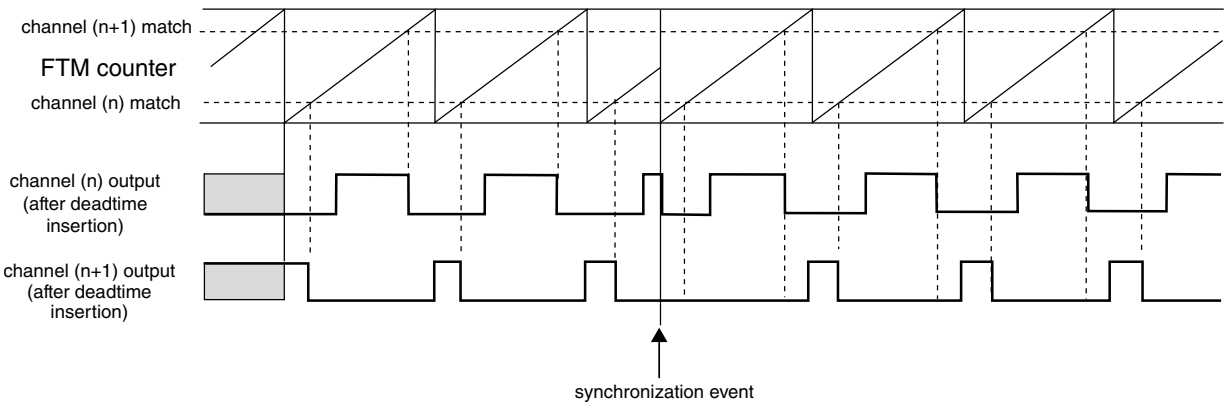


Figure 19-57. FTM counter synchronization

The FTM counter synchronization can be done by either the enhanced PWM synchronization (SYNCMODE = 1) or the legacy PWM synchronization (SYNCMODE = 0). However, the FTM counter must be synchronized only by the enhanced PWM synchronization.

In the case of enhanced PWM synchronization, the FTM counter synchronization depends on SWRSTCNT and HWRSTCNT bits according to the following flowchart.

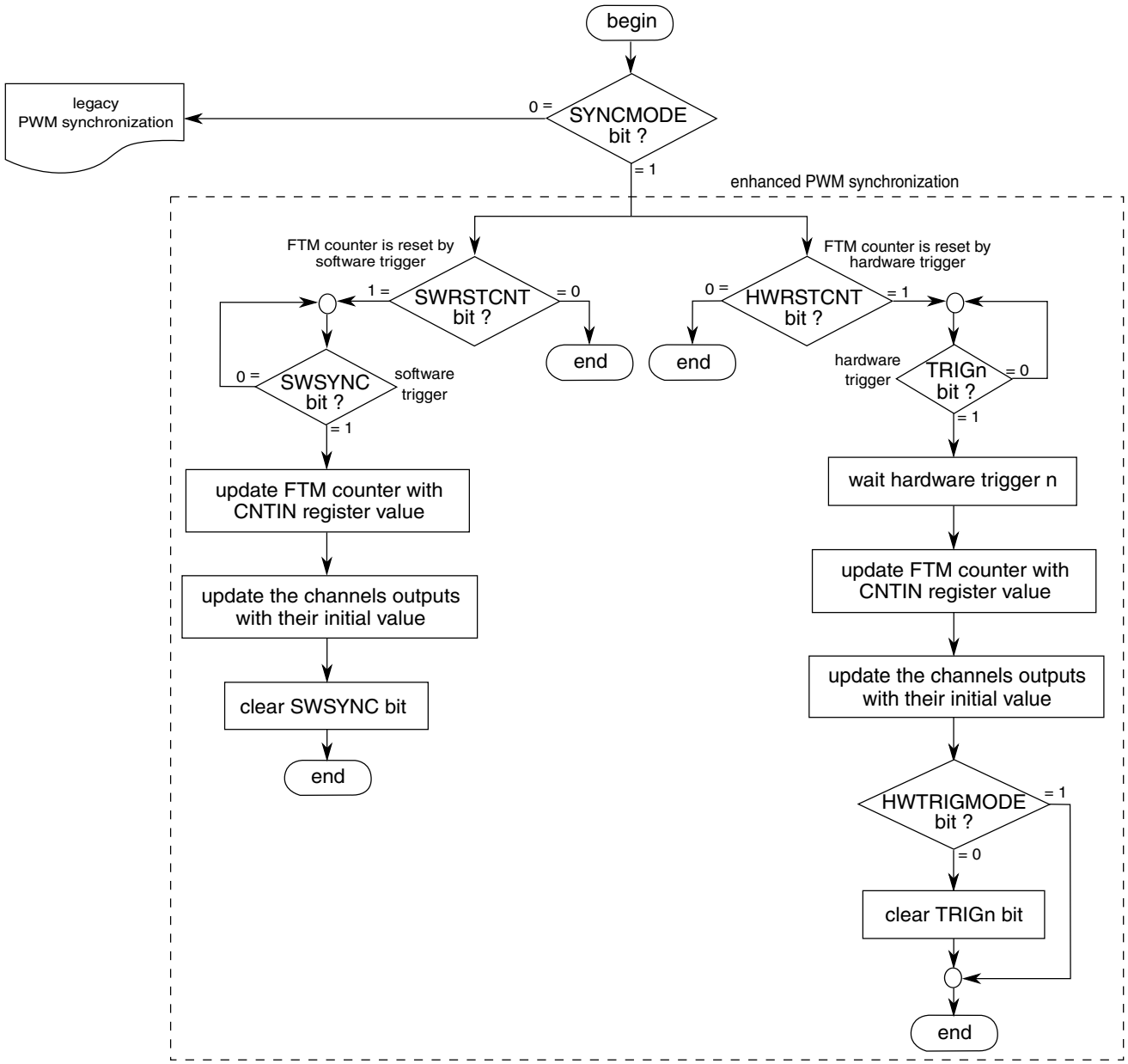


Figure 19-58. FTM counter synchronization flowchart

In the case of legacy PWM synchronization, the FTM counter synchronization depends on REINIT and PWMSYNC bits according to the following description.

If (SYNCMODE = 0), (REINIT = 1), and (PWMSYNC = 0) then this synchronization is made on the next enabled trigger event. If the trigger event was a software trigger then the SWSYNC bit is cleared according to the following example. If the trigger event was a hardware trigger then the TRIGn bit is cleared according to Hardware trigger. Examples with software and hardware triggers follow.

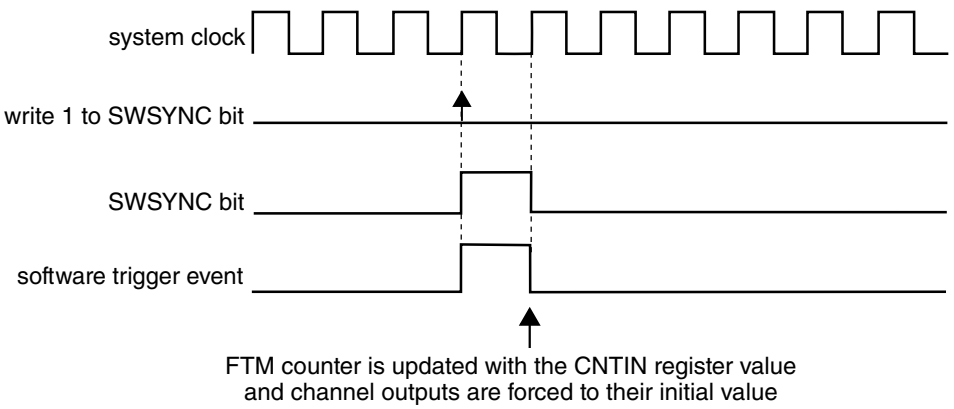


Figure 19-59. FTM counter synchronization with (SYNCMODE = 0), (REINIT = 1), (PWMSYNC = 0), and software trigger was used

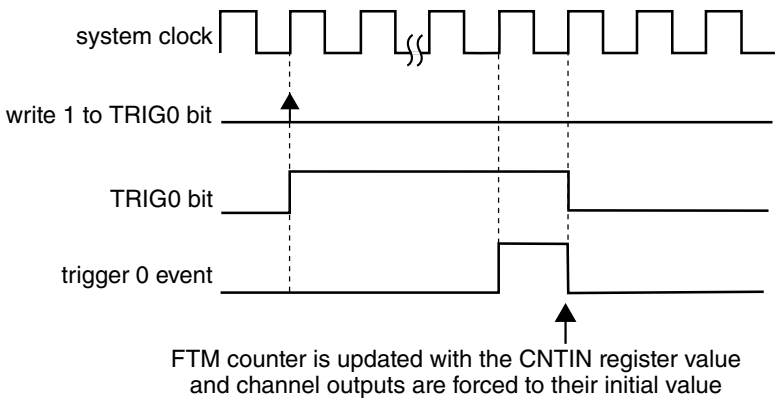


Figure 19-60. FTM counter synchronization with (SYNCMODE = 0), (HWTRIGMODE = 0), (REINIT = 1), (PWMSYNC = 0), and a hardware trigger was used

If (SYNCMODE = 0), (REINIT = 1), and (PWMSYNC = 1) then this synchronization is made on the next enabled hardware trigger. The TRIGn bit is cleared according to Hardware trigger.

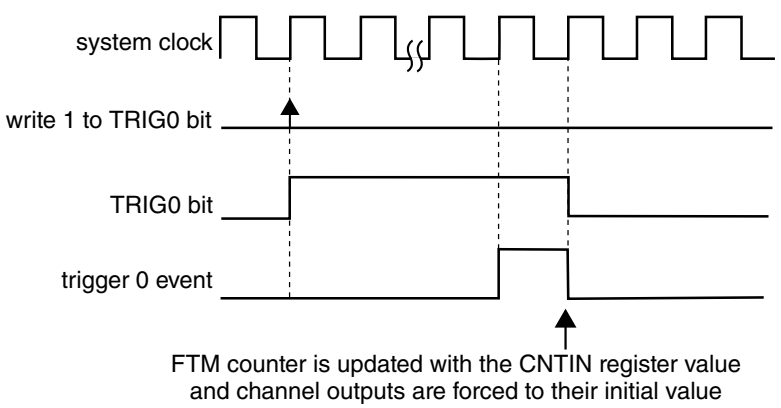


Figure 19-61. FTM counter synchronization with (SYNCMODE = 0), (HWTRIGMODE = 0), (REINIT = 1), (PWMSYNC = 1), and a hardware trigger was used

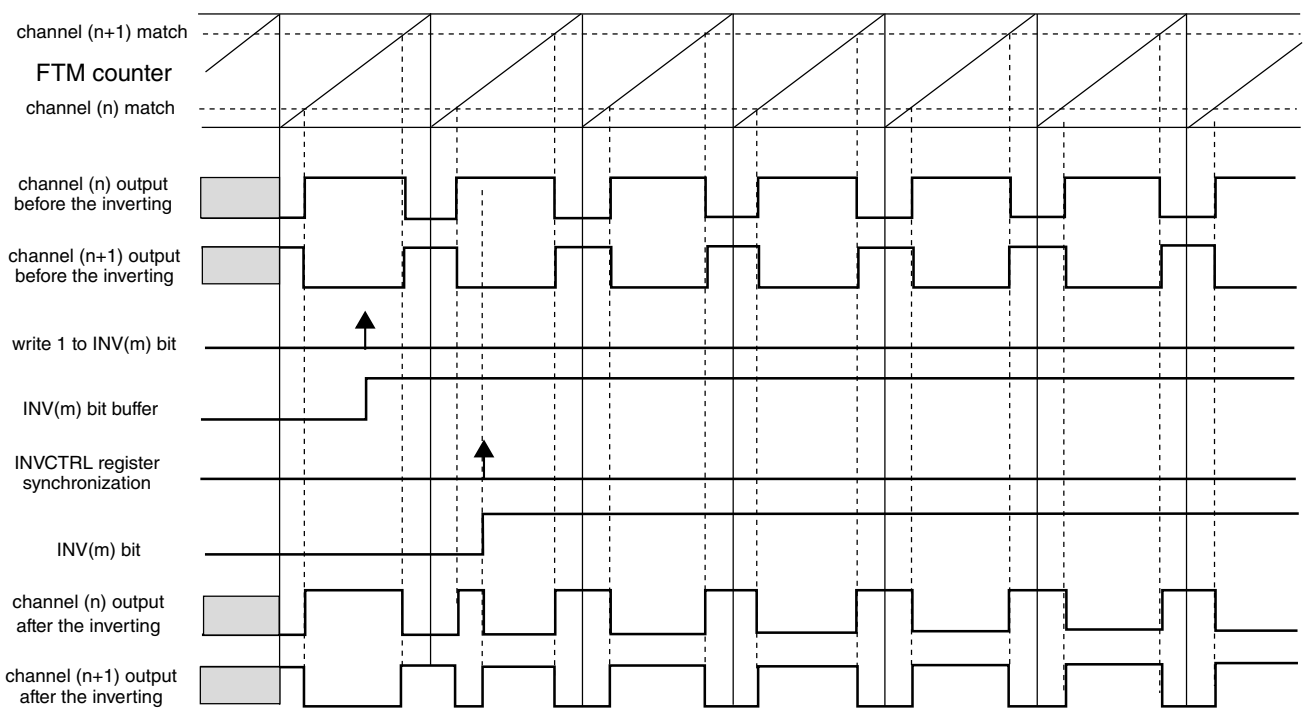
19.5.12 Inverting

The invert functionality swaps the signals between channel (n) and channel (n+1) outputs. The inverting operation is selected when:

- QUADEN = 0
- DECAPEN = 0
- COMP = 1, and
- INVm = 1 (where m represents a channel pair)

The INVm bit in INVCTRL register is updated with its buffer value according to INVCTRL register synchronization

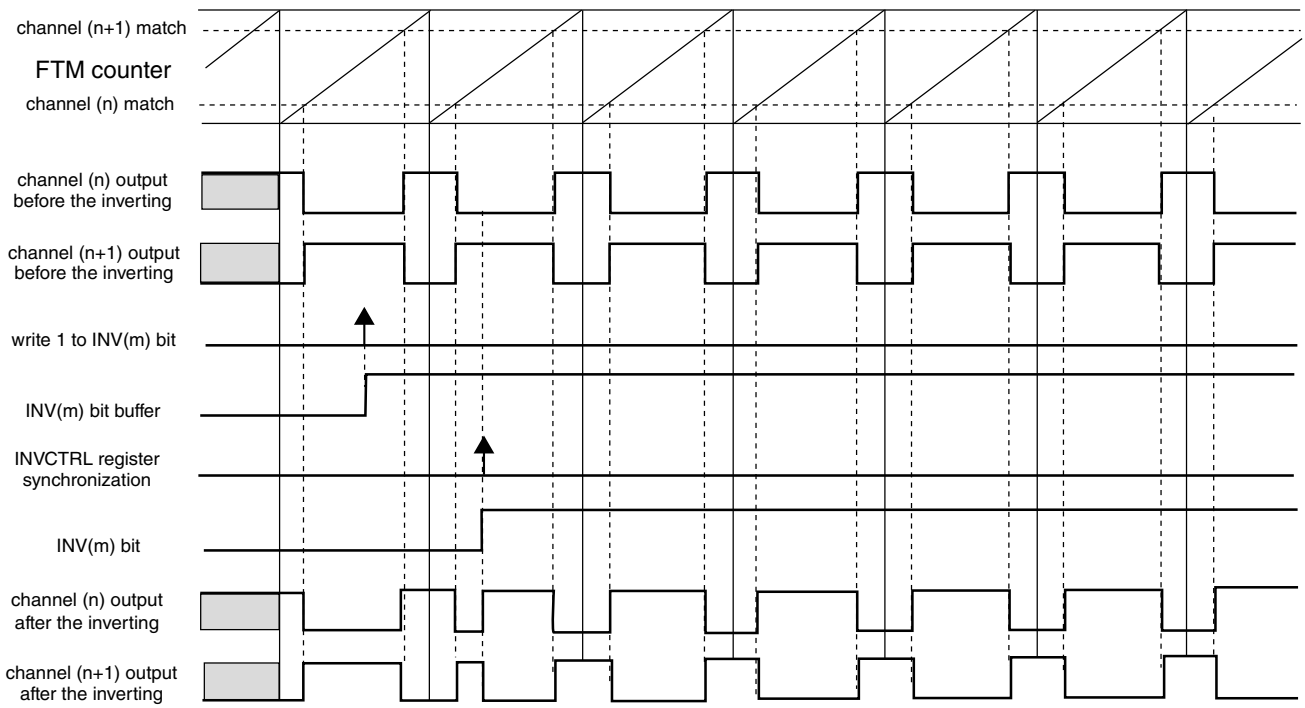
In High-True (ELSnB:ELSnA = 1:0) Combine mode, the channel (n) output is forced low at the beginning of the period (FTM counter = CNTIN), forced high at the channel (n) match and forced low at the channel (n+1) match. If the inverting is selected, the channel (n) output behavior is changed to force high at the beginning of the PWM period, force low at the channel (n) match and force high at the channel (n+1) match. See the following figure.



NOTE
INV(m) bit selects the inverting to the pair channels (n) and (n+1).

Figure 19-62. Channels (n) and (n+1) outputs after the inverting in High-True (ELSnB:ELSnA = 1:0) Combine mode

Note that the ELSnB:ELSnA bits value should be considered because they define the active state of the channels outputs. In Low-True (ELSnB:ELSnA = X:1) Combine mode, the channel (n) output is forced high at the beginning of the period, forced low at the channel (n) match and forced high at the channel (n+1) match. When inverting is selected, the channels (n) and (n+1) present waveforms as shown in the following figure.



NOTE
INV(m) bit selects the inverting to the pair channels (n) and (n+1).

Figure 19-63. Channels (n) and (n+1) outputs after the inverting in Low-True (ELSnB:ELSnA = X:1) Combine mode

Note

The inverting feature is not available in Output Compare mode.

19.5.13 Software output control

The software output control forces the channel output according to software defined values at a specific time in the PWM generation.

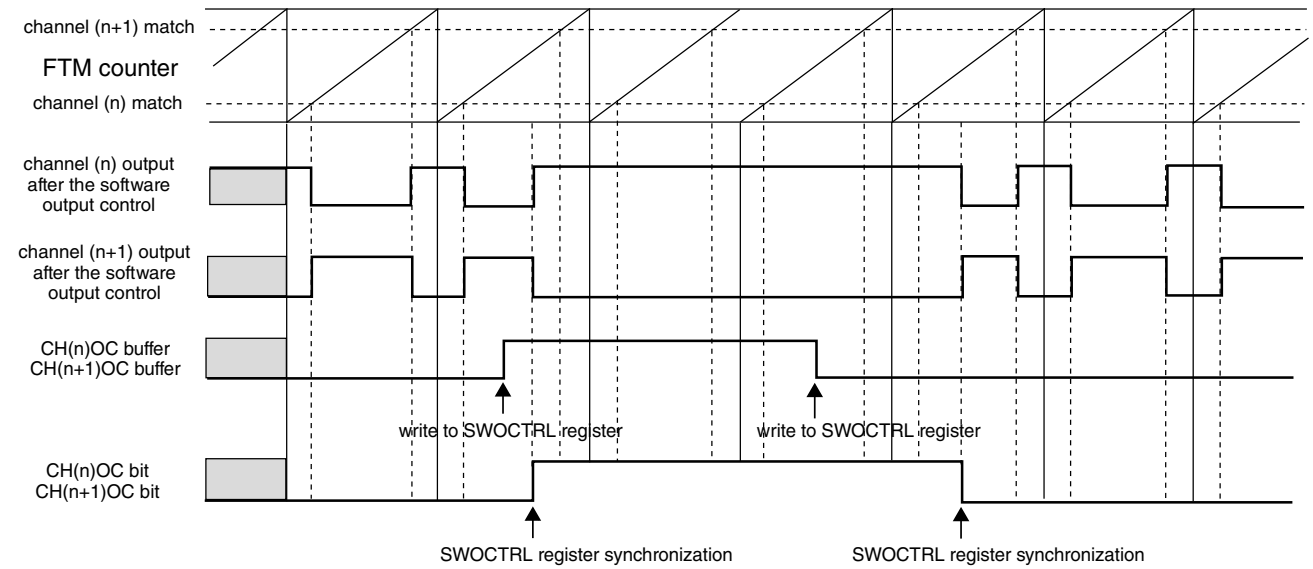
The software output control is selected when:

- QUADEN = 0
- DECAPEN = 0, and
- CHnOC = 1

The CHnOC bit enables the software output control for a specific channel output and the CHnOCV selects the value that is forced to this channel output.

Both CHnOC and CHnOCV bits in SWOCTRL register are buffered and updated with their buffer value according to SWOCTRL register synchronization.

The following figure shows the channels (n) and (n+1) outputs signals when the software output control is used. In this case the channels (n) and (n+1) are set to Combine and Complementary mode.



NOTE
CH(n)OCV = 1 and CH(n+1)OCV = 0.

Figure 19-64. Example of software output control in Combine and Complementary mode

Software output control forces the following values on channels (n) and (n+1) when the COMP bit is zero.

Table 19-12. Software ouput control behavior when (COMP = 0)

CH(n)OC	CH(n+1)OC	CH(n)OCV	CH(n+1)OCV	Channel (n) Output	Channel (n+1) Output
0	0	X	X	is not modified by SWOC	is not modified by SWOC
1	1	0	0	is forced to zero	is forced to zero
1	1	0	1	is forced to zero	is forced to one
1	1	1	0	is forced to one	is forced to zero
1	1	1	1	is forced to one	is forced to one

Software output control forces the following values on channels (n) and (n+1) when the COMP bit is one.

Table 19-13. Software ouput control behavior when (COMP = 1)

CH(n)OC	CH(n+1)OC	CH(n)OCV	CH(n+1)OCV	Channel (n) Output	Channel (n+1) Output
0	0	X	X	is not modified by SWOC	is not modified by SWOC
1	1	0	0	is forced to zero	is forced to zero
1	1	0	1	is forced to zero	is forced to one
1	1	1	0	is forced to one	is forced to zero
1	1	1	1	is forced to one	is forced to zero

Note

- The CH(n)OC and CH(n+1)OC bits should be equal.
- The COMP bit must not be modified when software output control is enabled, that is, CH(n)OC = 1 and/or CH(n +1)OC = 1.
- Software output control has the same behavior with disabled or enabled FTM counter (see the CLKS field description in the Status and Control register).

19.5.14 Deadtime insertion

The deadtime insertion is enabled when (DTEN = 1) and (DTVAL[5:0] is non- zero).

DEADTIME register defines the deadtime delay that can be used for all FTM channels. The DTPS[1:0] bits define the prescaler for the system clock and the DTVAL[5:0] bits define the deadtime modulo, that is, the number of the deadtime prescaler clocks.

The deadtime delay insertion ensures that no two complementary signals (channels (n) and (n+1)) drive the active state at the same time.

If POL(n) = 0, POL(n+1) = 0, and the deadtime is enabled, then when the channel (n) match (FTM counter = C(n)V) occurs, the channel (n) output remains at the low value until the end of the deadtime delay when the channel (n) output is set. Similarly, when the channel (n+1) match (FTM counter = C(n+1)V) occurs, the channel (n+1) output remains at the low value until the end of the deadtime delay when the channel (n+1) output is set. See the following figures.

If POL(n) = 1, POL(n+1) = 1, and the deadtime is enabled, then when the channel (n) match (FTM counter = C(n)V) occurs, the channel (n) output remains at the high value until the end of the deadtime delay when the channel (n) output is cleared. Similarly,

when the channel (n+1) match (FTM counter = C(n+1)V) occurs, the channel (n+1) output remains at the high value until the end of the deadtime delay when the channel (n +1) output is cleared.

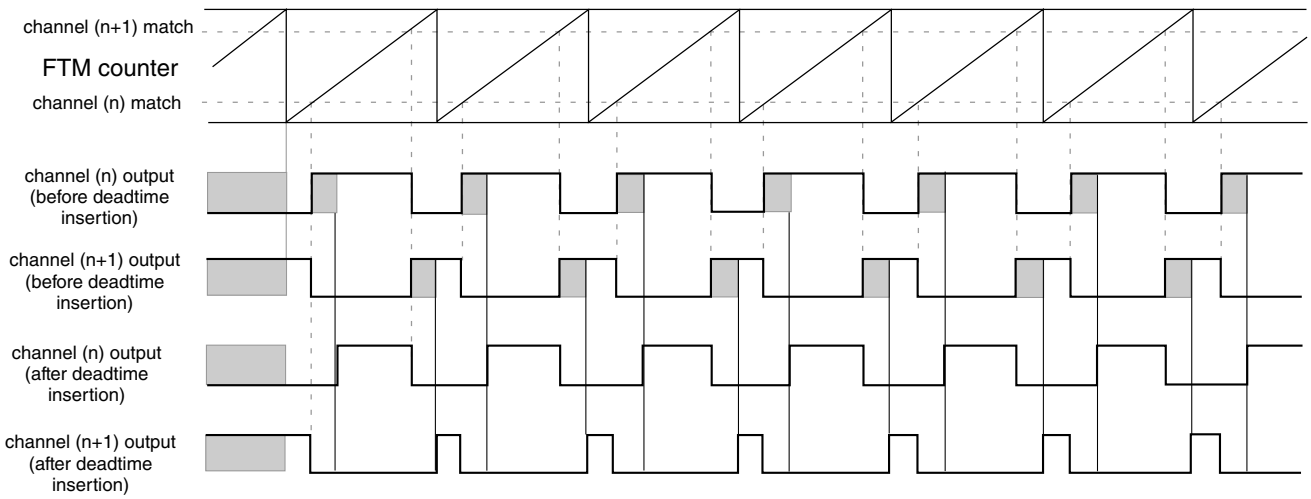


Figure 19-65. Deadtime insertion with ELSnB:ELSnA = 1:0, POL(n) = 0, and POL(n+1) = 0

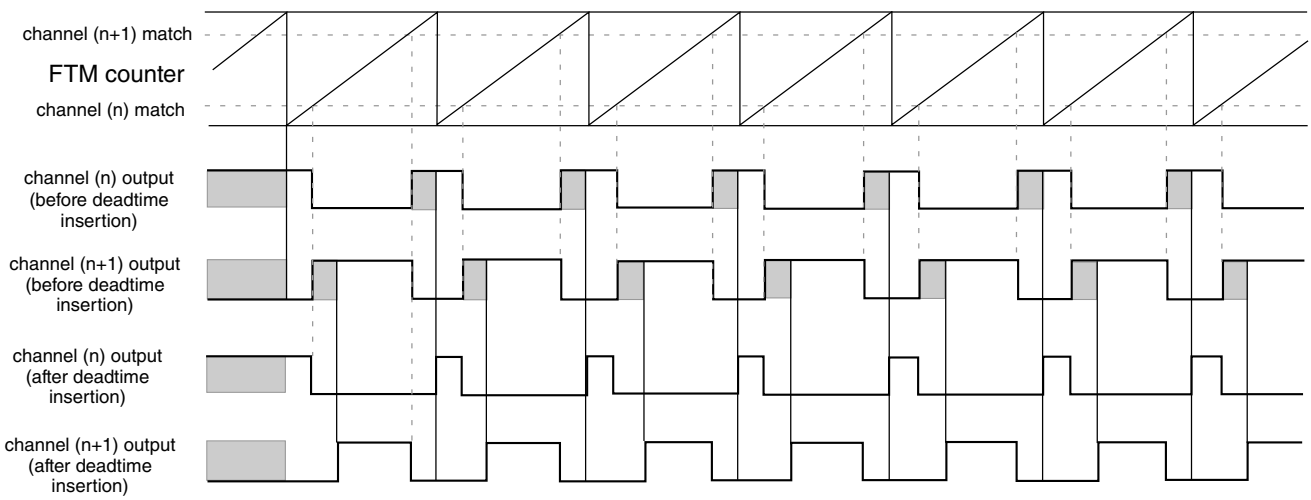


Figure 19-66. Deadtime insertion with ELSnB:ELSnA = X:1, POL(n) = 0, and POL(n+1) = 0

NOTE

- The deadtime feature must be used only in Complementary mode.
- The deadtime feature is not available in Output Compare mode.

19.5.14.1 Deadtime insertion corner cases

If (PS[2:0] is cleared), (DTPS[1:0] = 0:0 or DTPS[1:0] = 0:1):

- and the deadtime delay is greater than or equal to the channel (n) duty cycle ((C(n +1)V – C(n)V) × system clock), then the channel (n) output is always the inactive value (POL(n) bit value).
- and the deadtime delay is greater than or equal to the channel (n+1) duty cycle ((MOD – CNTIN + 1 – (C(n+1)V – C(n)V)) × system clock), then the channel (n+1) output is always the inactive value (POL(n+1) bit value).

Although, in most cases the deadtime delay is not comparable to channels (n) and (n+1) duty cycle, the following figures show examples where the deadtime delay is comparable to the duty cycle.

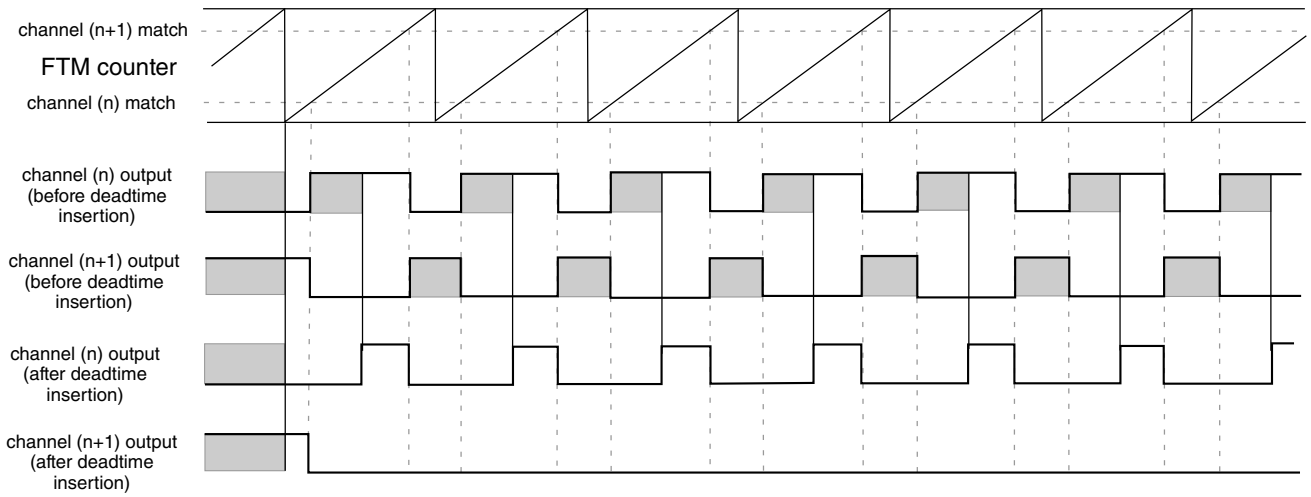


Figure 19-67. Example of the deadtime insertion (ELSnB:ELSnA = 1:0, POL(n) = 0, and POL(n+1) = 0) when the deadtime delay is comparable to channel (n+1) duty cycle

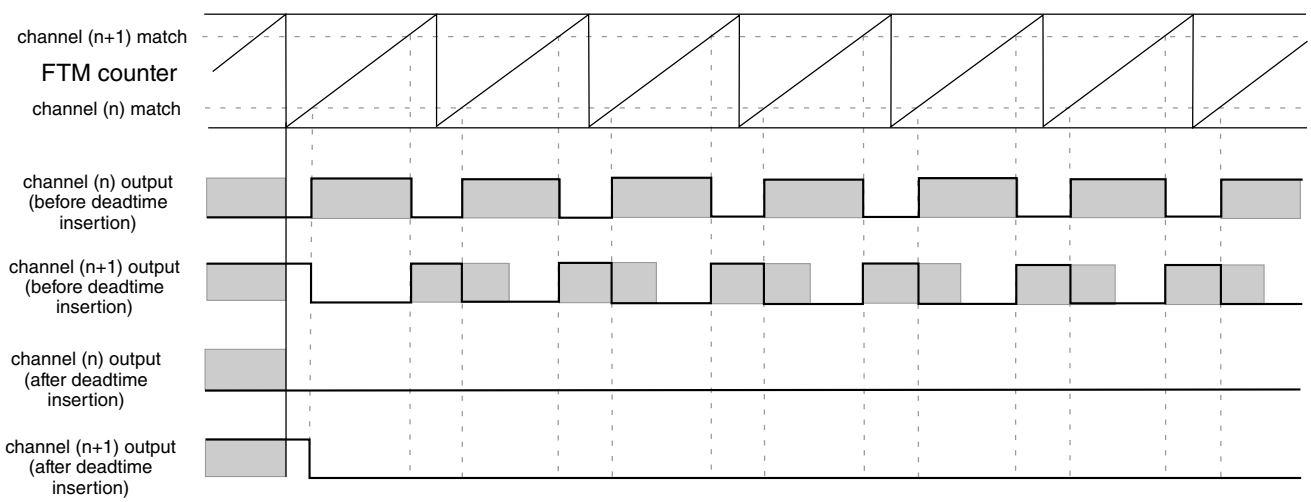


Figure 19-68. Example of the deadtime insertion (ELSnB:ELSnA = 1:0, POL(n) = 0, and POL(n+1) = 0) when the deadtime delay is comparable to channels (n) and (n+1) duty cycle

19.5.15 Output mask

The output mask can be used to force channels output to their inactive state through software. For example: to control a BLDC motor.

Any write to the OUTMASK register updates its write buffer. The OUTMASK register is updated with its buffer value by PWM synchronization; see OUTMASK register synchronization.

If CHnOM = 1, then the channel (n) output is forced to its inactive state (POLn bit value). If CHnOM = 0, then the channel (n) output is unaffected by the output mask. See the following figure.

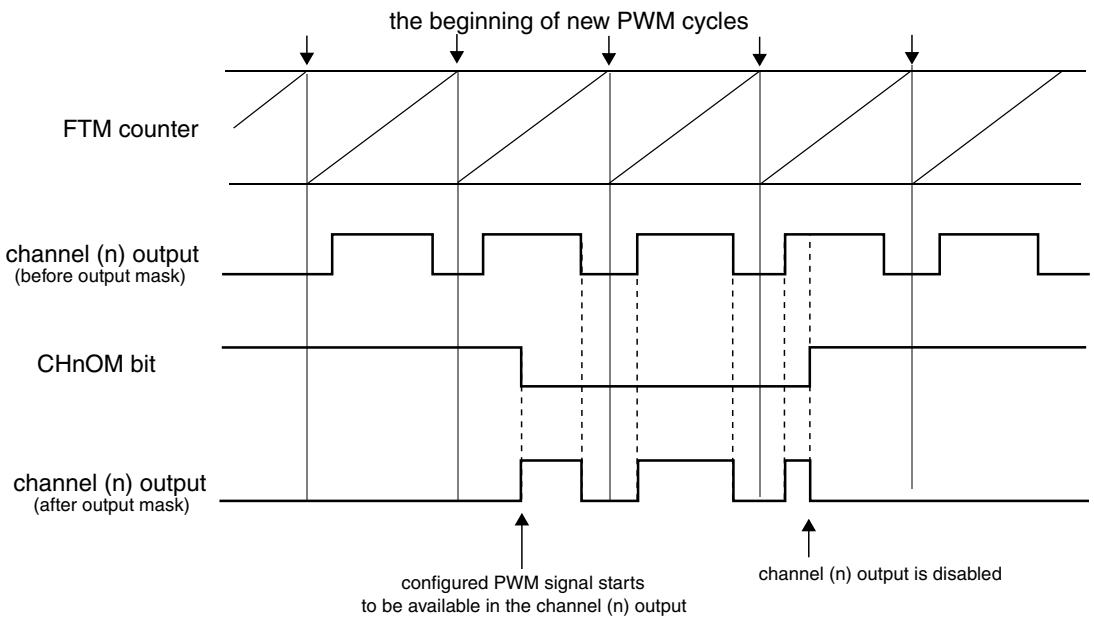


Figure 19-69. Output mask with POLn = 0

The following table shows the output mask result before the polarity control.

Table 19-14. Output mask result for channel (n) before the polarity control

CHnOM	Output Mask Input	Output Mask Result
0	inactive state	inactive state
	active state	active state
1	inactive state	inactive state
	active state	

19.5.16 Fault control

The fault control is enabled if (FAULTM[1:0] ≠ 0:0).

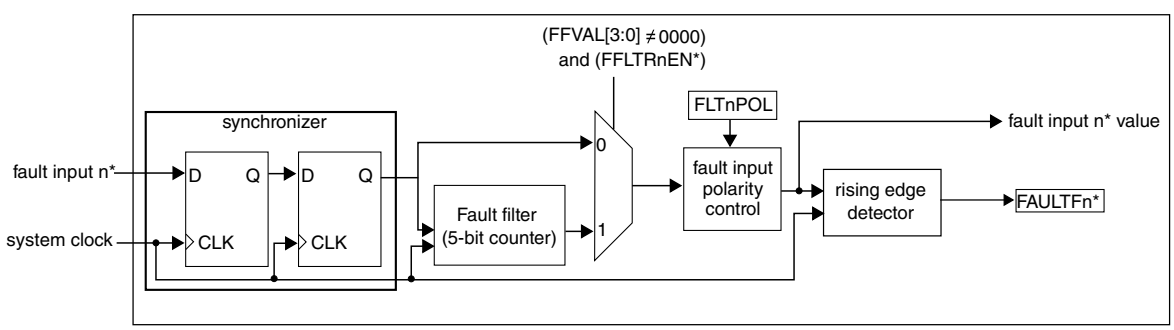
FTM can have up to four fault inputs. FAULTnEN bit (where n = 0, 1, 2, 3) enables the fault input n and FFLTRnEN bit enables the fault input n filter. FFVAL[3:0] bits select the value of the enabled filter in each enabled fault input.

First, each fault input signal is synchronized by the system clock; see the synchronizer block in the following figure. Following synchronization, the fault input n signal enters the filter block. When there is a state change in the fault input n signal, the 5-bit counter is reset and starts counting up. As long as the new state is stable on the fault input n, the counter continues to increment. If the 5-bit counter overflows, that is, the counter exceeds the value of the FFVAL[3:0] bits, the new fault input n value is validated. It is then transmitted as a pulse edge to the edge detector.

If the opposite edge appears on the fault input n signal before validation (counter overflow), the counter is reset. At the next input transition, the counter starts counting again. Any pulse that is shorter than the minimum value selected by FFVAL[3:0] bits (× system clock) is regarded as a glitch and is not passed on to the edge detector.

The fault input n filter is disabled when the FFVAL[3:0] bits are zero or when FAULTnEN = 0. In this case, the fault input n signal is delayed 2 rising edges of the system clock and the FAULTFn bit is set on 3th rising edge of the system clock after a rising edge occurs on the fault input n.

If FFVAL[3:0] ≠ 0000 and FAULTnEN = 1, then the fault input n signal is delayed (3 + FFVAL[3:0]) rising edges of the system clock, that is, the FAULTFn bit is set (4 + FFVAL[3:0]) rising edges of the system clock after a rising edge occurs on the fault input n.



* where n = 3, 2, 1, 0

Figure 19-70. Fault input n control block diagram

If the fault control and fault input n are enabled and a rising edge at the fault input n signal is detected, a fault condition has occurred and the FAULTFn bit is set. The FAULTF bit is the logic OR of FAULTFn[3:0] bits. See the following figure.

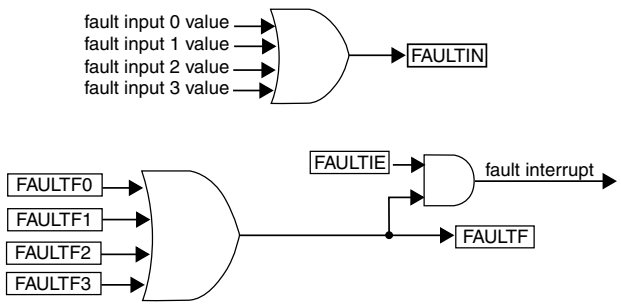


Figure 19-71. FAULTF and FAULTIN bits and fault interrupt

If the fault control is enabled (FAULTM[1:0] ≠ 0:0), a fault condition has occurred and (FAULTEN = 1), then outputs are forced to their safe values:

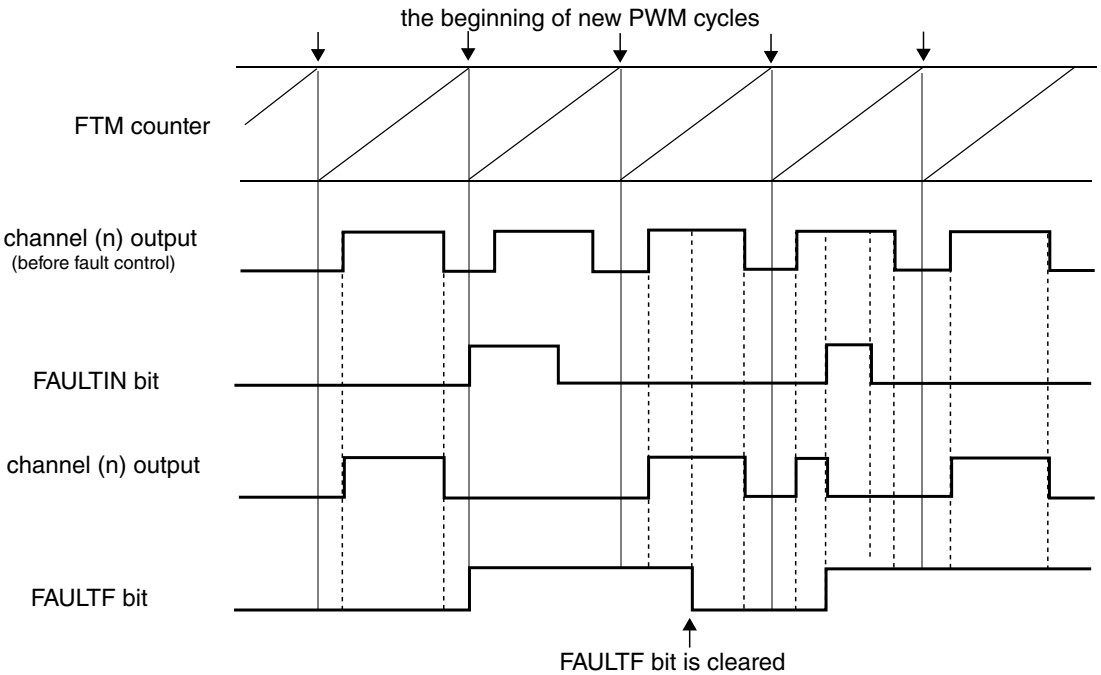
- Channel (n) output takes the value of POL(n)
- Channel (n+1) takes the value of POL(n+1)

The fault interrupt is generated when (FAULTF = 1) and (FAULTIE = 1). This interrupt request remains set until:

- Software clears the FAULTF bit by reading FAULTF bit as 1 and writing 0 to it
- Software clears the FAULTIE bit
- A reset occurs

19.5.16.1 Automatic fault clearing

If the automatic fault clearing is selected (FAULTM[1:0] = 1:1), then the channels output disabled by fault control is again enabled when the fault input signal (FAULTIN) returns to zero and a new PWM cycle begins. See the following figure.

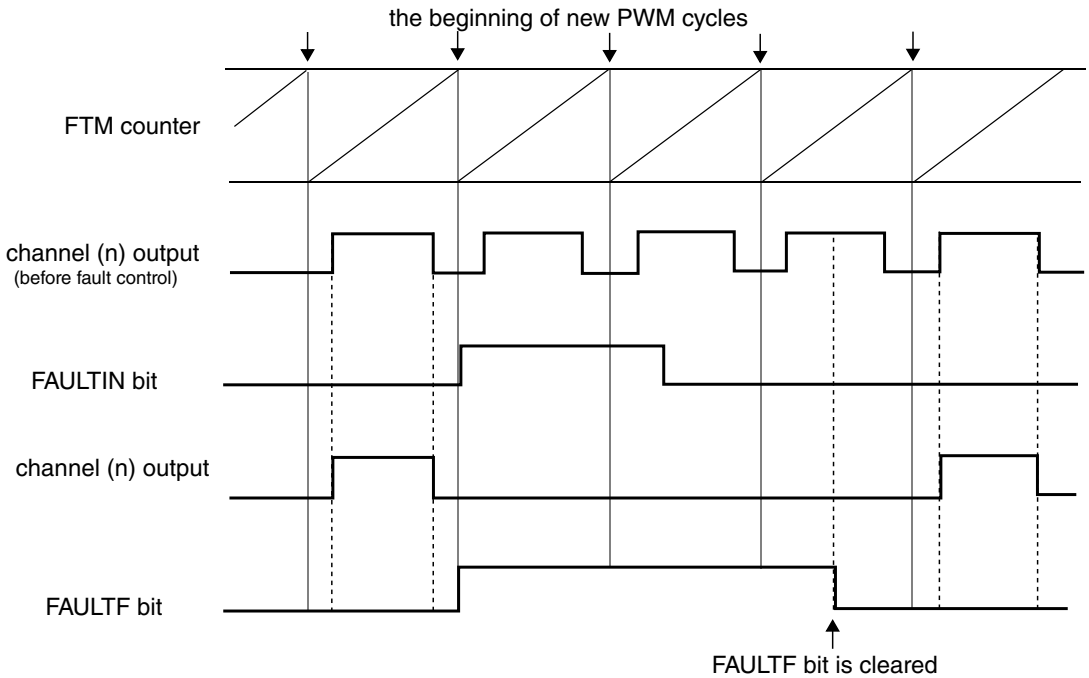


NOTE
The channel (n) output is after the fault control with automatic fault clearing and POLn = 0.

Figure 19-72. Fault control with automatic fault clearing

19.5.16.2 Manual fault clearing

If the manual fault clearing is selected (FAULTM[1:0] = 0:1 or 1:0), then the channels output disabled by fault control is again enabled when the FAULTF bit is cleared and a new PWM cycle begins. See the following figure.



NOTE
The channel (n) output is after the fault control with manual fault clearing and POLn = 0.

Figure 19-73. Fault control with manual fault clearing

19.5.16.3 Fault inputs polarity control

The FLTjPOL bit selects the fault input j polarity, where j = 0, 1, 2, 3:

- If FLTjPOL = 0, the fault j input polarity is high, so the logical one at the fault input j indicates a fault.
- If FLTjPOL = 1, the fault j input polarity is low, so the logical zero at the fault input j indicates a fault.

19.5.17 Polarity control

The POLn bit selects the channel (n) output polarity:

- If POLn = 0, the channel (n) output polarity is high, so the logical one is the active state and the logical zero is the inactive state.
- If POLn = 1, the channel (n) output polarity is low, so the logical zero is the active state and the logical one is the inactive state.

19.5.18 Initialization

The initialization forces the CHnOI bit value to the channel (n) output when a one is written to the INIT bit.

The initialization depends on COMP and DTEN bits. The following table shows the values that channels (n) and (n+1) are forced by initialization when the COMP and DTEN bits are zero.

Table 19-15. Initialization behavior when (COMP = 0 and DTEN = 0)

CH(n)OI	CH(n+1)OI	Channel (n) Output	Channel (n+1) Output
0	0	is forced to zero	is forced to zero
0	1	is forced to zero	is forced to one
1	0	is forced to one	is forced to zero
1	1	is forced to one	is forced to one

The following table shows the values that channels (n) and (n+1) are forced by initialization when (COMP = 1) or (DTEN = 1).

Table 19-16. Initialization behavior when (COMP = 1 or DTEN = 1)

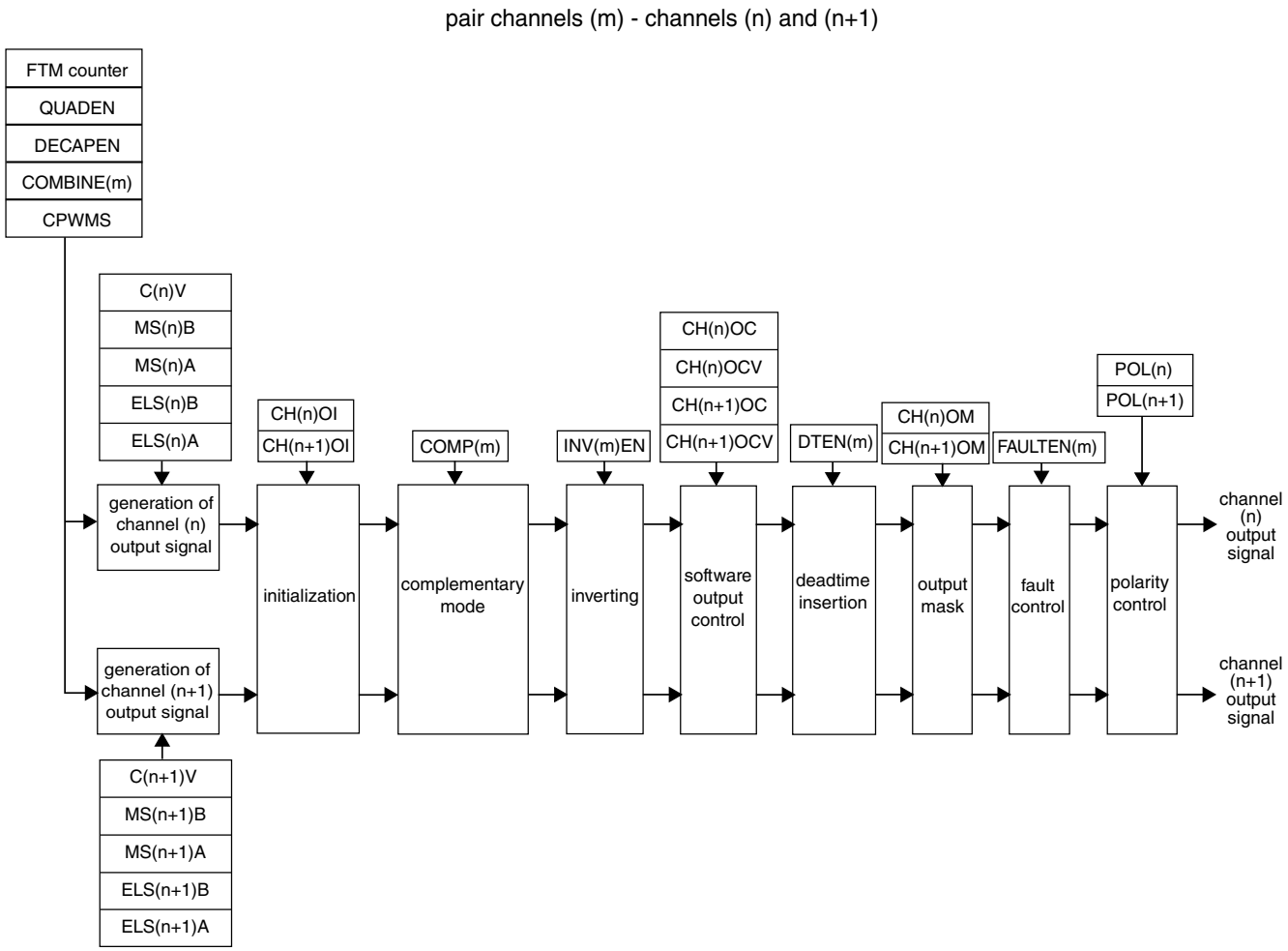
CH(n)OI	CH(n+1)OI	Channel (n) Output	Channel (n+1) Output
0	X	is forced to zero	is forced to one
1	X	is forced to one	is forced to zero

Note

The initialization feature must be used only with disabled FTM counter. See the description of the CLKS field in the Status and Control register.

19.5.19 Features priority

The following figure shows the priority of the features used at the generation of channels (n) and (n+1) outputs signals.



NOTE

The channels (n) and (n+1) are in output compare, EPWM, CPWM or combine modes.

Figure 19-74. Priority of the features used at the generation of channels (n) and (n+1) outputs signals

Note

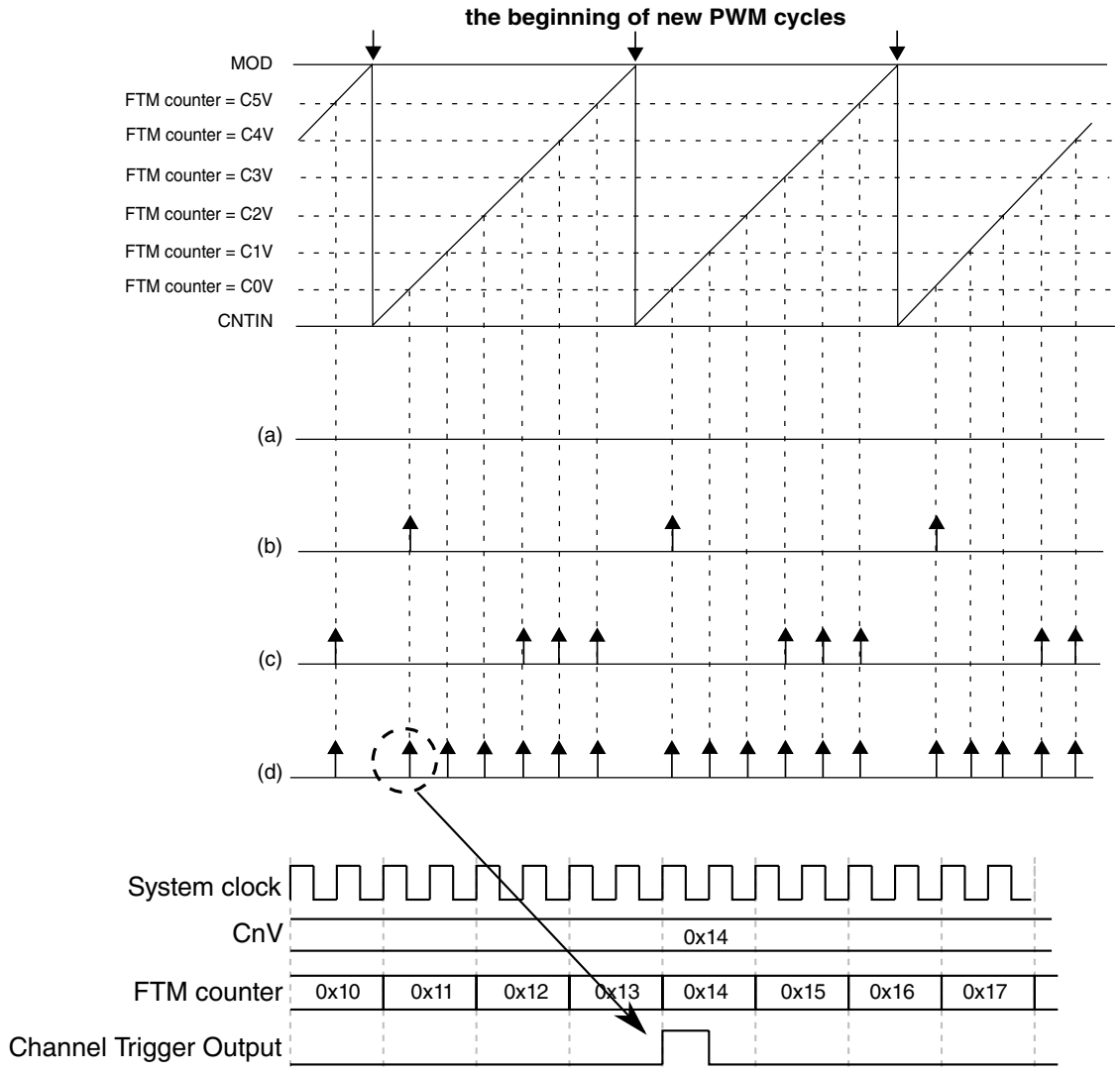
The Initialization feature must not be used with Inverting and Software output control features.

19.5.20 Channel trigger output

If CH(j)TRIG bit of the FTM External Trigger (FTM_EXTTRIG) register is set, where j = 0, 1, 2, 3, 4, or 5, then the FTM generates a trigger when the channel (j) match occurs (FTM counter = C(j)V).

The channel trigger output provides a trigger signal which has one FTM clock period width and is used for on-chip modules.

The FTM is able to generate multiple triggers in one PWM period. Because each trigger is generated for a specific channel, several channels are required to implement this functionality. This behavior is described in the following figure.



NOTE

(a) CH0TRIG = 0, CH1TRIG = 0, CH2TRIG = 0, CH3TRIG = 0, CH4TRIG = 0, CH5TRIG = 0
(b) CH0TRIG = 1, CH1TRIG = 0, CH2TRIG = 0, CH3TRIG = 0, CH4TRIG = 0, CH5TRIG = 0
(c) CH0TRIG = 0, CH1TRIG = 0, CH2TRIG = 0, CH3TRIG = 1, CH4TRIG = 1, CH5TRIG = 1
(d) CH0TRIG = 1, CH1TRIG = 1, CH2TRIG = 1, CH3TRIG = 1, CH4TRIG = 1, CH5TRIG = 1

Figure 19-75. Channel match trigger

19.5.21 Initialization trigger

If INITTRIGEN = 1, then the FTM generates a trigger when the FTM counter is updated with the CNTIN register value in the following cases.

- The FTM counter is automatically updated with the CNTIN register value by the selected counting mode.
- When there is a write to CNT register.
- When there is the FTM counter synchronization.
- If (CNT = CNTIN), (CLKS[1:0] = 0:0), and a value different from zero is written to CLKS[1:0] bits.

The following figures show these cases.

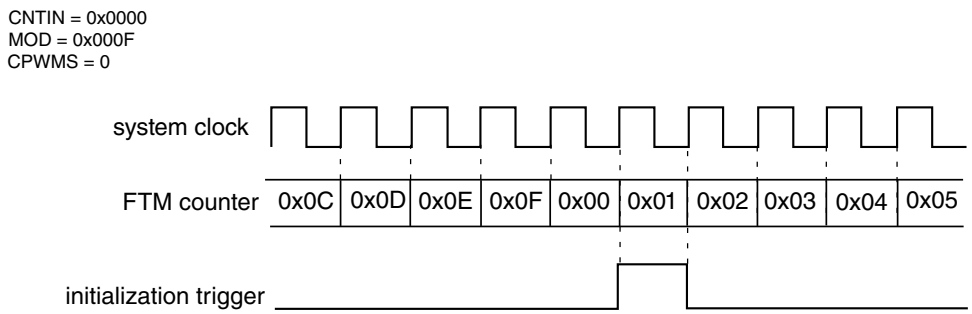


Figure 19-76. Initialization trigger is generated when the FTM counting achieves the CNTIN register value

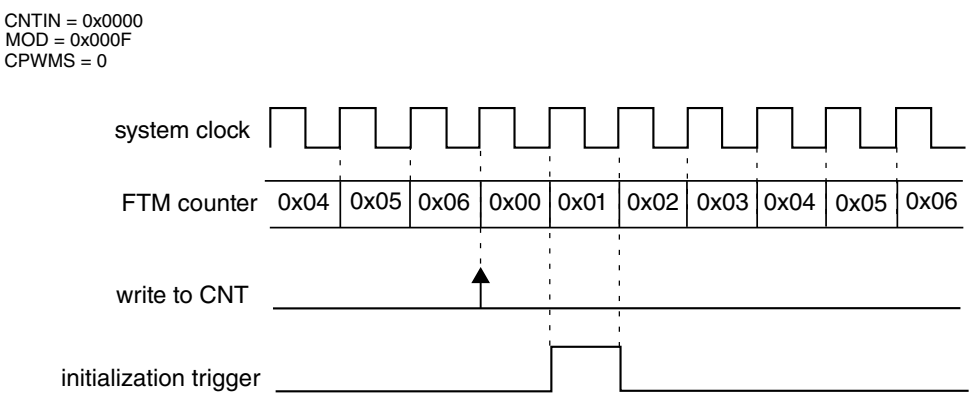


Figure 19-77. Initialization trigger is generated when there is a write to CNT register

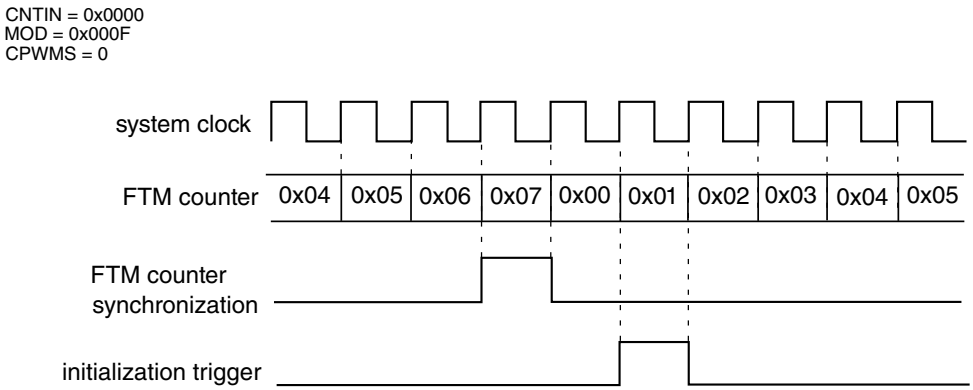


Figure 19-78. Initialization trigger is generated when there is the FTM counter synchronization

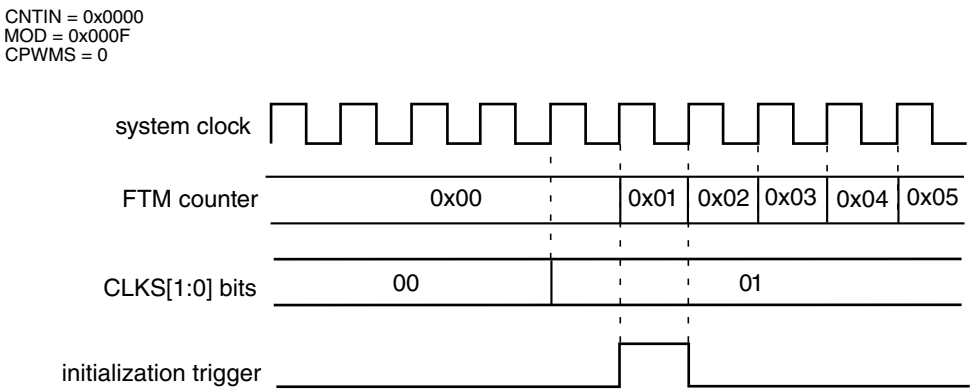


Figure 19-79. Initialization trigger is generated if (CNT = CNTIN), (CLKS[1:0] = 0:0), and a value different from zero is written to CLKS[1:0] bits

The initialization trigger output provides a trigger signal that is used for on-chip modules.

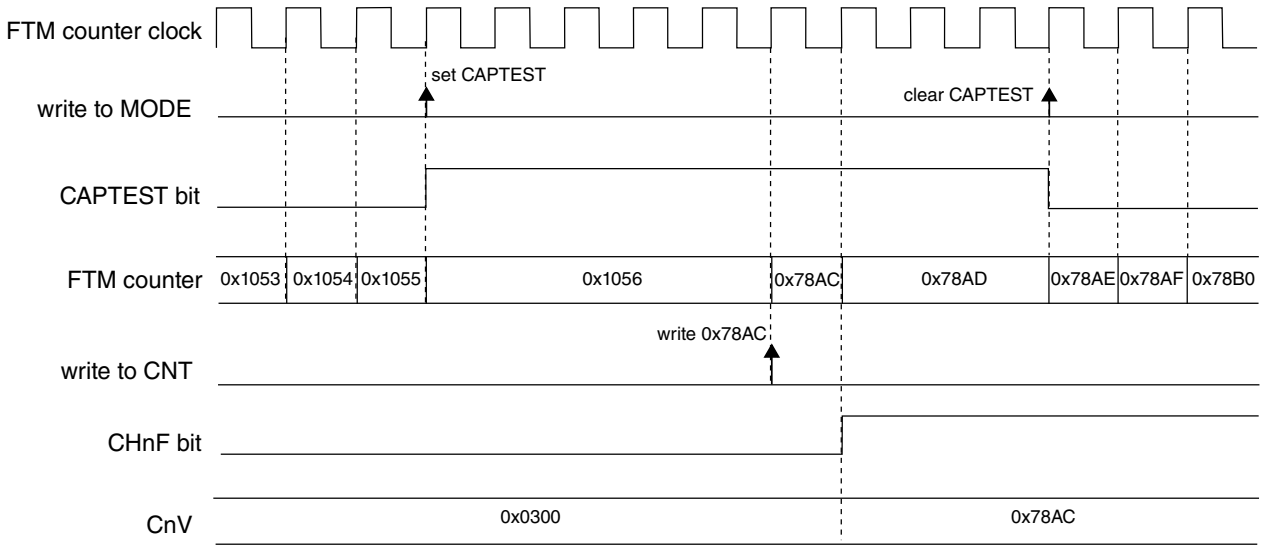
19.5.22 Capture Test mode

The Capture Test mode allows to test the CnV registers, the FTM counter and the interconnection logic between the FTM counter and CnV registers.

In this test mode, all channels must be configured for Input Capture mode and FTM counter must be configured to the Up counting.

When the Capture Test mode is enabled (CAPTEST = 1), the FTM counter is frozen and any write to CNT register updates directly the FTM counter; see the following figure. After it was written, all CnV registers are updated with the written value to CNT register and CHnF bits are set. Therefore, the FTM counter is updated with its next value according to its configuration. Its next value depends on CNTIN, MOD, and the written value to FTM counter.

The next reads of CnV registers return the written value to the FTM counter and the next reads of CNT register return FTM counter next value.



NOTE
- FTM counter configuration: (FTMEN = 1), (QUADEN = 0), (CAPTEST = 1), (CPWMS = 0), (CNTIN = 0x0000), and (MOD = 0xFFFF)
- FTM channel n configuration: input capture mode - (DECAPEN = 0), (COMBINE = 0), and (MSnB:MSnA = 0:0)

Figure 19-80. Capture Test mode

19.5.23 DMA

The channel generates a DMA transfer request according to DMA and CHnIE bits. See the following table.

Table 19-17. Channel DMA transfer request

DMA	CHnIE	Channel DMA Transfer Request	Channel Interrupt
0	0	The channel DMA transfer request is not generated.	The channel interrupt is not generated.
0	1	The channel DMA transfer request is not generated.	The channel interrupt is generated if (CHnF = 1).
1	0	The channel DMA transfer request is not generated.	The channel interrupt is not generated.
1	1	The channel DMA transfer request is generated if (CHnF = 1).	The channel interrupt is not generated.

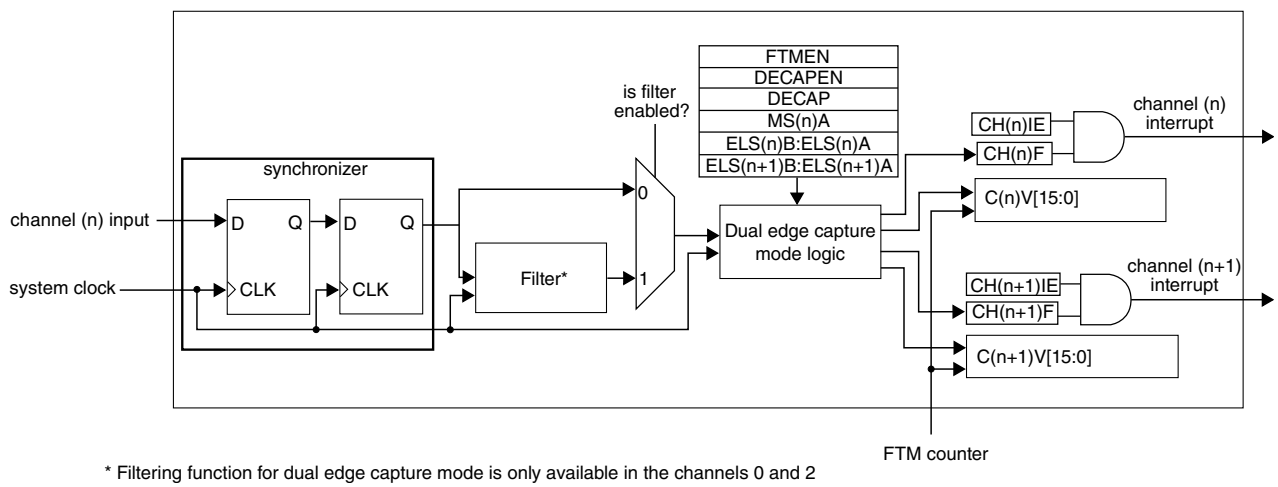
If DMA = 1, the CHnF bit is cleared either by channel DMA transfer done or reading CnSC while CHnF is set and then writing a zero to CHnF bit according to CHnIE bit. See the following table.

Table 19-18. Clear CHnF bit when DMA = 1

CHnIE	How CHnF Bit Can Be Cleared
0	CHnF bit is cleared either when the channel DMA transfer is done or by reading CnSC while CHnF is set and then writing a 0 to CHnF bit.
1	CHnF bit is cleared when the channel DMA transfer is done.

19.5.24 Dual Edge Capture mode

The Dual Edge Capture mode is selected if DECAPEN = 1. This mode allows to measure a pulse width or period of the signal on the input of channel (n) of a channel pair. The channel (n) filter can be active in this mode when n is 0 or 2.



* Filtering function for dual edge capture mode is only available in the channels 0 and 2

Figure 19-81. Dual Edge Capture mode block diagram

The MS(n)A bit defines if the Dual Edge Capture mode is one-shot or continuous.

The ELS(n)B:ELS(n)A bits select the edge that is captured by channel (n), and ELS(n +1)B:ELS(n+1)A bits select the edge that is captured by channel (n+1). If both ELS(n)B:ELS(n)A and ELS(n+1)B:ELS(n+1)A bits select the same edge, then it is the period measurement. If these bits select different edges, then it is a pulse width measurement.

In the Dual Edge Capture mode, only channel (n) input is used and channel (n+1) input is ignored.

If the selected edge by channel (n) bits is detected at channel (n) input, then CH(n)F bit is set and the channel (n) interrupt is generated (if CH(n)IE = 1). If the selected edge by channel (n+1) bits is detected at channel (n) input and (CH(n)F = 1), then CH(n+1)F bit is set and the channel (n+1) interrupt is generated (if CH(n+1)IE = 1).

The C(n)V register stores the value of FTM counter when the selected edge by channel (n) is detected at channel (n) input. The C(n+1)V register stores the value of FTM counter when the selected edge by channel (n+1) is detected at channel (n) input.

In this mode, a coherency mechanism ensures coherent data when the C(n)V and C(n +1)V registers are read. The only requirement is that C(n)V must be read before C(n +1)V.

Note

- The CH(n)F, CH(n)IE, MS(n)A, ELS(n)B, and ELS(n)A bits are channel (n) bits.
- The CH(n+1)F, CH(n+1)IE, MS(n+1)A, ELS(n+1)B, and ELS(n+1)A bits are channel (n+1) bits.
- The Dual Edge Capture mode must be used with ELS(n)B:ELS(n)A = 0:1 or 1:0, ELS(n+1)B:ELS(n+1)A = 0:1 or 1:0 and the FTM counter in Free running counter.

19.5.24.1 One-Shot Capture mode

The One-Shot Capture mode is selected when (DECAPEN = 1), and (MS(n)A = 0). In this capture mode, only one pair of edges at the channel (n) input is captured. The ELS(n)B:ELS(n)A bits select the first edge to be captured, and ELS(n+1)B:ELS(n+1)A bits select the second edge to be captured.

The edge captures are enabled while DECAP bit is set. For each new measurement in One-Shot Capture mode, first the CH(n)F and CH(n+1) bits must be cleared, and then the DECAP bit must be set.

In this mode, the DECAP bit is automatically cleared by FTM when the edge selected by channel (n+1) is captured. Therefore, while DECAP bit is set, the one-shot capture is in process. When this bit is cleared, both edges were captured and the captured values are ready for reading in the C(n)V and C(n+1)V registers.

Similarly, when the CH(n+1)F bit is set, both edges were captured and the captured values are ready for reading in the C(n)V and C(n+1)V registers.

19.5.24.2 Continuous Capture mode

The Continuous Capture mode is selected when (DECAPEN = 1), and (MS(n)A = 1). In this capture mode, the edges at the channel (n) input are captured continuously. The ELS(n)B:ELS(n)A bits select the initial edge to be captured, and ELS(n+1)B:ELS(n+1)A bits select the final edge to be captured.

The edge captures are enabled while DECAP bit is set. For the initial use, first the CH(n)F and CH(n+1)F bits must be cleared, and then DECAP bit must be set to start the continuous measurements.

When the CH(n+1)F bit is set, both edges were captured and the captured values are ready for reading in the C(n)V and C(n+1)V registers. The latest captured values are always available in these registers even after the DECAP bit is cleared.

In this mode, it is possible to clear only the CH(n+1)F bit. Therefore, when the CH(n+1)F bit is set again, the latest captured values are available in C(n)V and C(n+1)V registers.

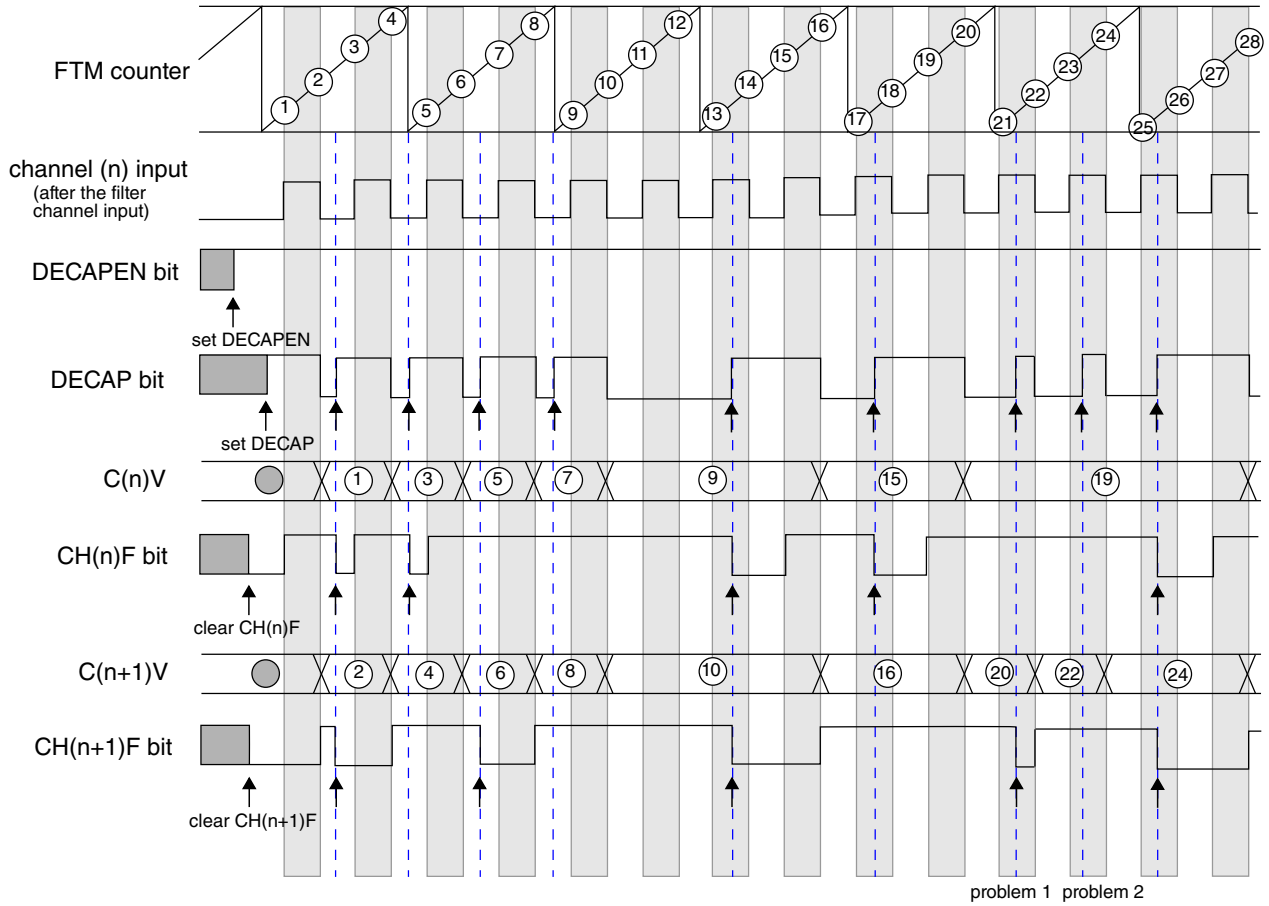
For a new sequence of the measurements in the Dual Edge Capture – Continuous mode, clear the CH(n)F and CH(n+1)F bits to start new measurements.

19.5.24.3 Pulse width measurement

If the channel (n) is configured to capture rising edges (ELS(n)B:ELS(n)A = 0:1) and the channel (n+1) to capture falling edges (ELS(n+1)B:ELS(n+1)A = 1:0), then the positive polarity pulse width is measured. If the channel (n) is configured to capture falling edges (ELS(n)B:ELS(n)A = 1:0) and the channel (n+1) to capture rising edges (ELS(n +1)B:ELS(n+1)A = 0:1), then the negative polarity pulse width is measured.

The pulse width measurement can be made in One-Shot Capture mode or Continuous Capture mode.

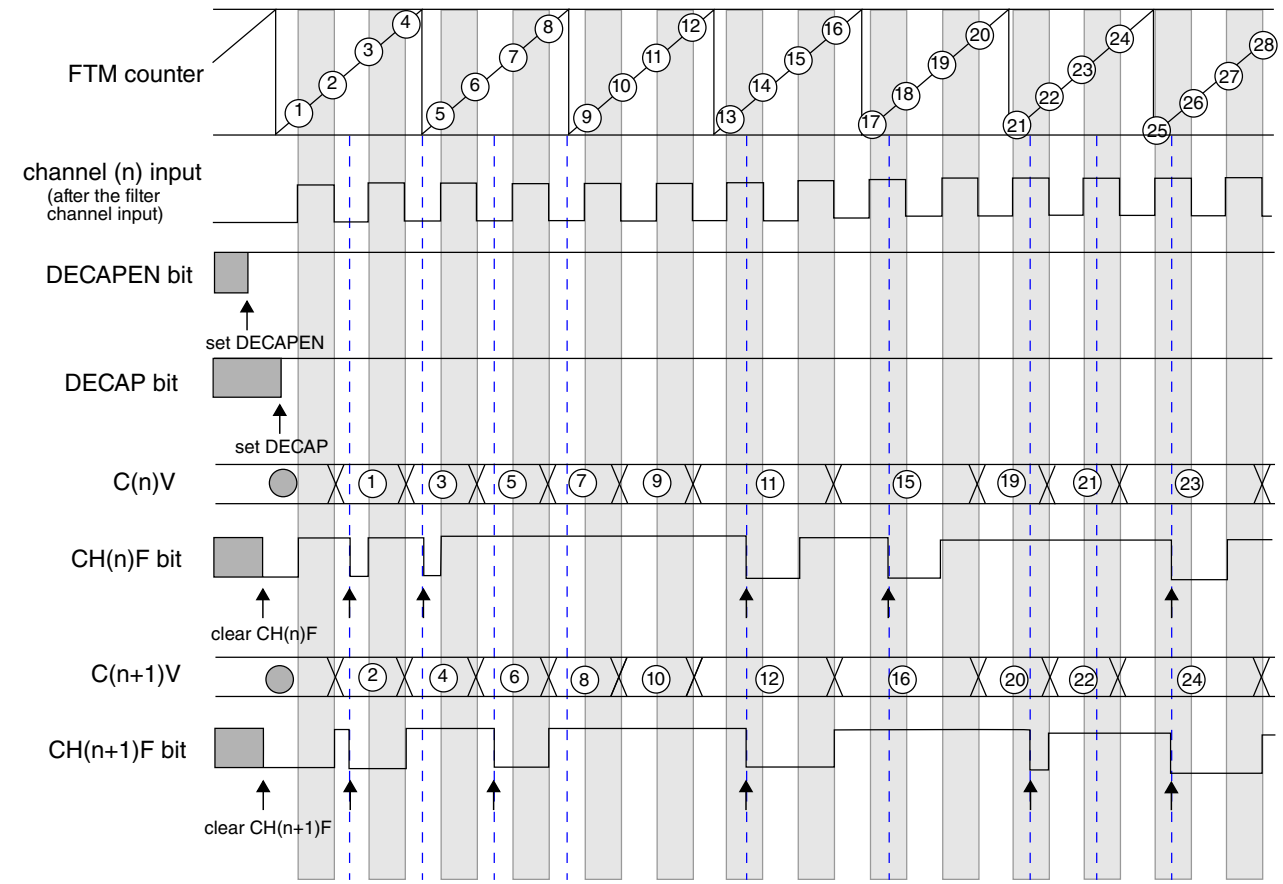
The following figure shows an example of the Dual Edge Capture – One-Shot mode used to measure the positive polarity pulse width. The DECAPEN bit selects the Dual Edge Capture mode, so it remains set. The DECAP bit is set to enable the measurement of next positive polarity pulse width. The CH(n)F bit is set when the first edge of this pulse is detected, that is, the edge selected by ELS(n)B:ELS(n)A bits. The CH(n+1)F bit is set and DECAP bit is cleared when the second edge of this pulse is detected, that is, the edge selected by ELS(n+1)B:ELS(n+1)A bits. Both DECAP and CH(n+1)F bits indicate when two edges of the pulse were captured and the C(n)V and C(n+1)V registers are ready for reading.



Note
- The commands set DECAPEN, set DECAP, clear CH(n)F, and clear CH(n+1)F are made by the user.
- Problem 1: channel (n) input = 1, set DECAP, not clear CH(n)F, and clear CH(n+1)F.
- Problem 2: channel (n) input = 1, set DECAP, not clear CH(n)F, and not clear CH(n+1)F.

Figure 19-82. Dual Edge Capture – One-Shot mode for positive polarity pulse width measurement

The following figure shows an example of the Dual Edge Capture – Continuous mode used to measure the positive polarity pulse width. The DECAPEN bit selects the Dual Edge Capture mode, so it remains set. While the DECAP bit is set the configured measurements are made. The CH(n)F bit is set when the first edge of the positive polarity pulse is detected, that is, the edge selected by ELS(n)B:ELS(n)A bits. The CH(n+1)F bit is set when the second edge of this pulse is detected, that is, the edge selected by ELS(n +1)B:ELS(n+1)A bits. The CH(n+1)F bit indicates when two edges of the pulse were captured and the C(n)V and C(n+1)V registers are ready for reading.



Note
- The commands set DECAPEN, set DECAP, clear CH(n)F, and clear CH(n+1)F are made by the user.

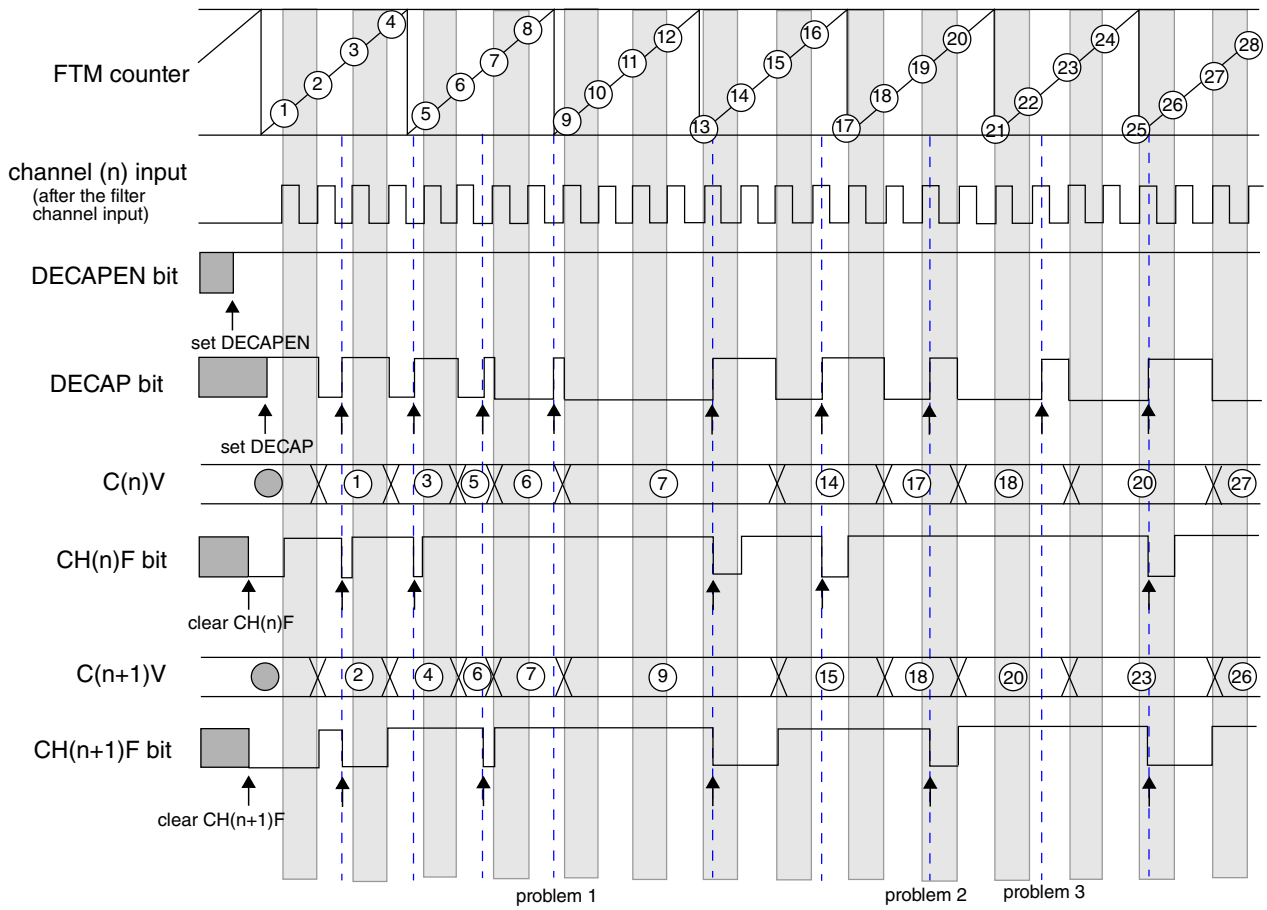
Figure 19-83. Dual Edge Capture – Continuous mode for positive polarity pulse width measurement

19.5.24.4 Period measurement

If the channels (n) and (n+1) are configured to capture consecutive edges of the same polarity, then the period of the channel (n) input signal is measured. If both channels (n) and (n+1) are configured to capture rising edges (ELS(n)B:ELS(n)A = 0:1 and ELS(n +1)B:ELS(n+1)A = 0:1), then the period between two consecutive rising edges is measured. If both channels (n) and (n+1) are configured to capture falling edges (ELS(n)B:ELS(n)A = 1:0 and ELS(n+1)B:ELS(n+1)A = 1:0), then the period between two consecutive falling edges is measured.

The period measurement can be made in One-Shot Capture mode or Continuous Capture mode.

The following figure shows an example of the Dual Edge Capture – One-Shot mode used to measure the period between two consecutive rising edges. The DECAPEN bit selects the Dual Edge Capture mode, so it remains set. The DECAP bit is set to enable the measurement of next period. The CH(n)F bit is set when the first rising edge is detected, that is, the edge selected by ELS(n)B:ELS(n)A bits. The CH(n+1)F bit is set and DECAP bit is cleared when the second rising edge is detected, that is, the edge selected by ELS(n +1)B:ELS(n+1)A bits. Both DECAP and CH(n+1)F bits indicate when two selected edges were captured and the C(n)V and C(n+1)V registers are ready for reading.

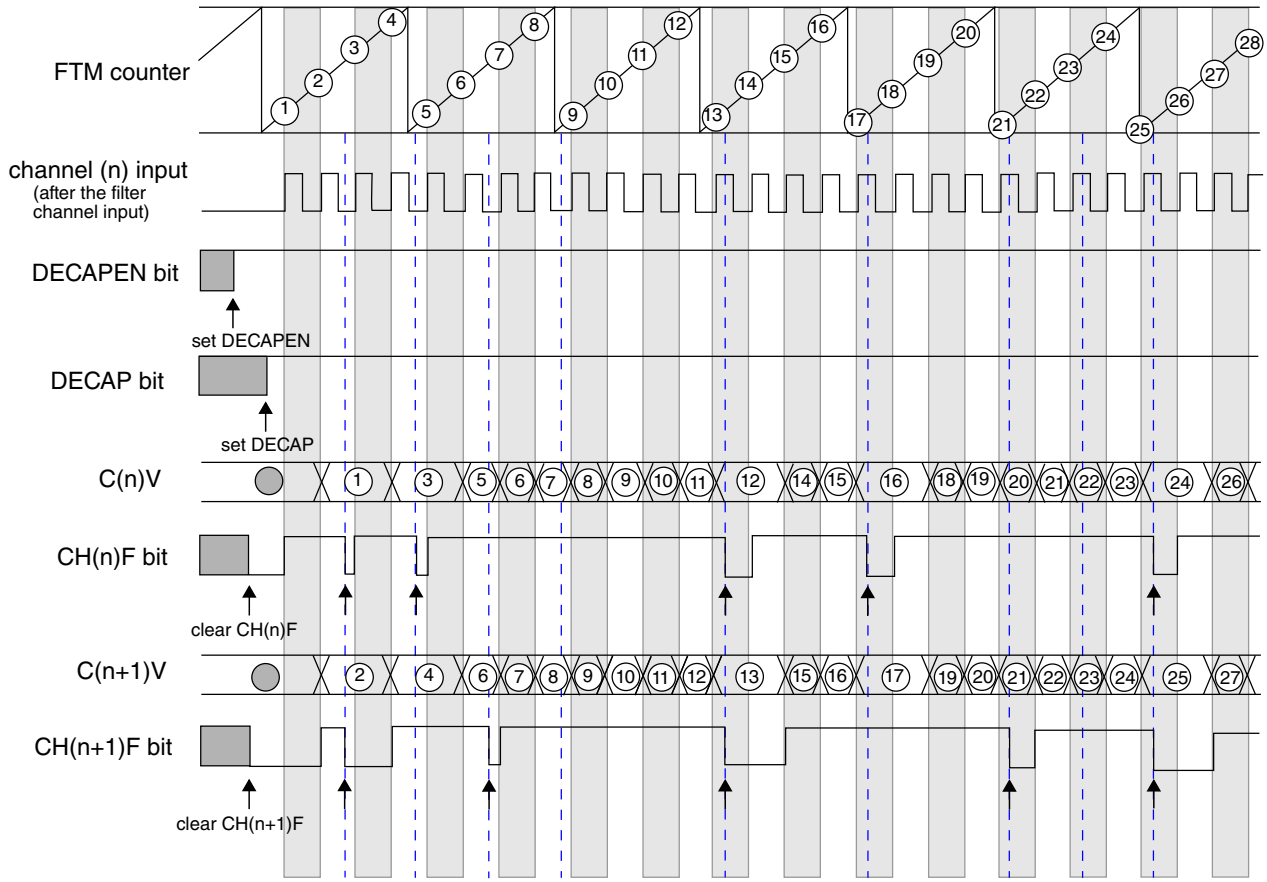


Note
- The commands set DECAPEN, set DECAP, clear CH(n)F, and clear CH(n+1)F are made by the user.
- Problem 1: channel (n) input = 0, set DECAP, not clear CH(n)F, and not clear CH(n+1)F.
- Problem 2: channel (n) input = 1, set DECAP, not clear CH(n)F, and clear CH(n+1)F.
- Problem 3: channel (n) input = 1, set DECAP, not clear CH(n)F, and not clear CH(n+1)F.

Figure 19-84. Dual Edge Capture – One-Shot mode to measure of the period between two consecutive rising edges

The following figure shows an example of the Dual Edge Capture – Continuous mode used to measure the period between two consecutive rising edges. The DECAPEN bit selects the Dual Edge Capture mode, so it remains set. While the DECAP bit is set the configured measurements are made. The CH(n)F bit is set when the first rising edge is detected, that is, the edge selected by ELS(n)B:ELS(n)A bits. The CH(n+1)F bit is set

when the second rising edge is detected, that is, the edge selected by ELS(n+1)B:ELS(n +1)A bits. The CH(n+1)F bit indicates when two edges of the period were captured and the C(n)V and C(n+1)V registers are ready for reading.



Note
- The commands set DECAPEN, set DECAP, clear CH(n)F, and clear CH(n+1)F are made by the user.

Figure 19-85. Dual Edge Capture – Continuous mode to measure of the period between two consecutive rising edges

19.5.24.5 Read coherency mechanism

The Dual Edge Capture mode implements a read coherency mechanism between the FTM counter value captured in C(n)V and C(n+1)V registers. The read coherency mechanism is illustrated in the following figure. In this example, the channels (n) and (n +1) are in Dual Edge Capture – Continuous mode for positive polarity pulse width measurement. Thus, the channel (n) is configured to capture the FTM counter value when there is a rising edge at channel (n) input signal, and channel (n+1) to capture the FTM counter value when there is a falling edge at channel (n) input signal.

When a rising edge occurs in the channel (n) input signal, the FTM counter value is captured into channel (n) capture buffer. The channel (n) capture buffer value is transferred to C(n)V register when a falling edge occurs in the channel (n) input signal. C(n)V register has the FTM counter value when the previous rising edge occurred, and the channel (n) capture buffer has the FTM counter value when the last rising edge occurred.

When a falling edge occurs in the channel (n) input signal, the FTM counter value is captured into channel (n+1) capture buffer. The channel (n+1) capture buffer value is transferred to C(n+1)V register when the C(n)V register is read.

In the following figure, the read of C(n)V returns the FTM counter value when the event 1 occurred and the read of C(n+1)V returns the FTM counter value when the event 2 occurred.

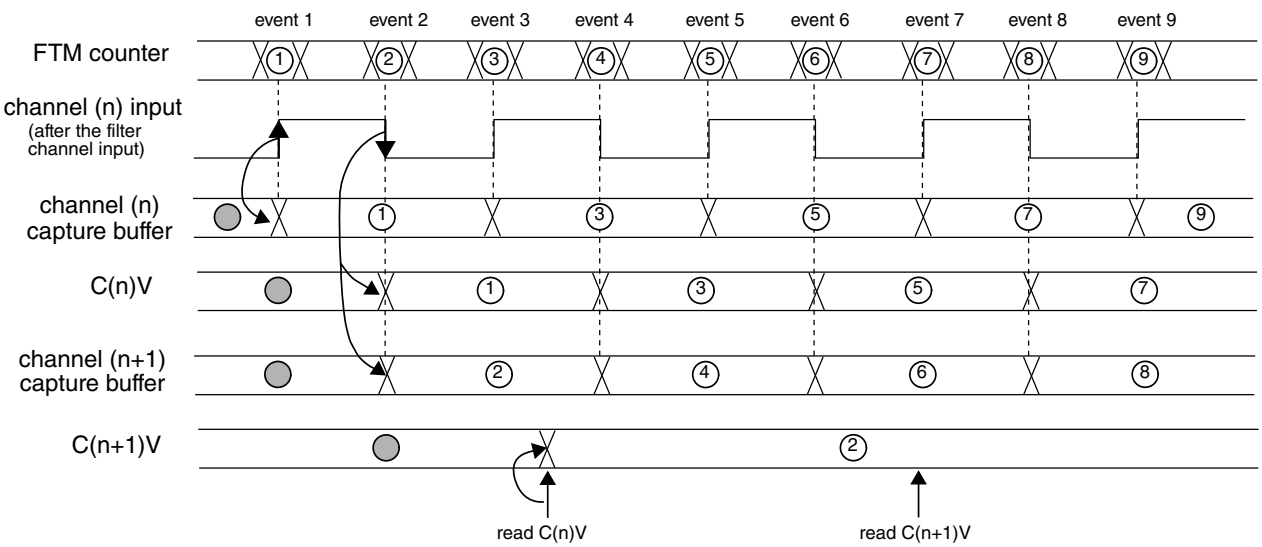


Figure 19-86. Dual Edge Capture mode read coherency mechanism

C(n)V register must be read prior to C(n+1)V register in dual edge capture one-shot and continuous modes for the read coherency mechanism works properly.

19.5.25 Quadrature Decoder mode

The Quadrature Decoder mode is selected if (QUADEN = 1). The Quadrature Decoder mode uses the input signals phase A and B to control the FTM counter increment and decrement. The following figure shows the quadrature decoder block diagram.

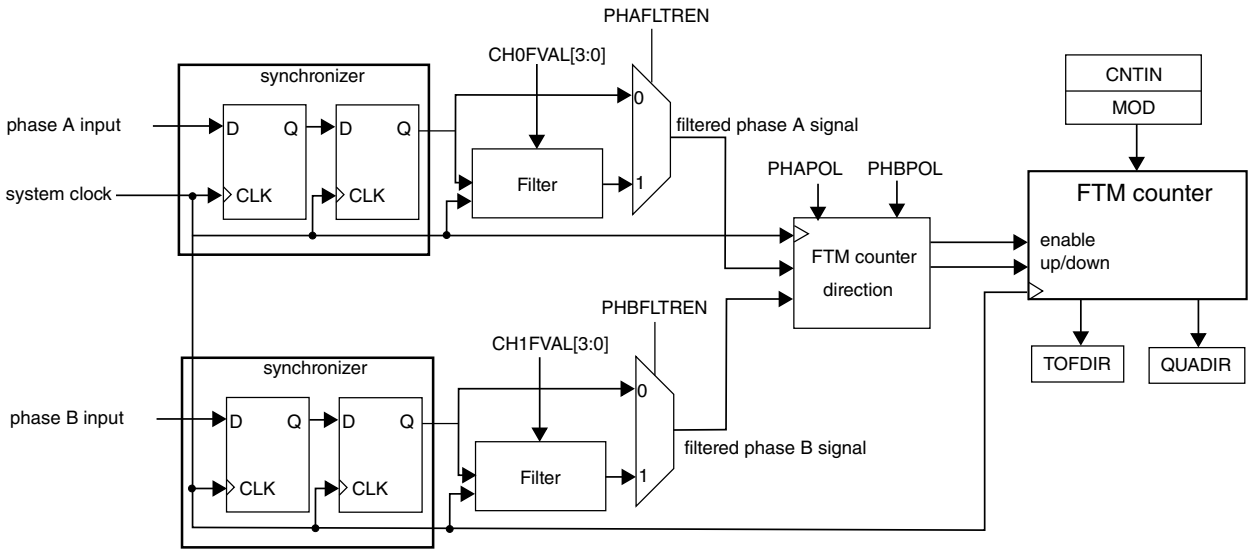


Figure 19-87. Quadrature Decoder block diagram

Each one of input signals phase A and B has a filter that is equivalent to the filter used in the channels input; Filter for Input Capture mode. The phase A input filter is enabled by PHAFLTREN bit and this filter's value is defined by CH0FVAL[3:0] bits (CH(n)FVAL[3:0] bits in FILTER0 register). The phase B input filter is enabled by PHBFLTREN bit and this filter's value is defined by CH1FVAL[3:0] bits (CH(n +1)FVAL[3:0] bits in FILTER0 register).

Except for CH0FVAL[3:0] and CH1FVAL[3:0] bits, no channel logic is used in Quadrature Decoder mode.

Note

Notice that the FTM counter is clocked by the phase A and B input signals when quadrature decoder mode is selected. Therefore it is expected that the Quadrature Decoder be used only with the FTM channels in input capture or output compare modes.

Note

An edge at phase A must not occur together an edge at phase B and vice-versa.

The PHAPOL bit selects the polarity of the phase A input, and the PHBPOL bit selects the polarity of the phase B input.

The QUADMODE selects the encoding mode used in the Quadrature Decoder mode. If QUADMODE = 1, then the count and direction encoding mode is enabled; see the following figure. In this mode, the phase B input value indicates the counting direction, and the phase A input defines the counting rate. The FTM counter is updated when there is a rising edge at phase A input signal.

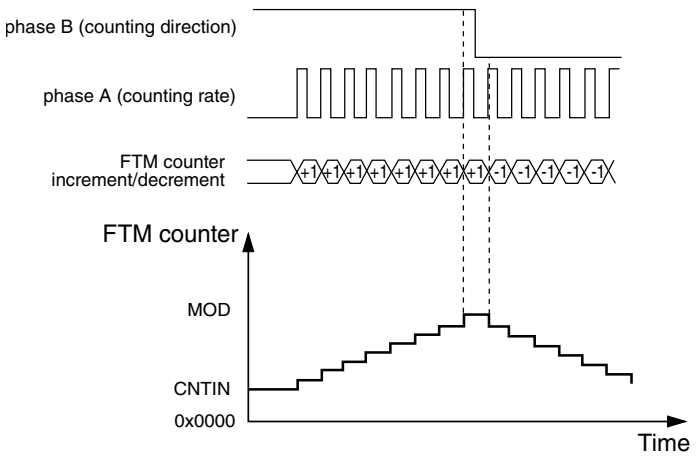


Figure 19-88. Quadrature Decoder – Count and Direction Encoding mode

If QUADMODE = 0, then the Phase A and Phase B Encoding mode is enabled; see the following figure. In this mode, the relationship between phase A and B signals indicates the counting direction, and phase A and B signals define the counting rate. The FTM counter is updated when there is an edge either at the phase A or phase B signals.

If PHAPOL = 0 and PHBPOL = 0, then the FTM counter increment happens when:

- there is a rising edge at phase A signal and phase B signal is at logic zero;
- there is a rising edge at phase B signal and phase A signal is at logic one;
- there is a falling edge at phase B signal and phase A signal is at logic zero;
- there is a falling edge at phase A signal and phase B signal is at logic one;

and the FTM counter decrement happens when:

- there is a falling edge at phase A signal and phase B signal is at logic zero;
- there is a falling edge at phase B signal and phase A signal is at logic one;
- there is a rising edge at phase B signal and phase A signal is at logic zero;
- there is a rising edge at phase A signal and phase B signal is at logic one.

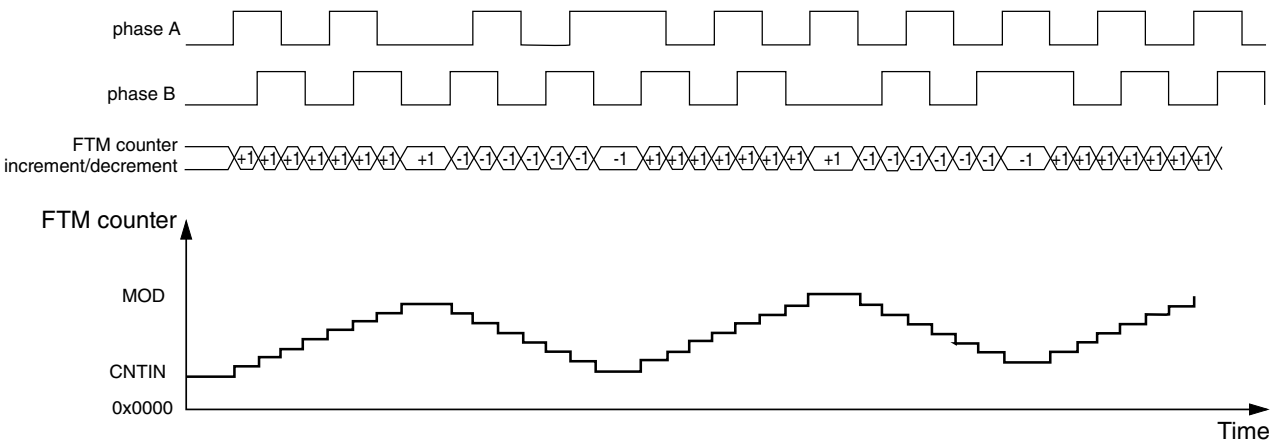


Figure 19-89. Quadrature Decoder – Phase A and Phase B Encoding mode

The following figure shows the FTM counter overflow in up counting. In this case, when the FTM counter changes from MOD to CNTIN, TOF and TOFDIR bits are set. TOF bit indicates the FTM counter overflow occurred. TOFDIR indicates the counting was up when the FTM counter overflow occurred.

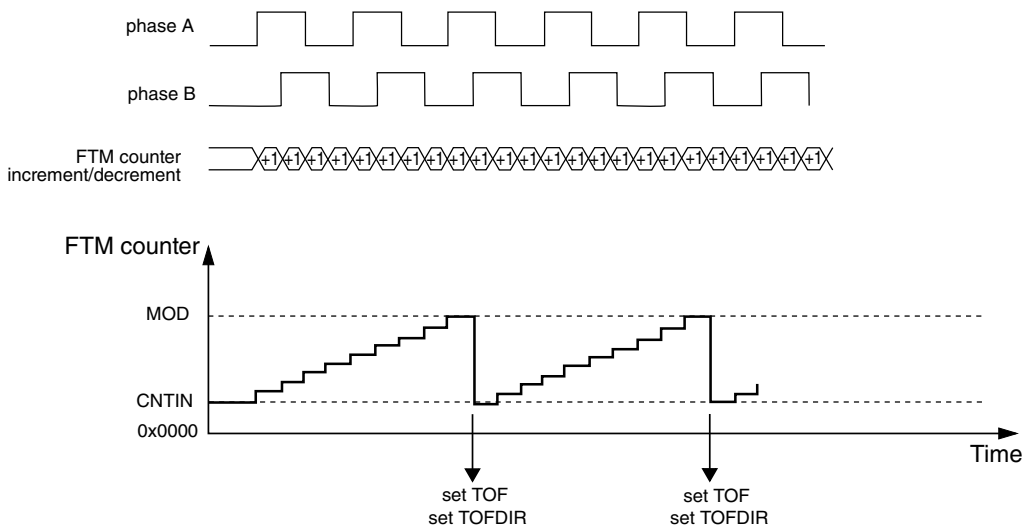


Figure 19-90. FTM Counter overflow in up counting for Quadrature Decoder mode

The following figure shows the FTM counter overflow in down counting. In this case, when the FTM counter changes from CNTIN to MOD, TOF bit is set and TOFDIR bit is cleared. TOF bit indicates the FTM counter overflow occurred. TOFDIR indicates the counting was down when the FTM counter overflow occurred.

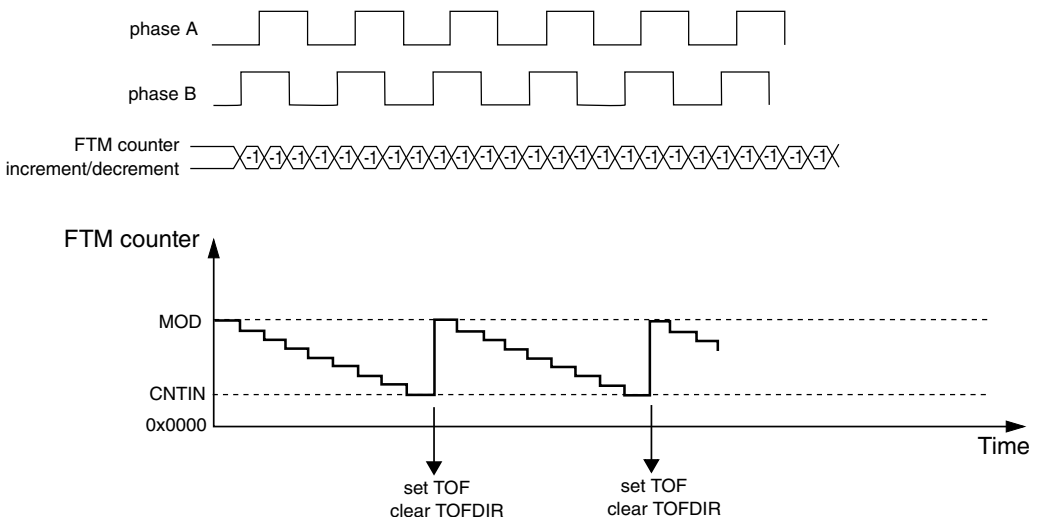


Figure 19-91. FTM counter overflow in down counting for Quadrature Decoder mode

19.5.25.1 Quadrature Decoder boundary conditions

The following figures show the FTM counter responding to motor jittering typical in motor position control applications.

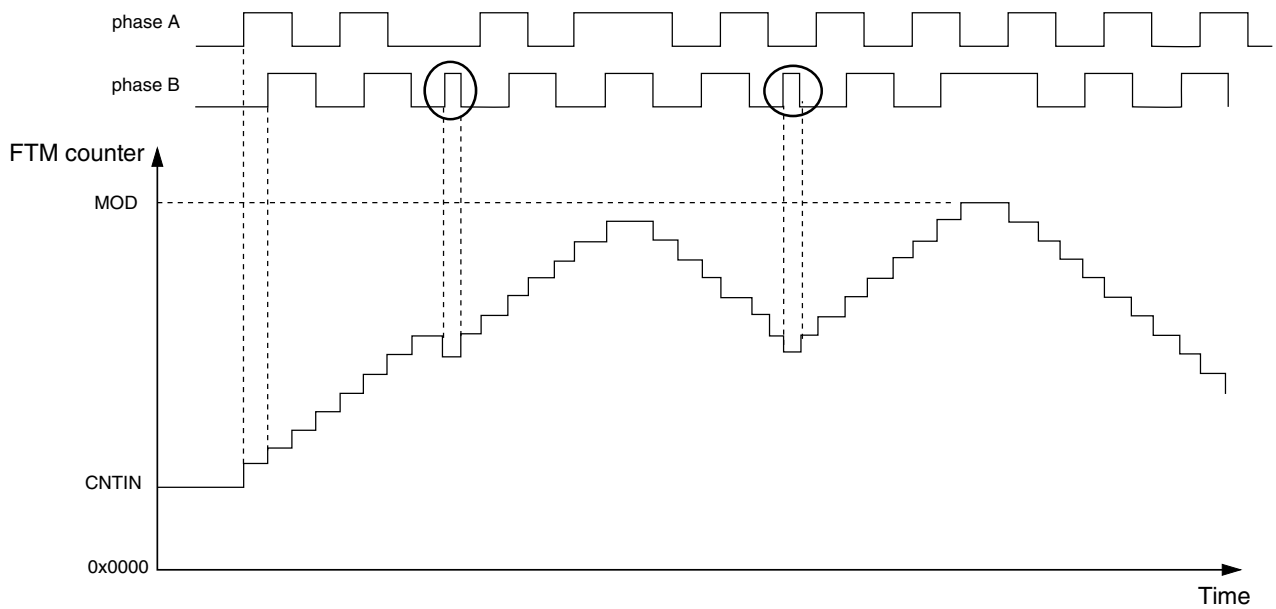


Figure 19-92. Motor position jittering in a mid count value

The following figure shows motor jittering produced by the phase B and A pulses respectively:

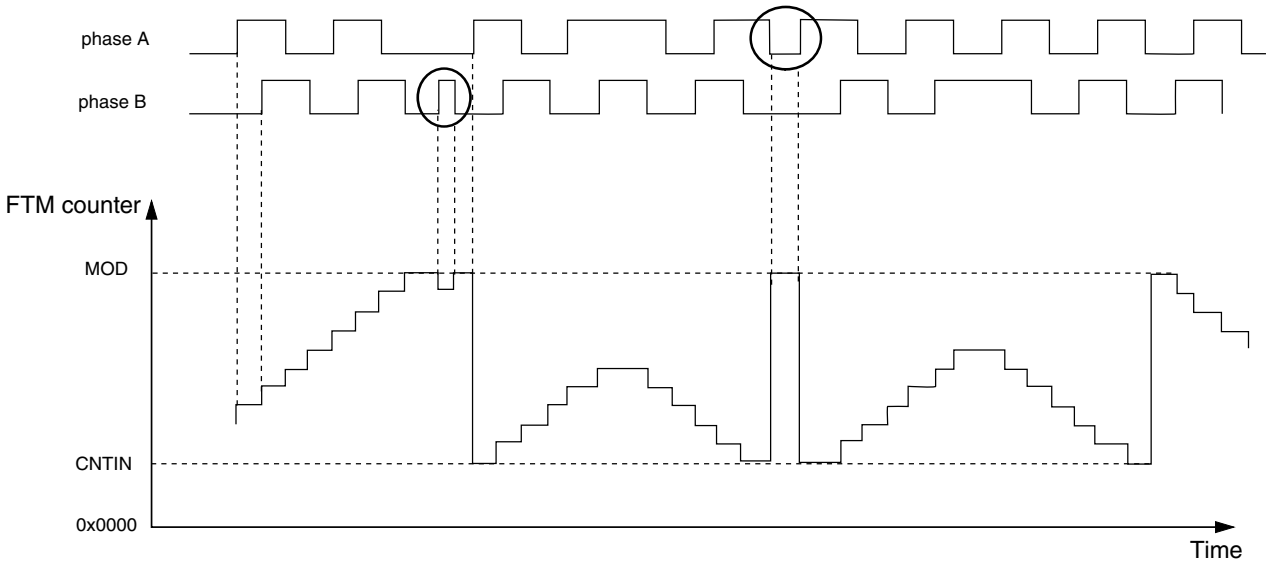


Figure 19-93. Motor position jittering near maximum and minimum count value

The first highlighted transition causes a jitter on the FTM counter value near the maximum count value (MOD). The second indicated transition occurs on phase A and causes the FTM counter transition between the maximum and minimum count values which are defined by MOD and CNTIN registers.

The appropriate settings of the phase A and phase B input filters are important to avoid glitches that may cause oscillation on the FTM counter value. The preceding figures show examples of oscillations that can be caused by poor input filter setup. Thus, it is important to guarantee a minimum pulse width to avoid these oscillations.

19.5.26 Intermediate load

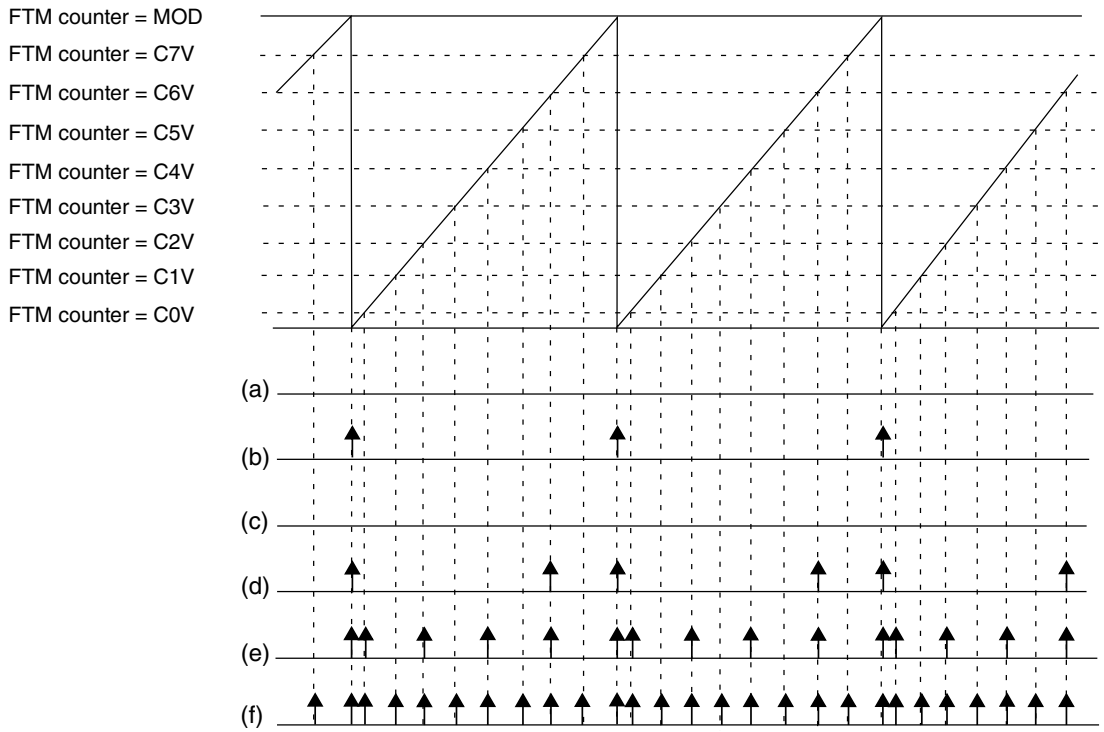
The PWMLOAD register allows to update the MOD, CNTIN, and C(n)V registers with the content of the register buffer at a defined load point. In this case, it is not required to use the PWM synchronization.

There are multiple possible loading points for intermediate load:

Table 19-19. When possible loading points are enabled

Loading point	Enabled
When the FTM counter wraps from MOD value to CNTIN value	Always
At the channel (j) match (FTM counter = C(j)V)	When CHjSEL = 1

The following figure shows some examples of enabled loading points.



NOTE

(a) LDOK = 0, CH0SEL = 0, CH1SEL = 0, CH2SEL = 0, CH3SEL = 0, CH4SEL = 0, CH5SEL = 0, CH6SEL = 0, CH7SEL = 0
(b) LDOK = 1, CH0SEL = 0, CH1SEL = 0, CH2SEL = 0, CH3SEL = 0, CH4SEL = 0, CH5SEL = 0, CH6SEL = 0, CH7SEL = 0
(c) LDOK = 0, CH0SEL = 0, CH1SEL = 0, CH2SEL = 0, CH3SEL = 1, CH4SEL = 0, CH5SEL = 0, CH6SEL = 0, CH7SEL = 0
(d) LDOK = 1, CH0SEL = 0, CH1SEL = 0, CH2SEL = 0, CH3SEL = 0, CH4SEL = 0, CH5SEL = 0, CH6SEL = 1, CH7SEL = 0
(e) LDOK = 1, CH0SEL = 1, CH1SEL = 0, CH2SEL = 1, CH3SEL = 0, CH4SEL = 1, CH5SEL = 0, CH6SEL = 1, CH7SEL = 0
(f) LDOK = 1, CH0SEL = 1, CH1SEL = 1, CH2SEL = 1, CH3SEL = 1, CH4SEL = 1, CH5SEL = 1, CH6SEL = 1, CH7SEL = 1

Figure 19-94. Loading points for intermediate load

After enabling the loading points, the LDOK bit must be set for the load to occur. In this case, the load occurs at the next enabled loading point according to the following conditions:

Table 19-20. Conditions for loads occurring at the next enabled loading point

When a new value was written	Then
To the MOD register	The MOD register is updated with its write buffer value.
To the CNTIN register and CNTINC = 1	The CNTIN register is updated with its write buffer value.
To the C(n)V register and SYNCENm = 1 – where m indicates the pair channels (n) and (n+1)	The C(n)V register is updated with its write buffer value.
To the C(n+1)V register and SYNCENm = 1 – where m indicates the pair channels (n) and (n+1)	The C(n+1)V register is updated with its write buffer value.

NOTE

- If ELSjB and ELSjA bits are different from zero, then the channel (j) output signal is generated according to the configured output mode. If ELSjB and ELSjA bits are zero,

then the generated signal is not available on channel (j) output.
- If CHjIE = 1, then the channel (j) interrupt is generated when the channel (j) match occurs.
- At the intermediate load neither the channels outputs nor the FTM counter are changed. Software must set the intermediate load at a safe point in time.

19.5.27 Global time base (GTB)

The global time base (GTB) is a FTM function that allows the synchronization of multiple FTM modules on a chip. The following figure shows an example of the GTB feature used to synchronize two FTM modules. In this case, the FTM A and B channels can behave as if just one FTM module was used, that is, a global time base.

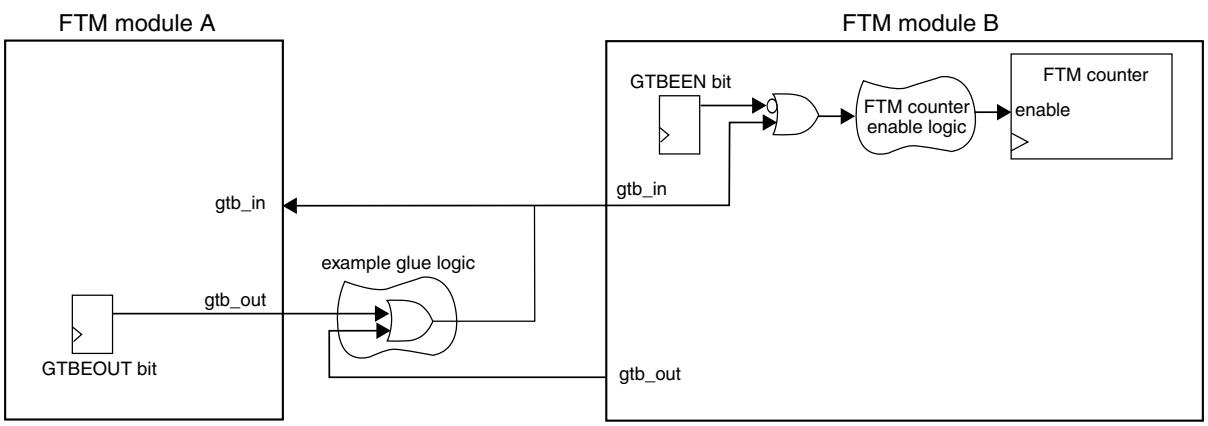


Figure 19-95. Global time base (GTB) block diagram

The GTB functionality is implemented by the GTBEEN and GTBEOUT bits in the CONF register, the input signal *gtb_in*, and the output signal *gtb_out*. The GTBEEN bit enables gtb_in to control the FTM counter enable signal:

- If GTBEEN = 0, each one of FTM modules works independently according to their configured mode.
- If GTBEEN = 1, the FTM counter update is enabled only when gtb_in is 1.

In the configuration described in the preceding figure, FTM modules A and B have their FTM counters enabled if at least one of the gtb_out signals from one of the FTM modules is 1. There are several possible configurations for the interconnection of the gtb_in and gtb_out signals, represented by the example glue logic shown in the figure. Note that these configurations are chip-dependent and implemented outside of the FTM modules. See the chip-specific FTM information for the chip's specific implementation.

NOTE

- In order to use the GTB signals to synchronize the FTM counter of different FTM modules, the configuration of each FTM module should guarantee that its FTM counter starts counting as soon as the gtb_in signal is 1.
- The GTB feature does not provide continuous synchronization of FTM counters, meaning that the FTM counters may lose synchronization during FTM operation. The GTB feature only allows the FTM counters to *start* their operation synchronously.

19.5.27.1 Enabling the global time base (GTB)

To enable the GTB feature, follow these steps for each participating FTM module:

1. Stop the FTM counter: Write 00b to SC[CLKS].
2. Program the FTM to the intended configuration. The FTM counter mode needs to be consistent across all participating modules.
3. Write 1 to CONF[GTBEEN] and write 0 to CONF[GTBEOUT] at the same time.
4. Select the intended FTM counter clock source in SC[CLKS]. The clock source needs to be consistent across all participating modules.
5. Reset the FTM counter: Write any value to the CNT register.

To initiate the GTB feature in the configuration described in the preceding figure, write 1 to CONF[GTBEOUT] in the FTM module used as the time base.

19.6 Reset overview

The FTM is reset whenever any chip reset occurs.

When the FTM exits from reset:

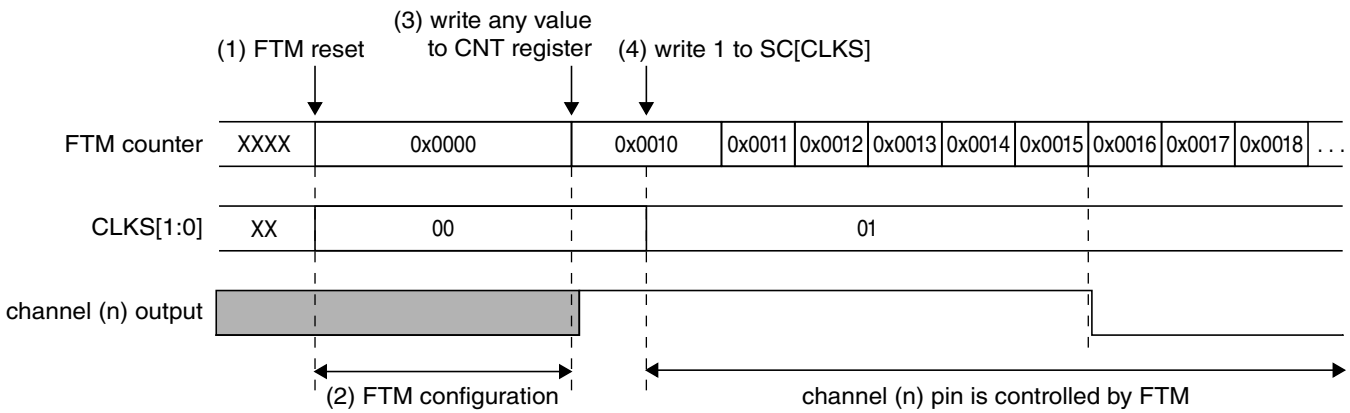
- the FTM counter and the prescaler counter are zero and are stopped (CLKS[1:0] = 00b);
- the timer overflow interrupt is zero, see Timer Overflow Interrupt;
- the channels interrupts are zero, see Channel (n) Interrupt;
- the fault interrupt is zero, see Fault Interrupt;
- the channels are in input capture mode, see Input Capture mode;
- the channels outputs are zero;
- the channels pins are not controlled by FTM (ELS(n)B:ELS(n)A = 0:0) (See the table in the description of CnSC register).

The following figure shows the FTM behavior after the reset. At the reset (item 1), the FTM counter is disabled (see the description of the CLKS field in the Status and Control register), its value is updated to zero and the pins are not controlled by FTM (See the table in the description of CnSC register).

After the reset, the FTM should be configurated (item 2). It is necessary to define the FTM counter mode, the FTM counting limits (MOD and CNTIN registers value), the channels mode and CnV registers value according to the channels mode.

Thus, it is recommended to write any value to CNT register (item 3). This write updates the FTM counter with the CNTIN register value and the channels output with its initial value (except for channels in output compare mode) (Counter reset).

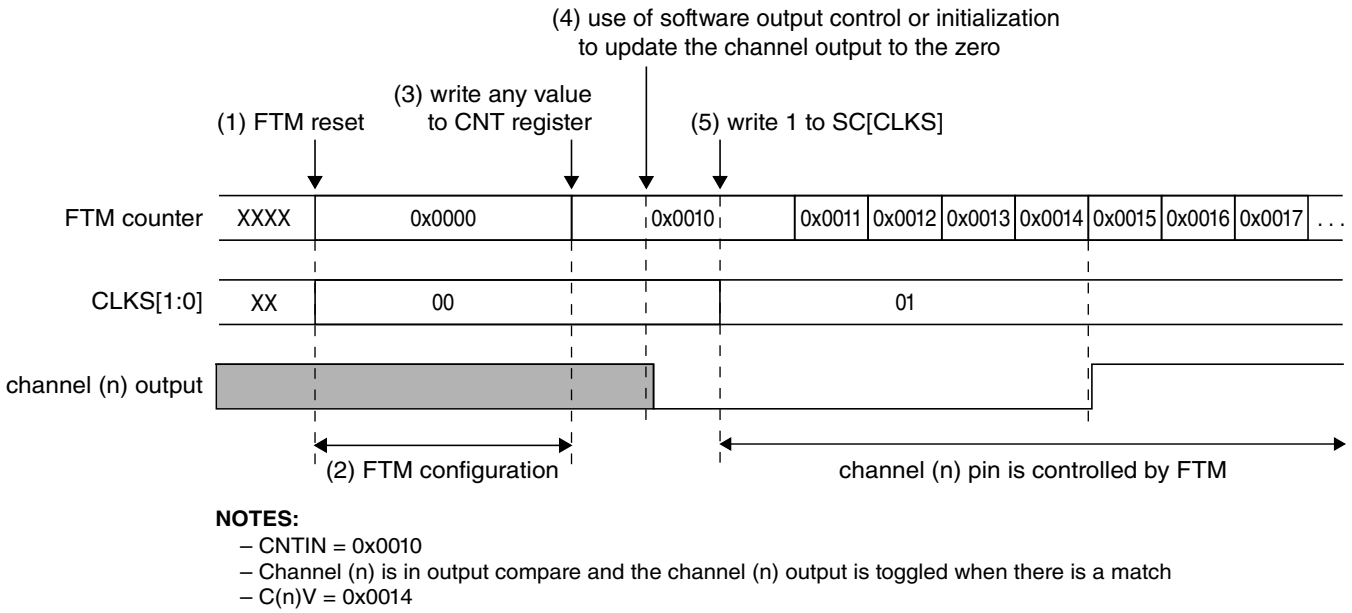
The next step is to select the FTM counter clock by the CLKS[1:0] bits (item 4). It is important to highlight that the pins are only controlled by FTM when CLKS[1:0] bits are different from zero (See the table in the description of CnSC register).



NOTES:
- CNTIN = 0x0010
- Channel (n) is in low-true combine mode with CNTIN < C(n)V < C(n+1)V < MOD
- C(n)V = 0x0015

Figure 19-96. FTM behavior after reset when the channel (n) is in Combine mode

The following figure shows an example when the channel (n) is in Output Compare mode and the channel (n) output is toggled when there is a match. In the Output Compare mode, the channel output is not updated to its initial value when there is a write to CNT register (item 3). In this case, use the software output control (Software output control) or the initialization (Initialization) to update the channel output to the selected value (item 4).



NOTES:
- CNTIN = 0x0010
- Channel (n) is in output compare and the channel (n) output is toggled when there is a match
- C(n)V = 0x0014

Figure 19-97. FTM behavior after reset when the channel (n) is in Output Compare mode

19.7 FTM Interrupts

19.7.1 Timer Overflow Interrupt

The timer overflow interrupt is generated when (TOIE = 1) and (TOF = 1).

19.7.2 Channel (n) Interrupt

The channel (n) interrupt is generated when (CHnIE = 1) and (CHnF = 1).

19.7.3 Fault Interrupt

The fault interrupt is generated when (FAULTIE = 1) and (FAULTF = 1).

19.8 Initialization Procedure

The following initialization procedure is recommended to configure the FlexTimer operation. This procedure can also be used to do a new configuration of the FlexTimer operation.

- Define the POL bits.
- Mask the channels outputs using SYNCHOM = 0. Two clocks after the write to OUTMASK, the channels output are in the safe value.
- (Re)Configuration FTM counter and channels to generation of periodic signals - Disable the clock. If the selected mode is Quadrature Decoder, then disable this mode. Examples of the (re)configuration:
  - Write to MOD.
  - Write to CNTIN.
  - Select OC, EPWM, CPWM, Combine, Complement modes for all channels that will be used
  - Select the high-true and low-true channels modes.
  - Write to CnV for all channels that will be used .
  - (Re)Configure deadtime and fault control.
  - Do not use the SWOC without SW synchronization (see item 6).
  - Do not use the Inverting without SW synchronization (see item 6).
  - Do not use the Initialization.
  - Do not change the polarity control.
  - Do not configure the HW synchronization
- Write any value to CNT. The FTM Counter is reset and the channels output are updated according to new configuration.
- Enable the clock. Write to CLKS[1:0] bits a value different from zero. If in the Quadrature Decoder mode, enable this mode.
- Configure the SW synchronization for SWOC (if it is necessary), Inverting (if it is necessary) and Output Mask (always)
  - Select synchronization for Output Mask Write to SYNC (SWSYNC = 0, TRIG2 = 0, TRIG1 = 0, TRIG0 = 0, SYNCHOM = 1, REINIT = 0, CNTMAX = 0, CNTMIN = 0)
  - Write to SYNCONF.
    - HW Synchronization can not be enabled (HWSOC = 0, HWINVC = 0, HWOM = 0, HWWRBUF = 0, HWRSTCNT = 0, HWTRIGMODE = 0).
    - SW Synchronization for SWOC (if it is necessary): SWSOC = [0/1] and SWOC = [0/1].
    - SW Synchronization for Inverting (if it is necessary): SWINVC = [0/1] and INVC = [0/1].
    - SW Synchronization for SWOM (always): SWOM = 1. No enable the SW Synchronization for write buffers (because the writes to registers with write buffer are done using CLKS[1:0] = 2’b00): SWWRBUF = 0 and CNTINC = 0 .
    - SW Synchronization for counter reset (always): SWRSTCNT = 1.
    - Enhanced synchronization (always): SYNCMODE = 1

    - If the SWOC is used (SWSOC = 1 and SWOC = 1), then write to SWOCTRL register.
    - If the Inverting is used (SWINVC = 1 and INVC = 1), then write to INVCTRL register.
    - Write to OUTMASK to enable the masked channels.
- Generate the Software Trigger Write to SYNC (SWSYNC = 1, TRIG2 = 0, TRIG1 = 0, TRIG0 = 0, SYNCHOM = 1, REINIT = 0, CNTMAX = 0, CNTMIN = 0)

Chapter 20
General Purpose I/O (GPIO)

20.1 The GPIO module as implemented on the chip

This section provides details about how the GPIO module is integrated into this chip.

Table 20-1. GPIO module as implemented on chip

Module name	Module base address	Supported[1] ports	Number of ports
GPIO1	0x230_0000	0:31	32
GPIO2	0x231_0000	0:7, 9:17	17

1. GPIO signals are typically multiplexed with other signals. “Supported” in this context means that any signal multiplexing configuration has selected GPIO functionality. See the Signals chapter for signal multiplexing details.

NOTE

Out of the 32 GPIO1 and 17 GPIO2 ports, some implementations may restrict specific ports to input-only, output-only, or reserved.

- Output only ports: GPIO1[0], GPIO1[4], GPIO1[5], GPIO1[11], GPIO1[12], GPIO1[20], GPIO1[23], GPIO1[29], GPIO1[31], GPIO2[0], and GPIO2[15]
- Input-only port: GPIO2[17]
- Reserved port: GPIO2[8]

20.2 GPIO overview

This chapter describes the general-purpose I/O (GPIO) module, including signal descriptions, register settings and interrupt capabilities.

This figure shows the block diagram for the GPIO module.

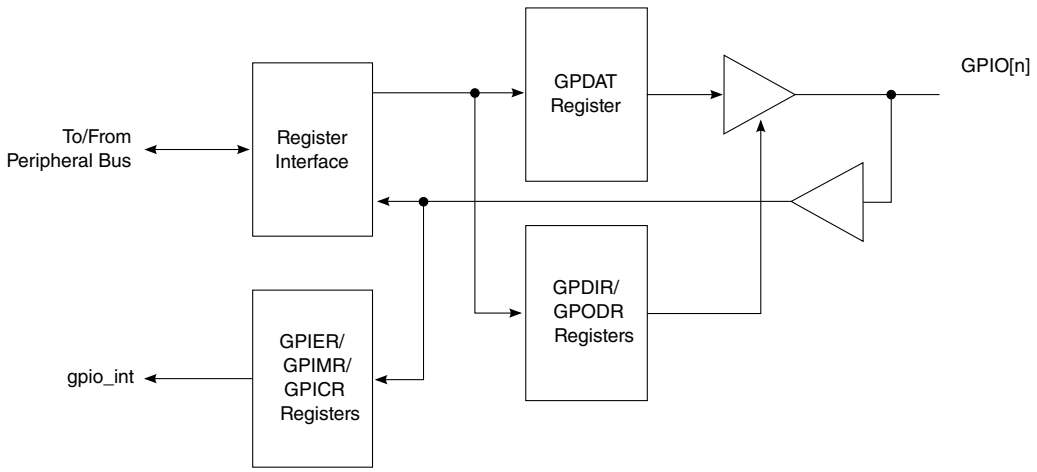


Figure 20-1. GPIO module block diagram

In general, the GPIO module supports up to 32 general-purpose I/O ports. Each port can be configured as an input or as an output. However, some implementations may restrict specific ports to input-only, output-only, or reserved (unimplemented). See "The GPIO module as implemented on the chip" section for more information. If a port is configured as an input, it can optionally generate an interrupt upon detection of a change. If a port is configured as an output, it can be individually configured as an open-drain or a fully active output.

20.3 GPIO features summary

The GPIO unit includes the following features:

- Supports 32 general-purpose input/output ports
- All signals are high-impedance during reset
- Open-drain capability on all ports
- All ports can optionally generate an interrupt upon changing their state
- Ports may be multiplexed with other functional signals

20.4 GPIO signal descriptions

This table provides detailed descriptions of the external GPIO signals. Note that depending on the signal multiplexing, some signals may not be available.

Table 20-2. GPIO external signals-detailed signal descriptions

Signal	I/O	Description	
GPIO*n*[0:31]	I/O	General Purpose I/O. Each signal can be individually set to act as input or output, according to application needs. Some implementations may restrict specific ports to input-only, output-only, or reserved (unimplemented). See "The GPIO module as implemented on the chip" section for more information.	
		State Meaning	Asserted/Negated-Defined per application.
		Timing	Assertion/Negation-Inputs can be asserted completely asynchronously. Outputs are asynchronous to any externally visible clock.

20.5 GPIO register descriptions

The programmable register map for the GPIO module occupies 28 bytes of memory-mapped space. The full register address is comprised of the base address (specified in CCSR address space) plus the module base address, plus the specific register's offset within the module. The table below shows the memory map for the GPIO module.

All GPIO registers are 32 bits wide located on 32-bit address boundaries. Note that reading undefined portions of the memory map returns all zeros and writing has no effect.

20.5.1 GPIO Memory map

GPIO1 base address: 230_0000h

GPIO2 base address: 231_0000h

Offset	Register	Width (In bits)	Access	Reset value
0h	GPIO direction register (GPDIR)	32	RW	0000_0000h
4h	GPIO open drain register (GPODR)	32	RW	0000_0000h

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
8h	GPIO data register (GPDAT)	32	RW	0000_0000h
Ch	GPIO interrupt event register (GPIER)	32	W1C	See description.
10h	GPIO interrupt mask register (GPIMR)	32	RW	0000_0000h
14h	GPIO interrupt control register (GPICR)	32	RW	0000_0000h

20.5.2 GPIO direction register (GPDIR)

20.5.2.1 Offset

Register	Offset
GPDIR	0h

20.5.2.2 Function

The GPIO direction register (GPDIR) defines the direction of the individual ports.

20.5.2.3 Diagram

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

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

20.5.2.4 Fields

Field	Function
0-31 DRn	Direction. Indicates whether a pin is used as an input or an output. 00000000000000000000000000000000b - The corresponding pin is an input. 00000000000000000000000000000001b - The corresponding pin is an output.

20.5.3 GPIO open drain register (GPODR)

20.5.3.1 Offset

Register	Offset
GPODR	4h

20.5.3.2 Function

The GPIO open drain register (GPODR) defines the way individual ports drive their output.

20.5.3.3 Diagram

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

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

20.5.3.4 Fields

Field	Function
0-31 ODn	Open drain configuration. Indicates whether a signal is actively driven as an output or is an open-drain driver. This register has no effect on signals programmed as inputs in GPDIR. 00000000000000000000000000000000b - The corresponding signal is actively driven as an output. 00000000000000000000000000000001b - The corresponding signal is an open-drain driver. As an output, the signal is driven active-low, otherwise it is not driven (high impedance).

20.5.4 GPIO data register (GPDAT)

20.5.4.1 Offset

Register	Offset
GPDAT	8h

20.5.4.2 Function

The GPIO data register (GPDAT) carries the data in/out for the individual ports.

20.5.4.3 Diagram

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

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

20.5.4.4 Fields

Field	Function
0-31 Dn	Data. Writes to this register latches the data which is presented on the external pins provided the corresponding GPDIR bit is configured as an output. When GPDIR is in output mode, GPDAT read operation returns data at pin. When GPDIR is in input mode, GPDAT read operation returns state of the port.

20.5.5 GPIO interrupt event register (GPIER)

20.5.5.1 Offset

Register	Offset
GPIER	Ch

20.5.5.2 Function

The GPIO interrupt event register (GPIER) carries information of the events that caused an interrupt. Each bit in GPIER, corresponds to an interrupt source. GPIER bits are cleared by writing ones. However, writing zero has no effect.

NOTE

Some implementations may ignore the interrupt mask as configured in GPIMR. In these implementations, a GPIER bit can be set even though the associated interrupt is masked. See The "GPIO module as implemented on the chip" section for more information.

20.5.5.3 Diagram

Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EVn															
W	W1C															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	EVn															
W	W1C															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

20.5.5.4 Fields

Field	Function
0-31 EVn	Interrupt events. Indicates whether an interrupt event occurred on the corresponding GPIO signal. 00000000000000000000000000000000b - No interrupt event occurred on the corresponding GPIO signal. 00000000000000000000000000000001b - An interrupt event occurred on the corresponding GPIO signal.

20.5.6 GPIO interrupt mask register (GPIMR)

20.5.6.1 Offset

Register	Offset
GPIMR	10h

20.5.6.2 Function

The GPIO interrupt mask register (GPIMR) defines the interrupt masking for the individual ports. When a masked interrupt request occurs, the corresponding GPIER bit is set, regardless of the GPIMR state. When one or more non-masked interrupt events occur, the GPIO module issues an interrupt to the interrupt controller.

20.5.6.3 Diagram

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

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

20.5.6.4 Fields

Field	Function
0-31 IMn	Interrupt mask. Indicates whether an interrupt event is masked or not masked for the corresponding GPIO signal. 00000000000000000000000000000000b - The input interrupt signal is masked (disabled). 00000000000000000000000000000001b - The input interrupt signal is not masked (enabled).

20.5.7 GPIO interrupt control register (GPICR)

20.5.7.1 Offset

Register	Offset
GPICR	14h

20.5.7.2 Function

The GPIO interrupt control register (GPICR) determines whether the corresponding port line asserts an interrupt request upon either a high-to-low change or any change on the state of the signal.

20.5.7.3 Diagram

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

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

20.5.7.4 Fields

Field	Function
0-31 EDn	Edge detection mode. The corresponding port line asserts an interrupt request according to the following: 00000000000000000000000000000000b - Any change on the state of the port generates an interrupt request. 00000000000000000000000000000001b - High-to-low change on the port generates an interrupt request.

Chapter 21
Inter-Integrated Circuit (I2C)

21.1 The I^{2}C module as implemented on the chip

This section provides details about how the I^{2}C module is implemented on the chip.

21.1.1 LS1012A I^{2}C module integration

The following table describes the I^{2}C module integration into the chip:

Table 21-1. I^{2}C module integration

Module	Module Base address
I2C1	218_0000
I2C2	219_0000

Additionally, both modules have been integrated with identical parameters and connections.

The remainder of this chapter refers to a single I^{2}C module. Notes are included to indicate variations for multiple instantiations.

21.1.2 LS1012A I^{2}C signals

The following table lists the SoC signal names and their corresponding I^{2}C module signal names used in this chapter:

Table 21-2. LS1012A I^{2}C signals

LS1012A signal name	I^{2}C module signal
IIC*n*_SCL	SCL
IIC*n*_SDA	SDA

21.1.3 LS1012A I²C module special consideration

The I²C module implements the following parameter settings in the chip:

Table 21-3. LS1012A I²C parameter settings

I²C parameters	LS1012A parameter value
Stop mode support	Yes. Refers to LPM20 low power mode of the chip.
Glitch filter	With the glitch filter enabled, the chip filters out any glitch on the SCL or SDA line that may cause the I2C module to enter an unrecoverable state. The glitch filter can be enabled using the I2C debug mode control register (I2CDBGCR). When I2C1/I2C2= 1, any pulse (rising or falling) on the SCL or SDA input line that is less than 7 IPG clock cycles is filtered out. When I2C1/I2C2= 0, the pulse is allowed/passed through. So, in cases where a glitch is not expected at the SCL or SDA inputs, you should not program I2C1/I2C2 = 1.

The table below provides the clock sources to each I²C module:

Table 21-4. LS1012A I²C clocking

Module	LS1012A clocking source
I2C1	Platform clock/2
I2C2	Platform clock/2

21.2 Overview

This chapter describes the Inter-Integrated Circuit (I²C) bus module implemented on this chip and presents the following topics:

- Introduction to I²C
- External signal descriptions
- Memory map and register definition
- Functional description
- Initialization/application information

21.3 Introduction to I²C

This section presents the following topics:

- Definition: I²C module
- Advantages of the I²C bus
- Module block diagram
- Features
- Modes of operation
- Definition: I²C conditions

21.3.1 Definition: I²C module

The I²C module is a functional unit that provides a two-wire— serial data (SDA) and serial clock (SCL) — bidirectional serial bus that provides a simple and efficient method of data exchange between this chip and other devices, such as microcontrollers, EEPROMs, real-time clock devices, analog-to-digital converters, and LCDs.

21.3.2 Advantages of the I²C bus

The synchronous, multiple-master two-wire I²C bus:

- Minimizes interconnections between devices
- Allows the connection of additional devices to the bus for expansion and system development
- Includes collision detection and arbitration that prevent data corruption if two or more masters attempt to control the bus simultaneously
- Does not require an external address decoder

21.3.3 Module block diagram

The following figure shows a block diagram of the I²C module.

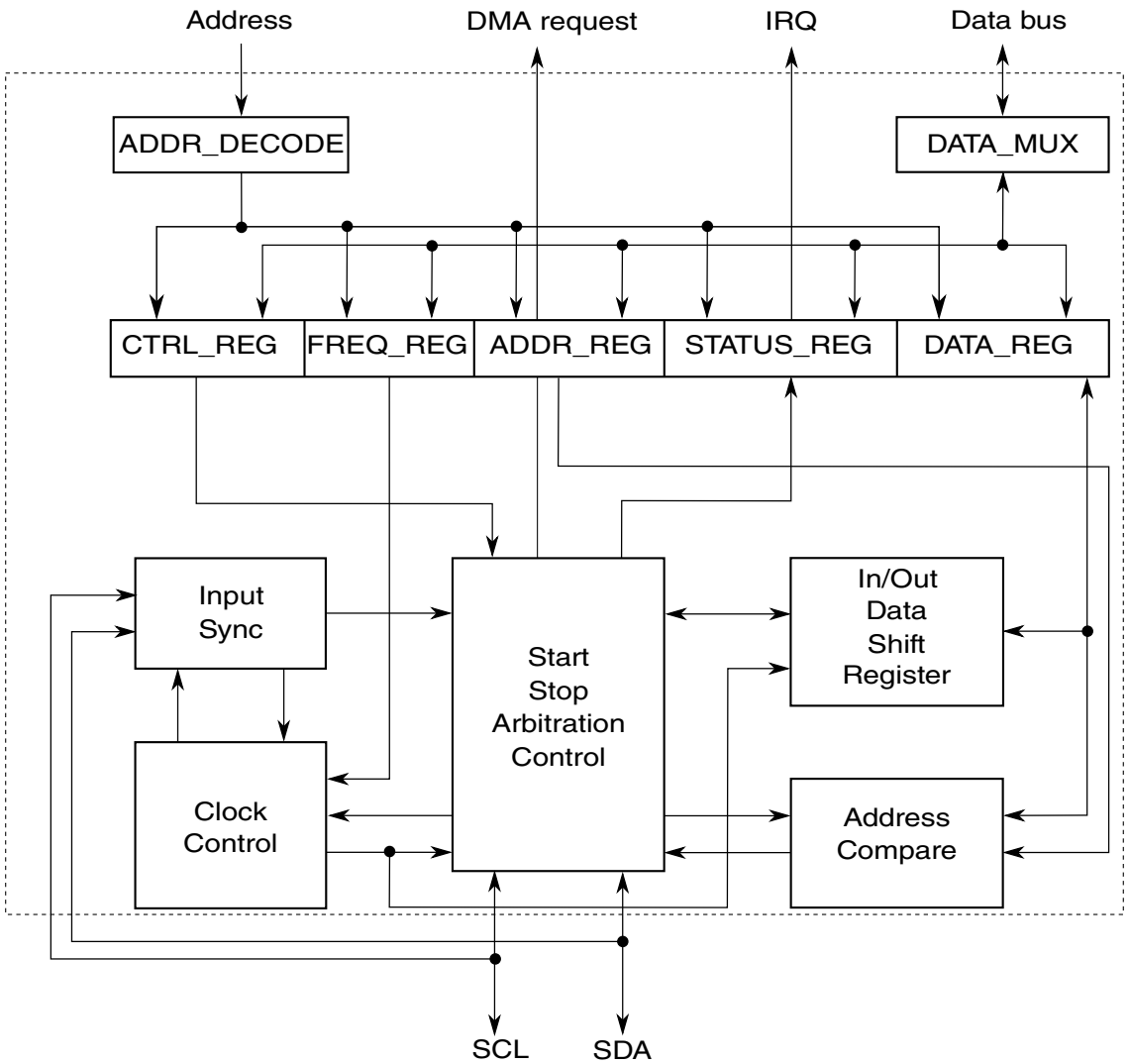


Figure 21-1. I²C block diagram

21.3.4 Features

The I²C module has the following key features:

- Compatible with I²C bus standard[1]
- Operating speeds
  - Up to 100 kbps in Standard Mode
  - Up to 400 kbps in Fast Mode

1. Compliant with I²C 2.0 standard with the exception that HS (high speed) mode is not supported

  - Operation at higher baud rates (up to a maximum of module clock/20) with reduced bus loading
  - Actual baud rate dependent on the SCL rise time (which depends on external pull-up resistor values and bus loading)
- Multi-master operation
- Software programmable for one of 256 different serial clock frequencies
- Software selectable acknowledge bit
- Interrupt driven byte-by-byte data transfer
- Arbitration lost interrupt with automatic mode switching from master to slave
- Calling address identification interrupt
- Start and stop signal generation/detection
- Repeated start signal generation
- Acknowledge bit generation/detection
- Bus busy detection
- Basic DMA interface
- Maximum communication length and the number of devices that can be connected are limited by a maximum bus capacitance of 400 pF

21.3.5 Modes of operation

The I^2C module supports the chip modes described in the following table.

Table 21-5. Chip modes supported by the I^2C module

Chip mode	Description	Important notes
RUN	Basic mode of operation	—
DOZE	A low-power mode that allows the system to turn off the clock depending on the state of an internal bit	The I^2C module can enter this mode when there are no active transfers (active data between valid START and STOP conditions) on the bus.
STOP	The lowest-power mode that allows the chip to turn off all the clocks to the I^2C module	The I^2C module can enter this mode when there are no active transfers (active data between valid START and STOP conditions) on the bus. See STOP mode.

In addition to chip modes, the I^{2}C module has several module-specific modes. These are described in the following table.

Table 21-6. Module-specific modes supported by the I^{2}C module

Module mode	Description	Important notes
Master mode	The I^{2}C module is the driver of the SDA line.	• Do not use the I^{2}C module's slave address as a calling address. • The I^{2}C module cannot be a master and a slave simultaneously.
Slave mode	The I^{2}C module is not the driver of the SDA line.	• Enable the I^{2}C module before a START condition from a non-I^{2}C master is detected. • By default the I^{2}C module performs as a slave receiver.

21.3.6 Definition: I^{2}C conditions

The following table shows the I^{2}C-specific conditions defined for the I^{2}C module.

Table 21-7. I^{2}C Conditions

Condition	Description
START	A condition that denotes the beginning of a new data transfer and awakens all slaves. Each data transfer contains several bytes of data. It is defined as a high-to-low transition of SDA while SCL is high, as shown in the following figure.
STOP	A condition generated by the master to terminate a transfer and free the bus. It is defined as a low-to-high transition of SDA while SCL is high, as shown in the following figure.
Repeated START	A START condition that is generated without a STOP condition to terminate the previous transfer.

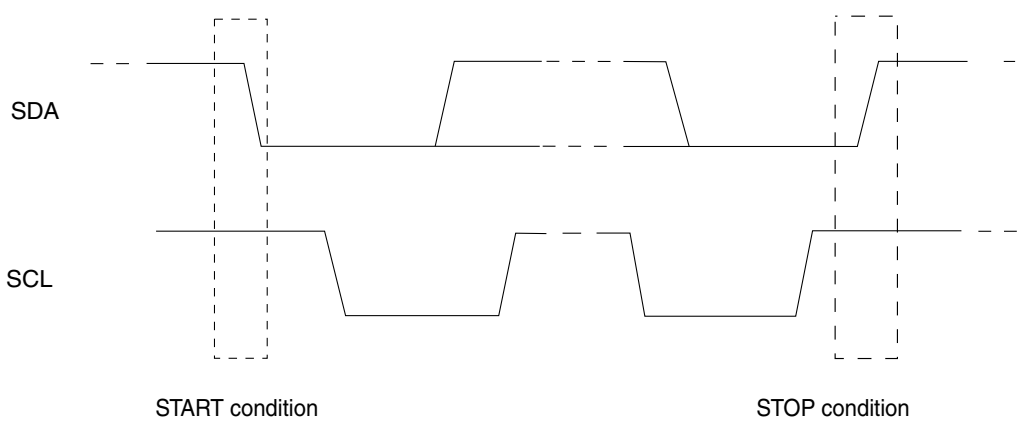


Figure 21-2. START and STOP conditions

21.4 External signal descriptions

This section presents the following topics:

- Signal overview
- Detailed external signal descriptions

21.4.1 Signal overview

The I^2C module uses the Serial Data (SDA) and Serial Clock (SCL) signals as a communication interconnect with other devices. The signal patterns driven on the SDA signal represent address, data, or read/write information at different stages of the protocol.

All devices connected to the SDA and SCL signals must have open-drain or open-collector outputs. The logical AND function is performed on both signals with external pull-up resistors. For the electrical characteristics of these signals, see the data sheet for this chip.

21.4.2 Detailed external signal descriptions

The SDA and SCL signals are described in the following table.

Table 21-8. External signal descriptions

Signal	Description
SCL	Bidirectional serial clock line of the module, compatible with the I^2C bus specification
SDA	Bidirectional serial data line of the module, compatible with the I^2C bus specification

21.5 Memory map and register definition

This section provides a detailed description of all memory-mapped registers in the I^2C module. It presents the following topics:

- Register accessibility
- Register figure conventions

- I2C Bus Address Register (I2C_IBAD)
- I2C Bus Frequency Divider Register (I2C_IBFD)
- I2C Bus Control Register (I2C_IBCR)
- I2C Bus Status Register (I2C_IBSR)
- I2C Bus Data I/O Register (I2C_IBDR)
- I2C Bus Interrupt Config Register (I2C_IBIC)

21.5.1 Register accessibility

Address location 0x0007 is a reserved location, but access to this location will not generate any bus error.

All the I^2C registers are one byte wide. Reads and writes to these registers must be byte-wide operations.

21.5.2 Register figure conventions

The register figures show the field structure using the conventions in the following figure.

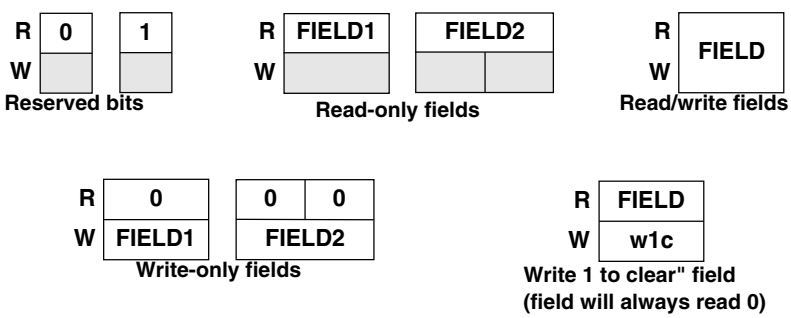


Figure 21-3. Register figure convention

The memory map for the I^2C module is given below. The total address for each register is the sum of the base address for the I^2C module and the address offset for each register.

I2C memory map

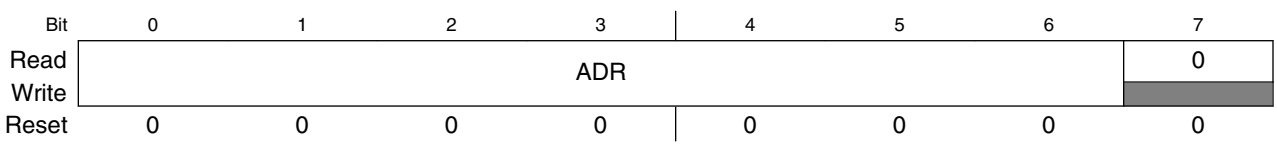
Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
218_0000	I2C Bus Address Register (I2C1_IBAD)	8	R/W	00h	21.5.3/1005
218_0001	I2C Bus Frequency Divider Register (I2C1_IBFD)	8	R/W	00h	21.5.4/1006
218_0002	I2C Bus Control Register (I2C1_IBCR)	8	R/W	80h	21.5.5/1006
218_0003	I2C Bus Status Register (I2C1_IBSR)	8	R/W	80h	21.5.6/1008
218_0004	I2C Bus Data I/O Register (I2C1_IBDR)	8	R/W	00h	21.5.7/1009
218_0005	I2C Bus Interrupt Config Register (I2C1_IBIC)	8	R/W	00h	21.5.8/1010
219_0000	I2C Bus Address Register (I2C2_IBAD)	8	R/W	00h	21.5.3/1005
219_0001	I2C Bus Frequency Divider Register (I2C2_IBFD)	8	R/W	00h	21.5.4/1006
219_0002	I2C Bus Control Register (I2C2_IBCR)	8	R/W	80h	21.5.5/1006
219_0003	I2C Bus Status Register (I2C2_IBSR)	8	R/W	80h	21.5.6/1008
219_0004	I2C Bus Data I/O Register (I2C2_IBDR)	8	R/W	00h	21.5.7/1009
219_0005	I2C Bus Interrupt Config Register (I2C2_IBIC)	8	R/W	00h	21.5.8/1010

21.5.3 I2C Bus Address Register (I2Cx_IBAD)

This register contains the address the I^2C Bus will respond to when addressed as a slave. This is not the address sent on the bus during the address transfer.

Access: Supervisor mode only (user mode accesses are ignored, and no error response is asserted while accessing this register in user mode)

Address: Base address + 0h offset



I2Cx_IBAD field descriptions

Field	Description
0–6 ADR	Slave Address. Specific slave address to be used by the I^2C Bus module. **NOTE:** The default mode of I^2C Bus is slave mode for an address match on the bus.
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

21.5.4 I2C Bus Frequency Divider Register (I2Cx_IBFD)

Access: Supervisor mode only (user mode accesses are ignored, and no error response is asserted while accessing this register in user mode)

Address: Base address + 1h offset

Bit	0	1	2	3	4	5	6	7
Read Write	IBC							
Reset	0	0	0	0	0	0	0	0

I2Cx_IBFD field descriptions

Field	Description
0–7 IBC	I-Bus Clock Rate. This field is used to prescale the bus clock for bit rate selection. See Clock rate and IBFD settings.

21.5.5 I2C Bus Control Register (I2Cx_IBCR)

Access: Supervisor mode only (user mode accesses are ignored, and no error response is asserted while accessing this register in user mode)

Address: Base address + 2h offset

Bit	0	1	2	3	4	5	6	7
Read	MDIS	IBIE	MSSL	TXRX	NOACK	0	DMAEN	IBDOZE
Write						RSTA		
Reset	1	0	0	0	0	0	0	0

I2Cx_IBCR field descriptions

Field	Description
0 MDIS	Module disable. This bit controls the software reset of the entire I^2C Bus module. **NOTE:** If the I^2C Bus module is enabled in the middle of a byte transfer, the interface behaves as follows: slave mode ignores the current transfer on the bus and starts operating whenever a subsequent start condition is detected. Master mode will not be aware that the bus is busy, hence if a start cycle is initiated then the current bus cycle may become corrupt. This would ultimately result in either the current bus master or the I^2C Bus module losing arbitration, after which, bus operation would return to normal.

Table continues on the next page...

I2Cx_IBCR field descriptions (continued)

Field	Description
	0 The I^2C Bus module is enabled. This bit must be cleared before any other IBCR bits have any effect 1 The module is reset and disabled. This is the power-on reset situation. When high, the interface is held in reset, but registers can still be accessed. Status register bits (IBSR) are not valid when module is disabled.
1 IBIE	I-Bus Interrupt Enable. 0 Interrupts from the I^2C Bus module are disabled. This does not clear any currently pending interrupt condition. 1 Interrupts from the I^2C Bus module are enabled. An I^2C Bus interrupt occurs provided the IBIF bit in the status register is also set.
2 MSSL	Master/Slave mode select. When this bit is changed from 0 to 1, a START signal is generated on the bus and the master mode is selected. When this bit is changed from 1 to 0, a STOP signal is generated and the operation mode changes from master to slave. A STOP signal should be generated only if the IBIF flag is set. This field is cleared without generating a STOP signal when the master loses arbitration. 0 Slave Mode 1 Master Mode
3 TXRX	Transmit/Receive mode select. This bit selects the direction of master and slave transfers. When addressed as a slave this bit should be set by software according to the SRW bit in the status register. In master mode this bit should be set according to the type of transfer required. Therefore, for address cycles, this bit will always be high. 0 Receive 1 Transmit
4 NOACK	Data Acknowledge disable. This bit specifies the value driven onto SDA during data acknowledge cycles for both master and slave receivers. The I^2C module will always acknowledge address matches, provided it is enabled, regardless of the value of NOACK. **NOTE:** Values written to this bit are only used when the I^2C Bus is a receiver, not a transmitter. 0 An acknowledge signal will be sent out to the bus at the 9th clock bit after receiving one byte of data 1 No acknowledge signal response is sent (i.e., acknowledge bit = 1)
5 RSTA	Repeat Start. Writing a one to this bit will generate a repeated START condition on the bus, provided it is the current bus master. This bit will always be read as a low. Attempting a repeated start at the wrong time, if the bus is owned by another master, will result in loss of arbitration. 0 No effect 1 Generate repeat start cycle
6 DMAEN	DMA Enable. When this bit is set, the DMA Tx and Rx lines will be asserted when the I^2C module requires data to be read or written to the data register. No Transfer Done interrupts will be generated when this bit is set, however an interrupt will be generated if the loss of arbitration or addressed as slave conditions occur. The DMA mode is only valid when the I^2C module is configured as a Master and the DMA transfer still requires CPU intervention at the start and the end of each frame of data. See the DMA Application Information section for more details. 0 Disable the DMA TX/RX request signals 1 Enable the DMA TX/RX request signals
7 IBDOZE	I-Bus Interface Stop in DOZE mode. If the IBDOZE mode is SET, the I^2C module will enter DOZE mode when the DOZE signal is asserted, if there are no current transactions on the bus. The I^2C module would then signal to the system that the clock can be shut down.

Table continues on the next page...

I2Cx_IBCR field descriptions (continued)

Field	Description
	If the IBDOZE bit is cleared when the DOZE signal is asserted, the I ^{2} C Bus module clock remains alive, and any current transactions continue as normal. 0 I ^{2} C Bus module clock operates normally 1 Halt I ^{2} C Bus module clock generation (if DOZE mode signal asserted)

21.5.6 I2C Bus Status Register (I2Cx_IBSR)

Access: Supervisor mode only (user mode accesses are ignored, and no error response is asserted while accessing this register in user mode)

Address: Base address + 3h offset

Bit	0	1	2	3	4	5	6	7
Read	TCF	IAAS	IBB	IBAL	0	SRW	IBIF	RXAK
Write				w1c			w1c	
Reset	1	0	0	0	0	0	0	0

I2Cx_IBSR field descriptions

Field	Description
0 TCF	Transfer complete. While one byte of data is being transferred, this bit is cleared. It is set by the falling edge of the 9th clock of a byte transfer. **NOTE:** This bit is only valid during or immediately following a transfer to the I^{2}C module or from the I^{2}C module. 0 Transfer in progress 1 Transfer complete
1 IAAS	Addressed as a slave. When its own specific address (I-Bus Address Register) is matched with the calling address, this bit is set. The CPU is interrupted provided the IBIE is set. Then the CPU needs to check the SRW bit and set the TXRX field accordingly. Writing to the I-Bus Control Register clears this bit. 0 Not addressed 1 Addressed as a slave
2 IBB	Bus busy. This bit indicates the status of the bus. When a START signal is detected, IBB is set. If a STOP signal is detected, IBB is cleared and the bus enters idle state. **NOTE:** Software must ensure that the I^{2}C bus is idle by checking the IBSR[IBB] field (bus busy) before switching to master mode and attempting a START cycle.

Table continues on the next page...

I2Cx_IBSR field descriptions (continued)

Field	Description
	0 Bus is Idle 1 Bus is busy
3 IBAL	Arbitration Lost. The arbitration lost bit (IBAL) is set by hardware when the arbitration procedure is lost. Arbitration is lost in the following circumstances: • SDA is sampled low when the master drives a high during an address or data transmit cycle. • SDA is sampled low when the master drives a high during the acknowledge bit of a data receive cycle. • A start cycle is attempted when the bus is busy. • A repeated start cycle is requested in slave mode. • A stop condition is detected when the master did not request it. This bit must be cleared by software, by writing a one to it. A write of zero has no effect.
4 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5 SRW	Slave Read/Write. When the IAAS bit is set, this bit indicates the value of the R/W command bit of the calling address sent from the master. This bit is only valid when the I-Bus is in slave mode, a complete address transfer has occurred with an address match and no other transfers have been initiated. By reading this field, the CPU can detect slave transmit/receive mode according to the command of the master. 0 Slave receive, master writing to slave 1 Slave transmit, master reading from slave
6 IBIF	I-Bus Interrupt Flag. The IBIF bit is set when one of the following conditions occurs: • Arbitration lost (IBAL bit set) • Byte transfer complete (TCF bit set and DMAEN bit not set) • Addressed as slave (IAAS bit set) • NoAck from Slave (MS & Tx bits set) • I^2C Bus going idle (IBB high-low transition and enabled by BIIE) A processor interrupt request will be caused if the IBIE bit is set. This bit must be cleared by software, by writing a one to it. A write of zero has no effect on this bit. In DMA mode (DMAEN set) a byte transfer complete condition will not trigger the setting of IBIF. All other conditions still apply.
7 RXAK	Received Acknowledge. This is the value of SDA during the acknowledge bit of a bus cycle. If the received acknowledge bit (RXAK) is low, it indicates an acknowledge signal has been received after the completion of 8 bits data transmission on the bus. If RXAK is high, it means no acknowledge signal is detected at the 9th clock. This bit is valid only after transfer is complete. 0 Acknowledge received 1 No acknowledge received

21.5.7 I2C Bus Data I/O Register (I2Cx_IBDR)

In master transmit mode, when data is written to the IBDR, a data transfer is initiated. The most significant bit is sent first. In master receive mode, reading this register initiates next byte data receiving. In slave mode, the same functions are available after an address match has occurred.

NOTE

The IBCR[TXRX] field must correctly reflect the desired direction of transfer in master and slave modes for the transmission to begin. For instance, if the I^2C is configured for master transmit but a master receive is desired, then reading the IBDR will not initiate the receive.

Reading the IBDR will return the most recent byte received while the I^2C is configured in either master receive or slave receive modes. The IBDR does not reflect every byte that is transmitted on the I^2C bus, nor can software verify that a byte has been written to the IBDR correctly by reading it back.

In master transmit mode, the first byte of data written to IBDR following assertion of MSSL is used for the address transfer and should comprise the calling address (in position DATA[7:1]) concatenated with the required R/ $\overline{W}$ bit (in position D0).

NOTE

When the I^2C is configured in master mode and receiving data from a slave that is transmitting data bytes on an irregular basis, the master cannot know whether the data received in the IBDR is the old latched data or the new data received from the slave. To avoid this, 2 consecutive intermittent data bytes from slave should be different.

Access: Supervisor mode only (user mode accesses are ignored, and no error response is asserted while accessing this register in user mode)

Address: Base address + 4h offset

Bit	0	1	2	3	4	5	6	7
Read Write	DATA							
Reset	0	0	0	0	0	0	0	0

I2Cx_IBDR field descriptions

Field	Description
0–7 DATA	Data transmitted or received

21.5.8 I2C Bus Interrupt Config Register (I2Cx_IBIC)

To program BIIE = 1, you must ensure that IBCR[MDIS] = 0.

Access: Supervisor mode only (user mode accesses are ignored, and no error response is asserted while accessing this register in user mode)

Address: Base address + 5h offset

Bit	0	1	2	3	4	5	6	7
Read	BIIE	BYTERXIE	0					
Write								
Reset	0	0	0	0	0	0	0	0

I2Cx_IBIC field descriptions

Field	Description
0 BIIE	Bus Idle Interrupt Enable bit. This config bit can be used to enable the generation of an interrupt once the I^2C bus becomes idle. Once this bit is set, an IBB high-low transition will set the IBIF bit. This feature can be used to signal to the CPU the completion of a STOP on the I^2C bus. 0 Bus Idle Interrupts disabled 1 Bus Idle Interrupts enabled
1 BYTERXIE	Byte receive interrupt enable This field is used to generate an interrupt every time the I^2C master/slave receives a new byte. This feature can be useful when an I^2C master is receiving data from a slave that is transmitting on an irregular basis. BYTERXIE is updated only when the I^2C is enabled (IBCR[MDIS]=0).
2–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

21.6 Functional description

This section presents the following topics:

- Notes about module operation
- Transactions
- Arbitration procedure
- Clock behavior
- Interrupts
- DMA interface

21.6.1 Notes about module operation

- The I^2C module always performs as a slave receiver by default, unless explicitly programmed to be a master or slave transmitter.
- When the I^2C module is acting as a master, it must not try to call its own slave address.

21.6.2 Transactions

This section presents the following topics:

- Protocol overview
- Transaction protocol definitions
- High-level protocol steps
- START condition
- Slave address transmission
- Data transmission
- STOP condition
- Repeated START condition

21.6.2.1 Protocol overview

The following figure shows the behavior of SCL and SDA during a typical I^2C transaction.

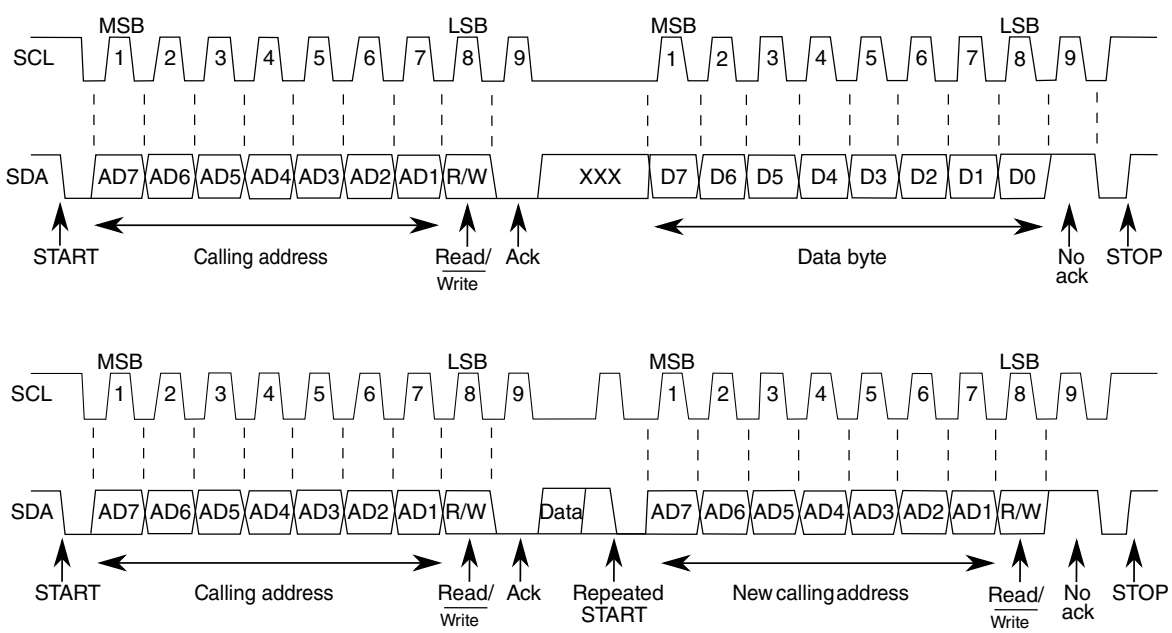


Figure 21-4. I^2C transaction protocol

21.6.2.2 Transaction protocol definitions

This section defines several important terms presented in Figure 21-4.

Table 21-9. I²C definitions

Term	Definition
START	A START condition, as defined in Section 1.2.6, Definition: I²C conditions"
STOP	A STOP condition, as defined in Section 1.2.6, Definition: I²C conditions"
Calling (slave) address	A seven-bit address used to identify a slave on the I²C bus. The requirements for specifying this address are presented in Section 1.5.2.3, I²C calling address requirements."
Read/write (R/W)	A bit that specifies the direction of the data transfer to the slave as follows: • 0=The data is being transferred from the master to the slave ("write") • 1=The data is being transferred from the slave to the master ("read")
Ack	A bit that specifies the acknowledgement of a calling address, indicated by pulling SDA low.

21.6.2.3 I²C calling address requirements

The calling addresses of the devices used on an I²C network are subject to the following requirements:

- Each slave must have a unique calling address.
- A master must not transmit a calling address that is the same as its own slave address.

21.6.2.4 High-level protocol steps

The I²C protocol conceptually supports two types of transfers, which are illustrated in Figure 21-4. The significant steps in these transfers are presented in the following table. Details of each of these steps are presented in subsequent sections.

Table 21-10. I²C high-level protocol steps

Standard Transfer	Repeated START Transfer
1. START condition	1. START condition
2. Slave target or general call address transmission	2. Slave target or general call address transmission
3. Acknowledgment from slave	3. Acknowledgment from slave

Table 21-10. I²C high-level protocol steps

Standard Transfer	Repeated START Transfer
4. Data transfer	4. Data transfer
5. STOP condition	5. Repeated START condition
6. (repeat Steps 1–4)	6. (repeat Steps 2–4 as needed)
	7. STOP condition.
	8. (repeat Steps 1–7)

21.6.2.5 START condition

When the bus is free, that is, no master device is engaging the bus (both SCL and SDA lines are at logical high), a master may initiate communication by sending a START condition (see Definition: I²C conditions). This signal denotes the beginning of a new data transfer (each data transfer may contain several bytes of data) and brings all slaves out of their idle states.

See Clock rate and IBFD settings and I2C Bus Frequency Divider Register (I2C_IBFD) for the associated timing requirements.

21.6.2.6 Slave address transmission

The master transmits the slave address on the next clock cycle after the START condition (see START condition). The process of slave address transmission is presented in the following table.

Table 21-11. Slave address transmission process

Step	Action
1	The master transmits the seven-bit slave address.
2	The master transmits the R/W bit.
3	Each slave examines the transmitted address and compares it to its own. If the addresses match, the slave device returns the acknowledge bit on the ninth SCL clock cycle.
4	The master waits for the acknowledge bit and determines the next step as follows: • The acknowledge bit is set: The master must initiate a data transfer followed by either a STOP condition or a repeated START condition. • The acknowledge bit is cleared: The master must wait for SCL to return to logic zero.

21.6.2.7 Data transmission

A data transfer session has the following characteristics:

- Can transmit one or more bytes of data
- Awakens all slaves
- Proceeds on a byte-by-byte basis in the direction specified by the R/W bit sent by the calling master

The transmitted data is subject to the following requirements:

- Each data byte must consist of 8 bits.
- Data bits can be changed only while SCL is low and must be held stable while SCL is high.
- One data bit is transmitted during one SCL clock pulse.
- The most significant bit (msb) must be transmitted first.
- Each data byte must be followed by an acknowledge bit on the ninth SCL clock pulse.

If the slave receiver does not acknowledge the master, the SDA line must be left high by the slave. The master can then generate a stop condition to abort the data transfer or a START condition (repeated START) to begin a new calling.

If the master receiver does not acknowledge the slave transmitter after a byte of transmission, the slave interprets that the end-of-data has been reached. Then the slave releases the SDA line for the master to generate a STOP or a START condition.

21.6.2.8 STOP condition

The master can terminate the communication by generating a STOP condition (see Definition: I^2C conditions). It can do so even if the slave has generated an acknowledge, at which point the slave must release the bus.

A master is not required to send a STOP condition at the end of every transfer. For more information, see Repeated START condition.

See Clock rate and IBFD settings and I2C Bus Frequency Divider Register (I2C_IBFD) for the associated timing requirements.

21.6.2.9 Repeated START condition

The I^2C protocol also supports a repeated START condition, which can be generated without a preceding STOP condition. A master device can use this condition to communicate with another slave or with the same slave in a different mode without releasing the bus. This condition is illustrated in the second timing diagram of Figure 21-4.

21.6.3 Arbitration procedure

The I^2C bus is a true multi-master bus that allows more than one master to be connected to it. If two or more masters try to control the bus at the same time, a clock synchronization procedure determines the bus clock, for which the low period is equal to the longest clock low period and the high is equal to the shortest one among the masters. The relative priority of the contending masters is determined by a data arbitration procedure. A bus master loses arbitration if it transmits logic "1" while another master transmits logic "0". The losing masters immediately switch over to slave mode and stop driving the SDA output. In this case, the transition from master to slave mode does not generate a STOP condition. Meanwhile, a status bit is set by hardware to indicate loss of arbitration.

21.6.4 Clock behavior

This section presents the following topics:

- Clock synchronization
- Clock stretching
- Handshaking
- Clock rate and IBFD settings

21.6.4.1 Clock synchronization

Due to the wired-AND logic on the SCL line, a high-to-low transition on the SCL line affects all devices connected on the bus. The devices begin counting their low period when the master drives the SCL line low. After a device has driven SCL low, it holds the SCL line low until the clock high state is reached.

However, the change of low-to-high in a device clock may not change the state of the SCL line if another device is still within its low period. Therefore, the synchronized clock signal, SCL, is held low by the device with the longest low period. Devices with shorter low periods enter a high wait state during this time (see the following figure). When all devices concerned have counted off their low period, the synchronized SCL line is released and pulled high. Then there is no difference between the devices' clocks and the state of the SCL line, and all the devices begin counting their high periods. The first device to complete its high period pulls the SCL line low again.

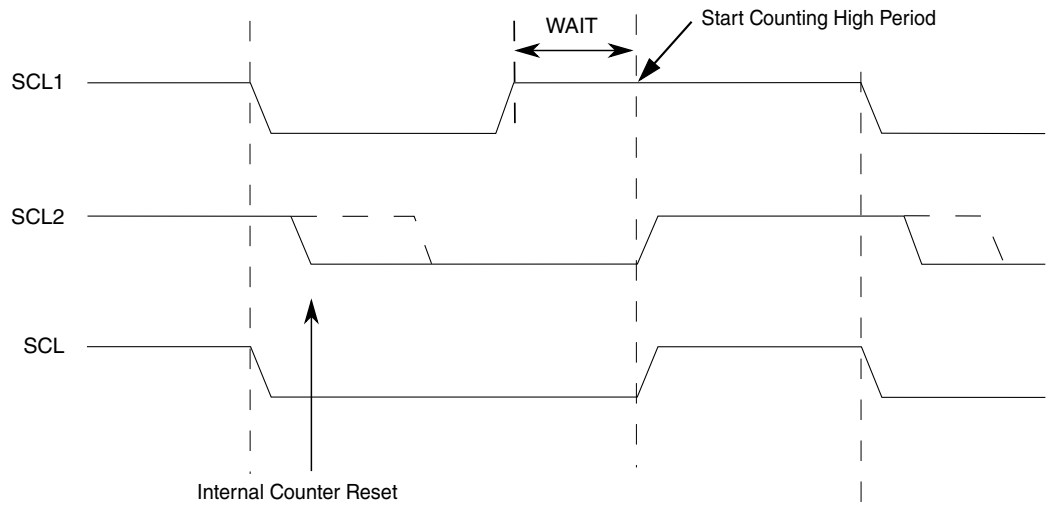


Figure 21-5. I²C bus clock synchronization

21.6.4.2 Clock stretching

The clock synchronization mechanism can be used by slaves to slow down the bit rate of a transfer. After the master has driven SCL low, the slave can drive SCL low for the required period and then release it. If the slave SCL low period is greater than the master SCL low period then the resulting SCL bus signal low period is stretched.

21.6.4.3 Handshaking

The clock synchronization mechanism can be used as a handshake in data transfer. Slave devices may hold the SCL low after completion of one byte transfer (9 bits). In such cases, it halts the bus clock and forces the master clock into wait state until the slave releases the SCL line.

21.6.4.4 Clock rate and IBFD settings

21.6.4.4.1 Timing definitions

Table 21-12. Timing definitions relevant to clock rate and IBFD settings

Term	Definition
SCL Divider	The factor used to prescale the CPU clock for bit rate selection (see Figure 21-6 and Table 21-13)
SCL period	(CPU clock period) × (SCL Divider)
SCL Hold	The required number of CPU clocks to generate a START or STOP condition (see Figure 21-6 and Table 21-13)
SDA Hold	See Figure 21-7 and Table 21-13

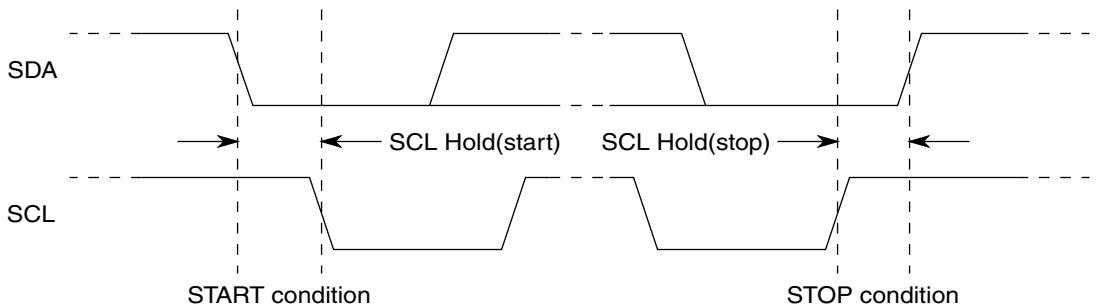


Figure 21-6. SCL Divider and SDA Hold

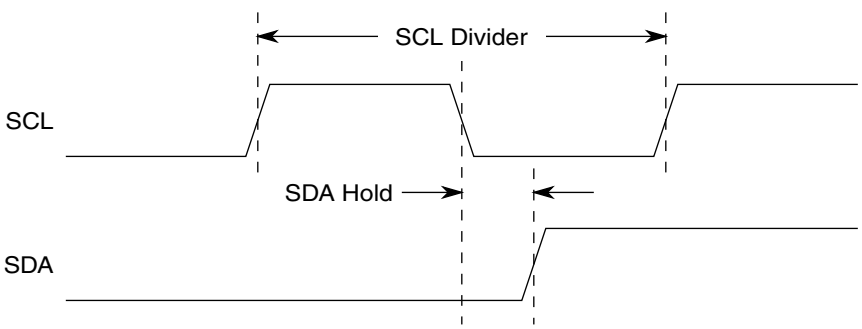


Figure 21-7. SDA Hold time

21.6.4.4.2 Divider and hold values

Table 21-13. I²C divider and hold values

	IBC (hex)	SCL Divider (clocks)	SDA Hold (clocks)	SCL Hold (start)	SCL Hold (stop)
MUL=1	00	20	7	6	11
	01	22	7	7	12
	02	24	8	8	13
	03	26	8	9	14
	04	28	9	10	15

Table continues on the next page...

Table 21-13. I²C divider and hold values (continued)

	IBC (hex)	SCL Divider (clocks)	SDA Hold (clocks)	SCL Hold (start)	SCL Hold (stop)
	05	30	9	11	16
	06	34	10	13	18
	07	40	10	16	21
	08	28	7	10	15
	09	32	7	12	17
	0A	36	9	14	19
	0B	40	9	16	21
	0C	44	11	18	23
	0D	48	11	20	25
	0E	56	13	24	29
	0F	68	13	30	35
	10	48	9	18	25
	11	56	9	22	29
	12	64	13	26	33
	13	72	13	30	37
	14	80	17	34	41
	15	88	17	38	45
	16	104	21	46	53
	17	128	21	58	65
	18	80	9	38	41
	19	96	9	46	49
	1A	112	17	54	57
	1B	128	17	62	65
	1C	144	25	70	73
	1D	160	25	78	81
	1E	192	33	94	97
	1F	240	33	118	121
	20	160	17	78	81
	21	192	17	94	97
	22	224	33	110	113
	23	256	33	126	129
MUL=1	24	288	49	142	145
	25	320	49	158	161
	26	384	65	190	193
	27	480	65	238	241
	28	320	33	158	161
	29	384	33	190	193
	2A	448	65	222	225

Table continues on the next page...

Table 21-13. I²C divider and hold values (continued)

	IBC (hex)	SCL Divider (clocks)	SDA Hold (clocks)	SCL Hold (start)	SCL Hold (stop)
	2B	512	65	254	257
	2C	576	97	286	289
	2D	640	97	318	321
	2E	768	129	382	385
	2F	960	129	478	481
	30	640	65	318	321
	31	768	65	382	385
	32	896	129	446	449
	33	1024	129	510	513
	34	1152	193	574	577
	35	1280	193	638	641
	36	1536	257	766	769
	37	1920	257	958	961
	38	1280	129	638	641
	39	1536	129	766	769
	3A	1792	257	894	897
	3B	2048	257	1022	1025
	3C	2304	385	1150	1153
	3D	2560	385	1278	1281
	3E	3072	513	1534	1537
	3F	3840	513	1918	1921
MUL=2	40	40	14	12	22
	41	44	14	14	24
	42	48	16	16	26
	43	52	16	18	28
	44	56	18	20	30
	45	60	18	22	32
	46	68	20	26	36
	47	80	20	32	42
	48	56	14	20	30
	49	64	14	24	34
MUL=2	4A	72	18	28	38
	4B	80	18	32	42
	4C	88	22	36	46
	4D	96	22	40	50
	4E	112	26	48	58
	4F	136	26	60	70
	50	96	18	36	50

Table continues on the next page...

Table 21-13. I²C divider and hold values (continued)

	IBC (hex)	SCL Divider (clocks)	SDA Hold (clocks)	SCL Hold (start)	SCL Hold (stop)
	51	112	18	44	58
	52	128	26	52	66
	53	144	26	60	74
	54	160	34	68	82
	55	176	34	76	90
	56	208	42	92	106
	57	256	42	116	130
	58	160	18	76	82
	59	192	18	92	98
	5A	224	34	108	114
	5B	256	34	124	130
	5C	288	50	140	146
	5D	320	50	156	162
	5E	384	66	188	194
	5F	480	66	236	242
	60	320	34	156	162
	61	384	34	188	194
	62	448	66	220	226
	63	512	66	252	258
	64	576	98	284	290
	65	640	98	316	322
	66	768	130	380	386
	67	960	130	476	482
	68	640	66	316	322
	69	768	66	380	386
	6A	896	130	444	450
	6B	1024	130	508	514
MUL=2	6C	1152	194	572	578
	6D	1280	194	636	642
	6E	1536	258	764	770
	6F	1920	258	956	962
	70	1280	130	636	642
	71	1536	130	764	770
	72	1792	258	892	898
	73	2048	258	1020	1026
	74	2304	386	1148	1154
	75	2560	386	1276	1282
	76	3072	514	1532	1538

Table continues on the next page...

Table 21-13. I²C divider and hold values (continued)

	IBC (hex)	SCL Divider (clocks)	SDA Hold (clocks)	SCL Hold (start)	SCL Hold (stop)
	77	3840	514	1916	1922
	78	2560	258	1276	1282
	79	3072	258	1532	1538
	7A	3584	514	1788	1794
	7B	4096	514	2044	2050
	7C	4608	770	2300	2306
	7D	5120	770	2556	2562
	7E	6144	1026	3068	3074
	7F	7680	1026	3836	3842
MUL=4	80	80	28	24	44
	81	88	28	28	48
	82	96	32	32	52
	83	104	32	36	56
	84	112	36	40	60
	85	120	36	44	64
	86	136	40	52	72
	87	160	40	64	84
	88	112	28	40	60
	89	128	28	48	68
	8A	144	36	56	76
	8B	160	36	64	84
	8C	176	44	72	92
	8D	192	44	80	100
	8E	224	52	96	116
	8F	272	52	120	140
	90	192	36	72	100
	91	224	36	88	116
	92	256	52	104	132
MUL=4	93	288	52	120	148
	94	320	68	136	164
	95	352	68	152	180
	96	416	84	184	212
	97	512	84	232	260
	98	320	36	152	164
	99	384	36	184	196
	9A	448	68	216	228
	9B	512	68	248	260
	9C	576	100	280	292

Table continues on the next page...

Table 21-13. I²C divider and hold values (continued)

	IBC (hex)	SCL Divider (clocks)	SDA Hold (clocks)	SCL Hold (start)	SCL Hold (stop)
	9D	640	100	312	324
	9E	768	132	376	388
	9F	960	132	472	484
	A0	640	68	312	324
	A1	768	68	376	388
	A2	896	132	440	452
	A3	1024	132	504	516
	A4	1152	196	568	580
	A5	1280	196	632	644
	A6	1536	260	760	772
	A7	1920	260	952	964
	A8	1280	132	632	644
	A9	1536	132	760	772
	AA	1792	260	888	900
	AB	2048	260	1016	1028
	AC	2304	388	1144	1156
	AD	2560	388	1272	1284
	AE	3072	516	1528	1540
	AF	3840	516	1912	1924
	30	2560	260	1272	1284
	B1	3072	260	1528	1540
	B2	3584	516	1784	1796
	B3	4096	516	2040	2052
	B4	4608	772	2296	2308
	B5	5120	772	2552	2564
	B6	6144	1028	3064	3076
	B7	7680	1028	3832	3844
	B8	5120	516	2552	2564
MUL=4	B9	6144	516	3064	3076
	BA	7168	1028	3576	3588
	BB	8192	1028	4088	4100
	BC	9216	1540	4600	4612
	BD	10240	1540	5112	5124
	BE	12288	2052	6136	6148
	BF	15360	2052	7672	7684

21.6.5 Interrupts

This section presents the following topics:

- Interrupt vector
- Interrupt description

21.6.5.1 Interrupt vector

The I^2C module uses only one interrupt vector.

Table 21-14. Interrupt summary

Interrupt	Offset	Vector	Priority	Source	Description
I^2C Interrupt	—	—	—	IBAL, TCF, IAAS, IBB bits in IBSR register	When any of IBAL, TCF or IAAS bits is set, an interrupt may be caused based on Arbitration lost, Transfer Complete or Address Detect conditions. If enabled by BIIE, the de-assertion of IBB can also cause an interrupt, indicating that the bus is idle.

21.6.5.2 Interrupt description

There are five types of internal interrupts in the I^2C. The interrupt service routine can determine the interrupt type by reading the Status register.

I^2C Interrupt can be generated on the following events:

- Arbitration Lost condition (IBAL bit set)
- Byte Transfer condition (TCF bit set and DMAEN bit not set)
- Address Detect condition (IAAS bit set)
- No Acknowledge from slave received when expected
- Bus Going Idle (IBB bit not set)

The I^2C interrupt is enabled by the IBCR[IBIE] bit. It must be cleared by writing '1' to the IBIF bit in the interrupt service routine. The Bus Going Idle interrupt needs to be additionally enabled by the IBIC[BIIE] bit.

21.6.6 STOP mode

This mode allows the software to put the I^2C module in power-down state. Once the STOP request is asserted, the I^2C module comes to a graceful halt after completing all the ongoing transactions.

As soon as the I^2C module enters STOP mode:

- The I^2C clock is disabled.
- No transaction can take place.
- All registers are inaccessible.

The I^2C module enters this mode only after successfully completing the current ongoing transaction, hence the low-power request is acknowledged once the STOP condition occurs. The user must ensure that the bus is free when STOP is requested. To request STOP for ongoing transmission the user must wait until the transmission is complete, followed by clearing of the IBCR[TXRX] field. See the figure below for more details.

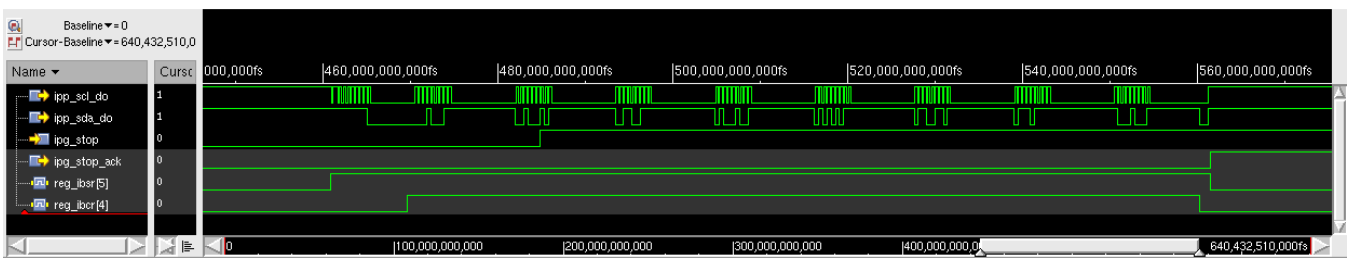


Figure 21-8. I^2C stop mode behavior when master is receiving and slave is transmitting

21.6.7 DMA interface

A simple DMA interface is implemented so that the I^2C can request data transfers with minimal support from the CPU (see DMA application information). DMA mode is enabled by setting bit 1 in the Control Register (IBCR).

The DMA interface is operational when the I^2C module is configured for Master mode.

At least three bytes of data per frame must be transferred from/to the slave when using DMA mode, although in practice it will only be worthwhile using the DMA mode when there is a large number of data bytes to transfer per frame.

Two internal signals, TX request and RX request, are used to signal the DMA controller when the I^2C module requires data to be written or read from the data register.

Further details of the DMA interface can be found in the Initialization/application information, of this document.

21.7 Initialization/application information

This section presents the following topics:

- Recommended interrupt service flow
- General programming guidelines (for both master and slave mode)
- Programming guidelines specific to master mode
- Programming guidelines specific to slave mode
- DMA application information

21.7.1 Recommended interrupt service flow

The following figure shows a flowchart for the recommended I^2C interrupt service routine. Deviation from the flowchart may result in unpredictable I^2C bus behavior.

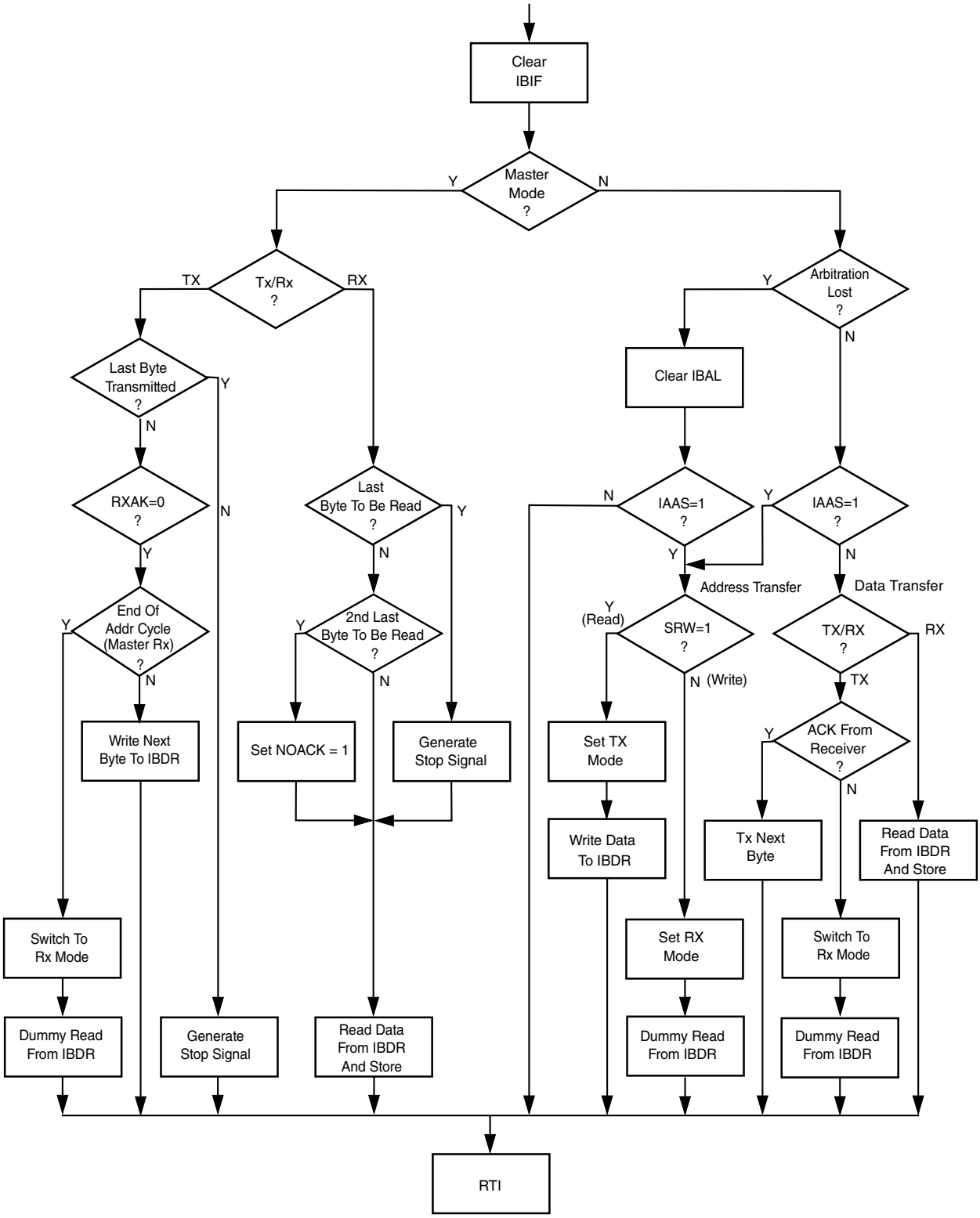


Figure 21-9. Recommended I²C interrupt service routine flowchart

21.7.2 General programming guidelines (for both master and slave mode)

This section provides programming guidelines recommended for the I2C module in both master and slave mode. It presents the following topics:

- Initializing the I2C module
- Software response after a transfer

21.7.2.1 Initializing the I2C module

The following table describes how to initialize the I2C module.

Table 21-15. I2C initialization procedure

Step	Action
1	Use I2C Bus Frequency Divider Register (I2C_IBFD) to select the required division ratio to obtain SCL frequency from the system clock.
2	Use I2C Bus Address Register (I2C_IBAD) to define the slave address.
3	Clear the MDIS field in I2C Bus Control Register (I2C_IBCR) to enable the I2C interface system.
4	Use I2C Bus Control Register (I2C_IBCR) to select Master/Slave mode, Transmit/Receive mode, and whether interrupts are enabled or disabled.
5	(Optional) Use I2C Bus Interrupt Config Register (I2C_IBIC) to further refine the interrupt behavior.

21.7.2.2 Software response after a transfer

Transmission or reception of a byte will set the data transferring bit (TCF) to 1, which indicates one byte communication is finished. The I2C Bus interrupt bit (IBIF) is set also; an interrupt will be generated if the interrupt function is enabled during initialization by setting the IBIE bit. The IBIF (interrupt flag) can be cleared by writing one (in the interrupt service routine, if interrupts are used).

The TCF bit will be cleared to indicate data transfer in progress whenever data register is written to in transmit mode, or during reading out from data register in receive mode. The TCF bit should not be used as a data transfer complete flag as the flag timing is dependent on a number of factors including the I2C bus frequency. This bit may not conclusively provide an indication of a transfer complete situation. It is recommended that transfer complete situations are detected using the IBIF flag.

Software may service the I^2C I/O in the main program by monitoring the IBIF bit if the interrupt function is disabled. Polling should monitor the IBIF bit rather than the TCF bit since their operation is different when arbitration is lost.

When a "Transfer Complete" interrupt occurs at the end of the address cycle, the master will always be in transmit mode, i.e. the address is transmitted. If master receive mode is required, indicated by R/W bit sent with slave calling address, then the Tx/Rx bit at Master side should be toggled at this stage. If Master does not receive an ACK from Slave, then transmission must be re-initiated or terminated.

In slave mode, IAAS bit will get set in IBSR if Slave address (IBAD) matches the Master calling address. This is an indication that Master-Slave data communication can now start. During address cycles (IBSR[IAAS]=1), the SRW bit in the status register is read to determine the direction of the subsequent transfer and the Tx/Rx bit is programmed accordingly. For slave mode data cycles (IBSR[IAAS]=0), the SRW bit is not valid. The Tx/Rx bit in the control register should be read to determine the direction of the current transfer.

21.7.3 Programming guidelines specific to master mode

This section presents the following topics:

- Generating START
- Transmit/receive sequence
- Generating STOP
- Generating repeated START
- Loss of arbitration

21.7.3.1 Generating START

After completion of the initialization procedure, serial data can be transmitted by selecting the 'master transmitter' mode. If the device is connected to a multi-master bus system, the state of the I^2C Bus Busy bit (IBB) must be tested to check whether the serial bus is free.

If the bus is free (IBB=0), the start condition and the first byte (the slave address) can be sent. The data written to the data register comprises the slave calling address and the LSB, which is set to indicate the direction of transfer required from the slave.

The bus free time (i.e., the time between a STOP condition and the following START condition) is built into the hardware that generates the START cycle. Depending on the relative frequencies of the system clock and the SCL period, it may be necessary to wait until the I^2C is busy after writing the calling address to the IBDR before proceeding with the following instructions. This is illustrated in the following example.

An example of the sequence of events which generates the START signal and transmits the first byte of data (slave address) is shown below:

```
while (IBSR[IBB]==1)                // wait in loop for IBB flag to clear
IBCR[MS/MSL] and IBCR[]Tx/Rx] = 1    // master and transmit mode, that is,
                                    // generate start condition
IBDR = calling_address              // send the calling address to the data register
while (bit 5, IBSR ==0)             // wait in loop for IBB flag to be set
```

21.7.3.2 Transmit/receive sequence

The following tables present the sequences for:

- Master transmit
- Master receive
- Slave transmit
- Slave receive

Table 21-16. Master transmit sequence

Step	Action
a	Use I2C Bus Frequency Divider Register (I2C_IBFD) to select the required division ratio to obtain SCL frequency from Platform clock/2.
b	Write 0 to the IBDIS field in I2C Bus Control Register (I2C_IBCR) to enable the I^2C interface system.
c	Use I2C Bus Control Register (I2C_IBCR) to select Master mode, Transmit mode, and interrupt enable.
d	Write 0 to the IBIF field in I2C Bus Status Register (I2C_IBSR).
e	Write data to I2C Bus Data I/O Register (I2C_IBDR).
f	Observe changes in the TCF field of I2C Bus Status Register (I2C_IBSR) : • When IBSR[TCF] becomes 0, the transfer is in progress. • When IBSR[TCF] becomes 1, the transfer is complete.
g	Wait until the IBIF field in I2C Bus Status Register (I2C_IBSR) becomes 1.
h	Read the fields in I2C Bus Status Register (I2C_IBSR) to determine what happened: • If TCF = 1, the transfer completed. • If RXAK = 1, a No Acknowledge condition occurred. • If IBB = 0, the bus transitioned from Busy to Idle state. • If IBB = 1, you can ignore check of Arbitration Loss (IBAL = 1). **NOTE:** You can ignore Address Detect (IAAS = 1) for master mode (it is valid only for slave mode).
i	Examine the RXAK field in I2C Bus Status Register (I2C_IBSR) for an acknowledgment from the slave.
j	Repeat steps d through i to transfer the next consecutive bytes of data.

Table 21-17. Master receive sequence

Step	Action
a	Follow steps a through i in Table 21-16 for address dispatch.
b	Write 0 to the IBIF field in I2C Bus Status Register (I2C_IBSR).
c	Write 0 to the TXRX field in I2C Bus Control Register (I2C_IBCR) to select Receive mode.
d	Perform a dummy read of I2C Bus Data I/O Register (I2C_IBDR) to initiate the receive operation.
e	Wait until the TCF field in I2C Bus Status Register (I2C_IBSR) becomes 1. (This proves that the transfer is complete.)
f	Wait until the IBIF field in I2C Bus Status Register (I2C_IBSR) becomes 1.
g	Read the fields in I2C Bus Status Register (I2C_IBSR) to determine what happened: • If TCF = 1, the reception completed. • If IBB = 0, the bus transitioned from Busy to Idle state. • If IBB = 1, you can ignore check of Arbitration Loss (IBAL = 1). **NOTE:** You can ignore the No Acknowledge condition (RXAK = 1) for receive mode.
h	Read I2C Bus Data I/O Register (I2C_IBDR) to determine the data received from the slave.

Table 21-18. Slave transmit sequence

Step	Action
a	Use I2C Bus Address Register (I2C_IBAD) to define the slave address.
b	Write 0 to the IBDIS field in I2C Bus Control Register (I2C_IBCR) to enable the I^2C interface system.
c	Examine fields in I2C Bus Status Register (I2C_IBSR) as follows: • If IAAS = 1, examine IBSR[SRW]. • If IAAS = 1 and SRW = 1, write 1 to IBCR[TXRX] to select Transmit mode.
d	Write data to I2C Bus Data I/O Register (I2C_IBDR).
e	Wait until the IBIF field in I2C Bus Status Register (I2C_IBSR) becomes 1.
f	Wait until the RXAK field in I2C Bus Status Register (I2C_IBSR) becomes 0.
g	Write 0 to the IBIF field in I2C Bus Status Register (I2C_IBSR).
h	Repeat steps d through g for the next consecutive data transfers.

Table 21-19. Slave receive sequence

Step	Action
a	Use I2C Bus Address Register (I2C_IBAD) to define the slave address.
b	Write 0 to the IBDIS field of I2C Bus Control Register (I2C_IBCR) to enable the I^2C interface system.
c	Examine fields in I2C Bus Status Register (I2C_IBSR) as follows: • If IAAS = 1, examine IBSR[SRW]. • If IAAS = 1 and SRW = 0, write 0 to IBCR[TXRX] to select Receive mode.
d	Write 0 to the IBIF field of I2C Bus Status Register (I2C_IBSR).
e	Perform a dummy read of I2C Bus Data I/O Register (I2C_IBDR) to initiate the receive operation.
f	Wait until the TCF field of I2C Bus Status Register (I2C_IBSR) becomes 1. (This proves that the transfer is complete.)
g	Wait until the IBIF field of I2C Bus Status Register (I2C_IBSR) becomes 1.

Table continues on the next page...

Table 21-19. Slave receive sequence (continued)

Step	Action
h	Read the fields in I2C Bus Status Register (I2C_IBSR) to determine what happened: • If TCF = 1, the reception completed. • If IBB = 0, the bus transitioned from Busy to Idle state. • If IBB = 1, you can ignore check of Arbitration Loss (IBAL = 1). **NOTE:** You can ignore the No Acknowledge condition (RXAK = 1) for receive mode.
i	Read I2C Bus Data I/O Register (I2C_IBDR) to determine the data received from the master.

21.7.3.3 Generating STOP

A data transfer ends with a STOP signal generated by the 'master' device. A master transmitter can simply generate a STOP signal after all the data has been transmitted. The following is an example showing how a STOP condition is generated by a master transmitter.

```
if (tx_count == 0) or      // check to see if all data bytes have been transmitted
  (bit 0, IBSR == 1) {     // or if no ACK generated
    clear bit 5, IBCR      // generate stop condition
  }
else {
 IBDR = data_to_transmit   // write byte of data to DATA register
   tx_count --             // decrement counter
 }                         // return from interrupt
```

If a master receiver wants to terminate a data transfer, it must inform the slave transmitter by not acknowledging the last byte of data which can be done by setting the NOACK bit in IBCR before reading the 2nd last byte of data. Before reading the last byte of data, a STOP signal must first be generated. The following is an example showing how a STOP signal is generated by a master receiver.

```
rx_count --              // decrease the rx counter
if (rx_count ==1)        // 2nd last byte to be read ?
 bit 3, IBCR = 1         // disable ACK
 if (rx_count == 0)      // last byte to be read ?
 bit 5, IBCR = 0         // generate stop signal
else
data_received = IBDR     // read RX data and store
```

21.7.3.4 Generating repeated START

At the end of data transfer, if the master still wants to communicate on the bus, it can generate another START signal followed by another slave address without first generating a STOP signal. A program example is as shown.

```
bit 2, IBCR = 1             // generate another start (restart)
IBDR == calling_address     // transmit the calling address
```

21.7.3.5 Loss of arbitration

If several masters try to engage the bus simultaneously, only one master wins and the others lose arbitration. The devices that lost arbitration are immediately switched to slave mode by the hardware. Their data output to the SDA line is stopped, but SCL is still generated until the end of the byte during which arbitration was lost. An interrupt occurs at the falling edge of the ninth clock of this transfer with IBAL=1 and MS/SL=0. If one master attempts to start transmission, while the bus is being engaged by another master, the hardware will inhibit the transmission, switch the MS/SL bit from 1 to 0 without generating a STOP condition, generate an interrupt to CPU, set the IBAL to indicate that the attempt to engage the bus is failed, and not set the TCF due to the loss of data during arbitration. When considering these cases, the slave service routine should test the IBAL first and the software should clear the IBAL bit if it is set.

21.7.4 Programming guidelines specific to slave mode

In the slave interrupt service routine, the module addressed as slave bit (IAAS) should be tested to check if a calling of its own address has just been received. If IAAS is set, software should set the transmit/receive mode select bit (Tx/Rx bit of IBCR) according to the R/W command bit (SRW). Writing to the IBCR clears IAAS automatically. Note that the only time IAAS is read as set is from the interrupt at the end of the address cycle where an address match occurred. Interrupts resulting from subsequent data transfers will have IAAS cleared. A data transfer may now be initiated by writing information to IBDR for slave transmits or dummy reading from IBDR in slave receive mode. The slave will drive SCL low in-between byte transfers, SCL is released when the IBDR is accessed in the required mode.

In slave transmitter routine, the received acknowledge bit (RXAK) must be tested before transmitting the next byte of data. Setting RXAK means an "end of data" signal from the master receiver, after which it must be switched from transmitter mode to receiver mode by software. A dummy read then releases the SCL line so that the master can generate a STOP signal.

21.7.5 DMA application information

The DMA interface on the I^2C is not completely autonomous and requires intervention from the CPU to start and to terminate the frame transfer. DMA mode is only valid for Master transmit and Master receive modes. Software must ensure that the DMAEN field in I2C Bus Control Register (I2C_IBCR) is not set when the I^2C module is configured in slave mode.

The DMA controller must only transfer one byte of data per Tx/Rx request. This is because there is no FIFO on the I^2C block.

The CPU should also keep the I^2C interrupt enabled during a DMA transfer to detect the arbitration lost condition and take action to recover from this situation. The DMAEN field in I2C Bus Control Register (I2C_IBCR) works as a disable for the transfer complete interrupt. This means that during normal transfers (no errors) there will always be either an interrupt or a request to the DMA controller, depending on the setting of the DMAEN field. All error conditions will trigger an interrupt and require CPU intervention. The address match condition will not occur in DMA mode as the I^2C should never be configured for slave operation.

The following sections detail how to set up a DMA transfer and what intervention is required from the CPU. It is assumed that the system DMA controller is capable of generating an interrupt after a certain number of DMA transfers have taken place.

The sections present the following topics:

- DMA mode, master transmit
- DMA mode, master reception
- Exiting DMA mode, system requirement considerations

21.7.5.1 DMA mode, master transmit

The following flow diagram details exactly the operation for using a DMA controller to transmit "n" data bytes to a slave. The first byte (the slave calling address) is always transmitted by the CPU. All subsequent data bytes (apart from the last data byte) can be transferred by the DMA controller. The last data byte must be transferred by the CPU.

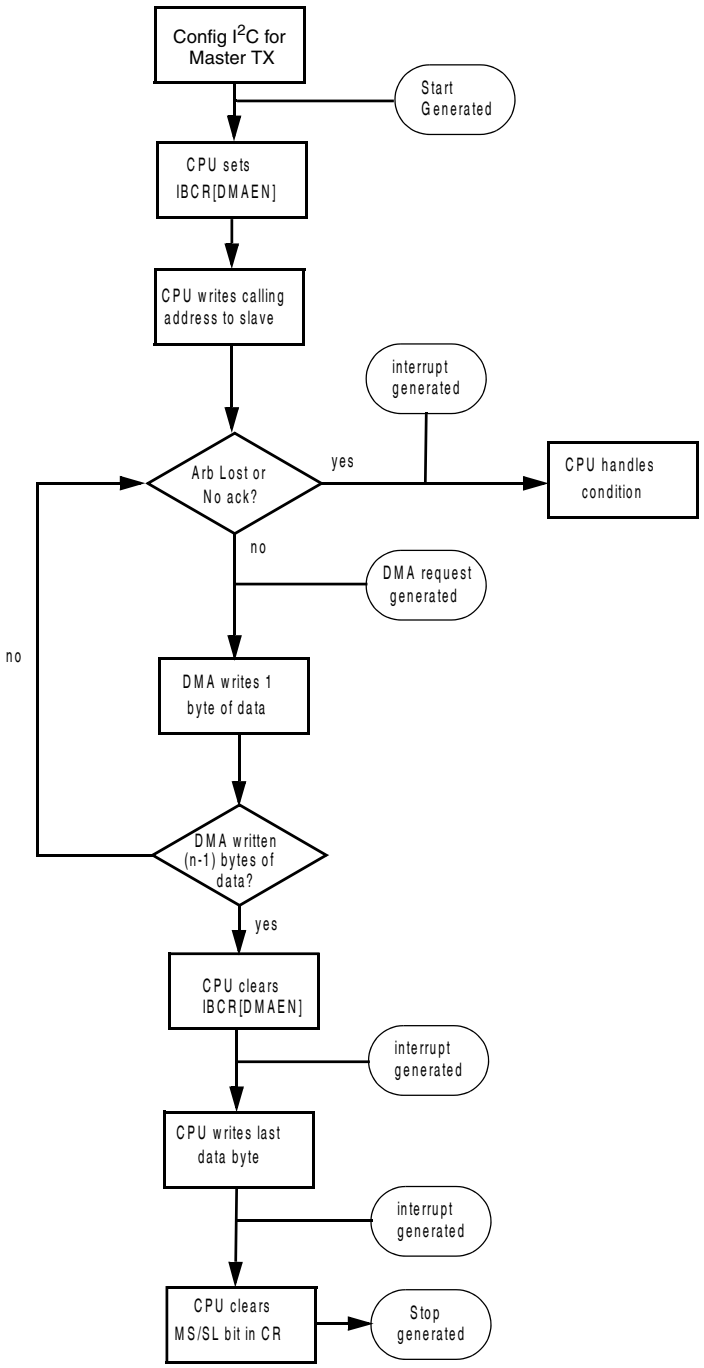


Figure 21-10. Flow-Chart of DMA mode master transmit

21.7.5.2 DMA mode, master reception

The following flow diagram details the exact operation for using a DMA controller to receive "n" data bytes from a slave. The first byte (the slave calling address) is always transmitted by the CPU. All subsequent data bytes (apart from the two last data bytes) can be read by the DMA controller. The last two data bytes must be transferred by the CPU.

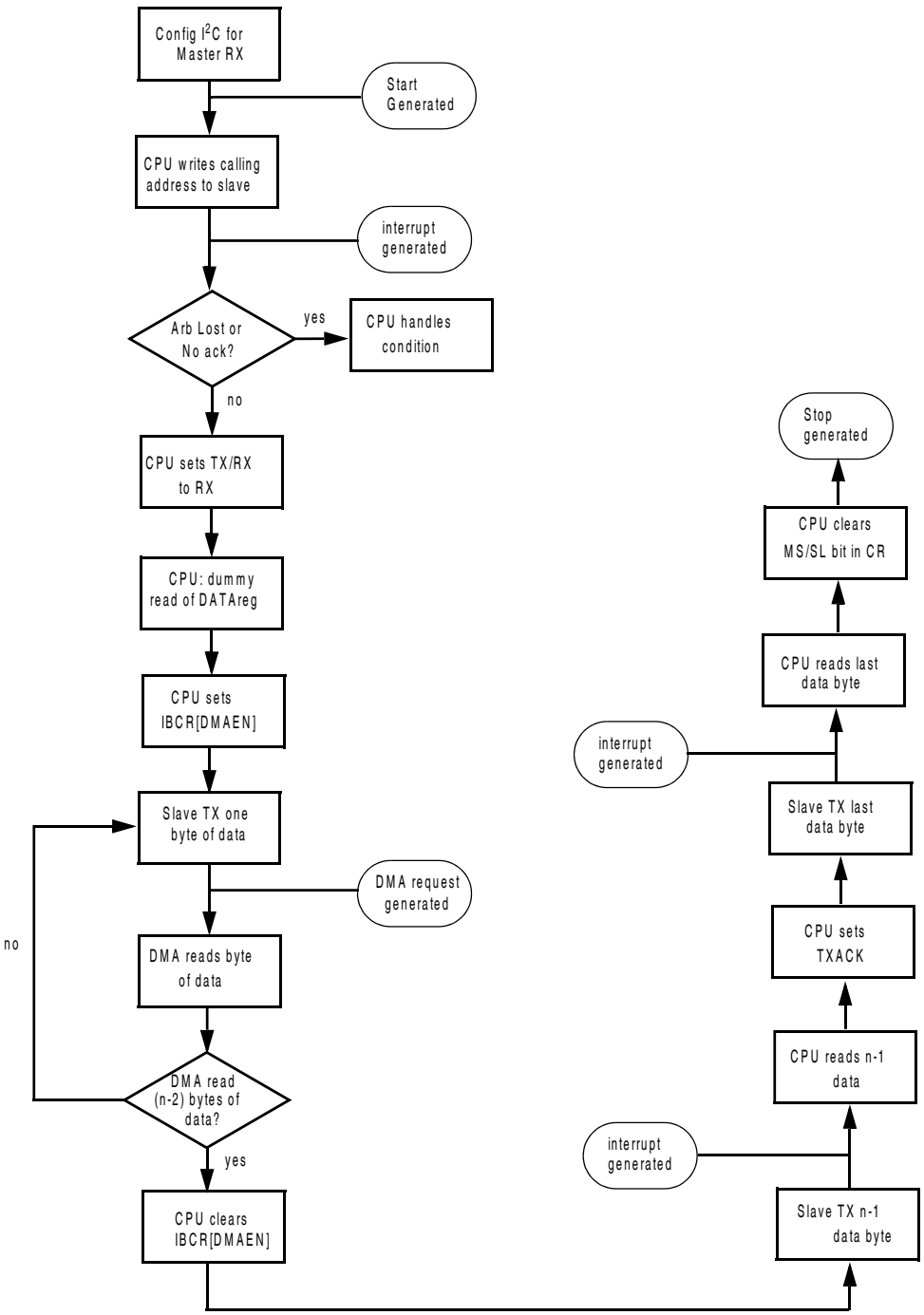


Figure 21-11. Flow-Chart of DMA mode master receive

21.7.5.3 Exiting DMA mode, system requirement considerations

As described above, the final transfers of both TX and RX transfers need to be handled via interrupt by the CPU. To change from DMA to interrupt driven transfers in the I^2C module, you have to disable the DMAEN bit in the IBCR register. The trigger to exit the DMA mode is that the programmed DMA Transfer Control Descriptor (TCD) has completed all its transfers to/from the I^2C module.

After the last DMA write (TX mode) to the I^2C the module will immediately start the next I^2C-bus transfer. The same is true for receive mode. After the DMA read from the IBDR register the module initiates the next I^2C-bus transfer. This results in two possible scenarios in the DMA mode exiting scheme.

1. Fast reaction

   The DMAEN bit is cleared before the next I^2C-bus transfer completes. In this case the module will raise an interrupt request to the CPU which can be serviced normally.

2. Slow reaction

   The DMAEN bit is cleared after the next I^2C-bus transfer has already completed. In this case, the module will not raise an interrupt request to the CPU. Instead the TCF bit can be read to determine that the transfer completed and the module is ready for further transfer.

What is fast/slow reaction?

The reaction time T_R for the system to disable DMAEN after the last DMA controller access to the I^2C is the time required for one byte transfer over the I^2C. For 'fast reaction' the disabling has to occur before the 9^{th} bit of the data transfer which is the ACK bit. So the time available is eight times the SCL period.

$$T_R = 8 \times T_{SCL}$$

In fast mode, with 400kbit/s, T_{SCL} is 2.5µs, so T_R is 20µs.

Depending on the system and DMA controller there are different possibilities for the de-assertion of DMAEN. Three options are:

1. CPU intervention via Interrupt

   The DMA controller is programmed to signal an interrupt to the CPU which is then responsible for the de-assertion of DMAEN. This scheme should be supported by most systems but it can result in a slow reaction time if other higher priority

interrupts interfere. Therefore the interrupt handling routine can become complicated as it has to check which of the two cases happened (check TCF bit) and act accordingly. In case of slow reaction you can force an interrupt for the I^2C in the interrupt controller to have the further transfer handled by the normal I^2C interrupt routine.

Note

The use of nested interrupts can still cause potential issues in this scenario, if someone tries to stall the DMA interrupt between the de-assertion and DMAEN bit and checks the TCF bit.

2. DMA channel linking

   If the Transfer control descriptor in the DMA controller that performs the data transfer is linked to another channel that does a write to IBCR to disable the DMAEN field, this might probably be the fastest system solution, but it uses two DMA channels.

   Note

   Here you have to make sure on system level that no higher priority DMA requests occur between the two linked TCDs as those could again create a scenario of slow reaction.

3. DMA scatter/gather process

   If the Transfer control descriptor in the DMA controller that performs the data transfer has the scatter/gather feature activated, this feature will initiate a reload of another TCD from system RAM after the completion of the first TCD. The new TCD will have its start bit already set and immediately start the required write to the IBCR to disable the DMAEN field. This TCD also has scatter/gather activated and is programmed to reload the initial TCD upon completion, bringing the system back into a "ready-for-I^2C-transfer" state. The advantage over the two other solutions is that this requires neither CPU intervention nor a second DMA channel. This comes at the cost of 64 bytes RAM (two TCDs), some system bus transfer overhead and a little increase in application code complexity.

Note

Here you have to make sure at system level that no higher priority DMA requests occur during the scatter/gather process, as those could again create a scenario of slow reaction.

Example latencies for a 32 MHz system with a full speed 32-bit AHB bus and an I^2C connected via half speed IPI bus:

- Accessing the I^2C from the DMA controller via IPI bus typically requires four cycles (consecutive accesses to the I^2C could be faster):

  $4 \times T_{IPI}$ = 4/16 MHz = 250 ns

- Reloading a new TCD (8 x 32-bit) via AHB to the DMA controller (scatter/gather process):

  $8 \times T_{AHD}$ = 8/32 MHz = 250 ns

Example latencies for a 150 MHz system with a full speed 32-bit AHB bus and an I^2C connected via half speed IPI bus:

- Accessing the I^2C from the DMA controller via IPI bus typically requires four cycles (consecutive accesses to the I^2C could be faster):

  $4 \times T_{IPI}$ = 4/150 MHz = 26.6 ns

- Reloading a new TCD (4 x 64-bit) via AHB to the DMA controller (scatter/gather process):

  $4 \times T_{AHD}$ = 4/150 MHz = 26.6 ns

With the DMA scatter/gather process the required IBCR access can be done in 0.5 µs, leaving a large margin of 19.5 µs for additional system delays. In this way, the slow reaction case can be prevented. The system user needs to decide which usage model best suits his overall requirement.

Chapter 22
Integrated Interchip Sound (I2S) / Synchronous Audio Interface (SAI)

22.1 The I2S/SAI module as implemented on the chip

This section provides details about how the I2S/SAI module is implemented on the chip.

22.1.1 LS1012A I2S/SAI module integration

The following table describes the I2S/SAI module integration into this chip:

Table 22-1. I2S/SAI module integration

Module	Module Base address	Notes
SAI1	2B5_0000	The SAI audio modules are clocked with the platform clock/2. For the clock requirements other than this, the external SAI clocks (SAIn_TX_BCLK and SAIn_RX_BCLK) can be configured as inputs to the system.
SAI2	2B6_0000	
SAI3	2B7_0000	
SAI4	2B8_0000	
SAI5	2B9_0000	

Additionally,

- All SAI modules support only the slave mode.
- All five modules have been integrated with identical parameters.
- Synchronous operation among multiple SAI modules is not supported.
- All SAI modules support one data input and one data output port.
- SAI1 and SAI2 support full-duplex operation. However, SAI3, SAI4, and SAI5 support only simplex operation.

The remainder of this chapter refers to a single I2S/SAI module. Notes are included to indicate variations for multiple instantiations.

22.1.2 I2Sx_TCR2[MSEL] and I2Sx_RCR2[MSEL] mapping

The following table shows the MCLK selection for the I2S module in the chip.

Table 22-2. I2Sx_TCR2/I2Sx_RCR2[MSEL] field description

Field	Description
4-5 MSEL	MCLK Select SAI master clock is externally generated, so these bits will take no effect when configured for an externally generated bit clock. 00-11 Reserved

22.1.3 LS1012A SAI/I2S module special consideration

The I2S/SAI module implements the following parameter settings in the chip:

Table 22-3. LS1012A I2S/SAI parameter settings

I2S/SAI parameters	LS1012A parameter value
Stop mode support	Yes. Refers to LPM20 low power mode of the chip.
Debug mode support	Yes. The chip entered in this mode using the DBGACK of the core

22.2 Introduction

The I^2S (or I2S) module provides a synchronous audio interface (SAI) that supports full-duplex serial interfaces with frame synchronization such as I^2S, AC97, TDM, and codec/DSP interfaces.

22.2.1 Features

Note that some of the features are not supported across all SAI instances; see the chip-specific information in the first section of this chapter.

- Transmitter with independent bit clock and frame sync supporting 1 data line
- Receiver with independent bit clock and frame sync supporting 1 data line
- Maximum Frame Size of 32 words

- Word size of between 8-bits and 32-bits
- Word size configured separately for first word and remaining words in frame
- Asynchronous 32 × 32-bit FIFO for each transmit and receive channel
- Supports graceful restart after FIFO error

22.2.2 Block diagram

The following block diagram also shows the module clocks.

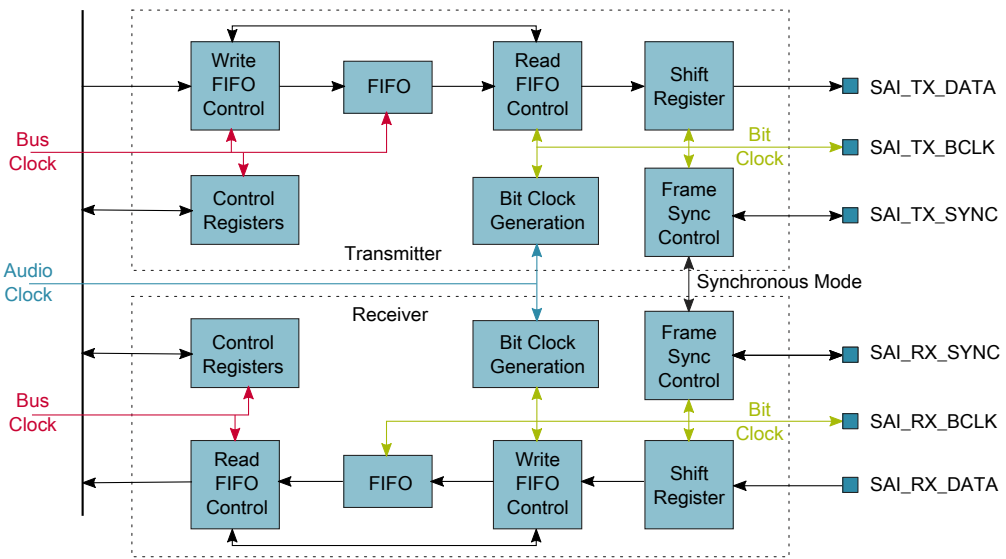


Figure 22-1. I²S/SAI block diagram

22.2.3 Modes of operation

Available power modes include: Run mode.

22.2.3.1 Run mode

In Run mode, the SAI transmitter and receiver operate normally.

22.3 External signals

Name	Function	I/O
SAI_TX_BCLK	Transmit Bit Clock. The bit clock is an input when externally generated and an output when internally generated.	I/O
SAI_TX_SYNC	Transmit Frame Sync. The frame sync is an input sampled synchronously by the bit clock when externally generated and an output generated synchronously by the bit clock when internally generated.	I/O
SAI_TX_DATA	Transmit Data. The transmit data is generated synchronously by the bit clock and is tristated whenever not transmitting a word.	O
SAI_RX_BCLK	Receive Bit Clock. The bit clock is an input when externally generated and an output when internally generated.	I/O
SAI_RX_SYNC	Receive Frame Sync. The frame sync is an input sampled synchronously by the bit clock when externally generated and an output generated synchronously by the bit clock when internally generated.	I/O
SAI_RX_DATA	Receive Data. The receive data is sampled synchronously by the bit clock.	I
SAI_MCLK	Audio Master Clock.	I

22.4 Memory map and register definition

A read or write access to an address from offset and above will result in a bus error.

I2S memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
2B5_0000	SAI Transmit Control Register (I2S1_TCSR)	32	R/W	0000_0000h	22.4.1/1048
2B5_0004	SAI Transmit Configuration 1 Register (I2S1_TCR1)	32	R/W	0000_0000h	22.4.2/1051
2B5_0008	SAI Transmit Configuration 2 Register (I2S1_TCR2)	32	R/W	0000_0000h	22.4.3/1052
2B5_000C	SAI Transmit Configuration 3 Register (I2S1_TCR3)	32	R/W	0000_0000h	22.4.4/1053
2B5_0010	SAI Transmit Configuration 4 Register (I2S1_TCR4)	32	R/W	0000_0000h	22.4.5/1054
2B5_0014	SAI Transmit Configuration 5 Register (I2S1_TCR5)	32	R/W	0000_0000h	22.4.6/1056

Table continues on the next page...

I2S memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
2B5_0020	SAI Transmit Data Register (I2S1_TDR0)	32	W (always reads 0)	0000_0000h	22.4.7/1056
2B5_0040	SAI Transmit FIFO Register (I2S1_TFR0)	32	R	0000_0000h	22.4.8/1057
2B5_0060	SAI Transmit Mask Register (I2S1_TMR)	32	R/W	0000_0000h	22.4.9/1057
2B5_0080	SAI Receive Control Register (I2S1_RCSR)	32	R/W	0000_0000h	22.4.10/ 1059
2B5_0084	SAI Receive Configuration 1 Register (I2S1_RCR1)	32	R/W	0000_0000h	22.4.11/ 1062
2B5_0088	SAI Receive Configuration 2 Register (I2S1_RCR2)	32	R/W	0000_0000h	22.4.12/ 1062
2B5_008C	SAI Receive Configuration 3 Register (I2S1_RCR3)	32	R/W	0000_0000h	22.4.13/ 1064
2B5_0090	SAI Receive Configuration 4 Register (I2S1_RCR4)	32	R/W	0000_0000h	22.4.14/ 1065
2B5_0094	SAI Receive Configuration 5 Register (I2S1_RCR5)	32	R/W	0000_0000h	22.4.15/ 1066
2B5_00A0	SAI Receive Data Register (I2S1_RDR0)	32	R	0000_0000h	22.4.16/ 1067
2B5_00C0	SAI Receive FIFO Register (I2S1_RFR0)	32	R	0000_0000h	22.4.17/ 1068
2B5_00E0	SAI Receive Mask Register (I2S1_RMR)	32	R/W	0000_0000h	22.4.18/ 1068
2B6_0000	SAI Transmit Control Register (I2S2_TCSR)	32	R/W	0000_0000h	22.4.1/1048
2B6_0004	SAI Transmit Configuration 1 Register (I2S2_TCR1)	32	R/W	0000_0000h	22.4.2/1051
2B6_0008	SAI Transmit Configuration 2 Register (I2S2_TCR2)	32	R/W	0000_0000h	22.4.3/1052
2B6_000C	SAI Transmit Configuration 3 Register (I2S2_TCR3)	32	R/W	0000_0000h	22.4.4/1053
2B6_0010	SAI Transmit Configuration 4 Register (I2S2_TCR4)	32	R/W	0000_0000h	22.4.5/1054
2B6_0014	SAI Transmit Configuration 5 Register (I2S2_TCR5)	32	R/W	0000_0000h	22.4.6/1056
2B6_0020	SAI Transmit Data Register (I2S2_TDR0)	32	W (always reads 0)	0000_0000h	22.4.7/1056
2B6_0040	SAI Transmit FIFO Register (I2S2_TFR0)	32	R	0000_0000h	22.4.8/1057
2B6_0060	SAI Transmit Mask Register (I2S2_TMR)	32	R/W	0000_0000h	22.4.9/1057
2B6_0080	SAI Receive Control Register (I2S2_RCSR)	32	R/W	0000_0000h	22.4.10/ 1059
2B6_0084	SAI Receive Configuration 1 Register (I2S2_RCR1)	32	R/W	0000_0000h	22.4.11/ 1062
2B6_0088	SAI Receive Configuration 2 Register (I2S2_RCR2)	32	R/W	0000_0000h	22.4.12/ 1062
2B6_008C	SAI Receive Configuration 3 Register (I2S2_RCR3)	32	R/W	0000_0000h	22.4.13/ 1064

Table continues on the next page...

I2S memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
2B6_0090	SAI Receive Configuration 4 Register (I2S2_RCR4)	32	R/W	0000_0000h	22.4.14/ 1065
2B6_0094	SAI Receive Configuration 5 Register (I2S2_RCR5)	32	R/W	0000_0000h	22.4.15/ 1066
2B6_00A0	SAI Receive Data Register (I2S2_RDR0)	32	R	0000_0000h	22.4.16/ 1067
2B6_00C0	SAI Receive FIFO Register (I2S2_RFR0)	32	R	0000_0000h	22.4.17/ 1068
2B6_00E0	SAI Receive Mask Register (I2S2_RMR)	32	R/W	0000_0000h	22.4.18/ 1068
2B7_0000	SAI Transmit Control Register (I2S3_TCSR)	32	R/W	0000_0000h	22.4.1/1048
2B7_0004	SAI Transmit Configuration 1 Register (I2S3_TCR1)	32	R/W	0000_0000h	22.4.2/1051
2B7_0008	SAI Transmit Configuration 2 Register (I2S3_TCR2)	32	R/W	0000_0000h	22.4.3/1052
2B7_000C	SAI Transmit Configuration 3 Register (I2S3_TCR3)	32	R/W	0000_0000h	22.4.4/1053
2B7_0010	SAI Transmit Configuration 4 Register (I2S3_TCR4)	32	R/W	0000_0000h	22.4.5/1054
2B7_0014	SAI Transmit Configuration 5 Register (I2S3_TCR5)	32	R/W	0000_0000h	22.4.6/1056
2B7_0020	SAI Transmit Data Register (I2S3_TDR0)	32	W (always reads 0)	0000_0000h	22.4.7/1056
2B7_0040	SAI Transmit FIFO Register (I2S3_TFR0)	32	R	0000_0000h	22.4.8/1057
2B7_0060	SAI Transmit Mask Register (I2S3_TMR)	32	R/W	0000_0000h	22.4.9/1057
2B7_0080	SAI Receive Control Register (I2S3_RCSR)	32	R/W	0000_0000h	22.4.10/ 1059
2B7_0084	SAI Receive Configuration 1 Register (I2S3_RCR1)	32	R/W	0000_0000h	22.4.11/ 1062
2B7_0088	SAI Receive Configuration 2 Register (I2S3_RCR2)	32	R/W	0000_0000h	22.4.12/ 1062
2B7_008C	SAI Receive Configuration 3 Register (I2S3_RCR3)	32	R/W	0000_0000h	22.4.13/ 1064
2B7_0090	SAI Receive Configuration 4 Register (I2S3_RCR4)	32	R/W	0000_0000h	22.4.14/ 1065
2B7_0094	SAI Receive Configuration 5 Register (I2S3_RCR5)	32	R/W	0000_0000h	22.4.15/ 1066
2B7_00A0	SAI Receive Data Register (I2S3_RDR0)	32	R	0000_0000h	22.4.16/ 1067
2B7_00C0	SAI Receive FIFO Register (I2S3_RFR0)	32	R	0000_0000h	22.4.17/ 1068
2B7_00E0	SAI Receive Mask Register (I2S3_RMR)	32	R/W	0000_0000h	22.4.18/ 1068
2B8_0000	SAI Transmit Control Register (I2S4_TCSR)	32	R/W	0000_0000h	22.4.1/1048
2B8_0004	SAI Transmit Configuration 1 Register (I2S4_TCR1)	32	R/W	0000_0000h	22.4.2/1051
2B8_0008	SAI Transmit Configuration 2 Register (I2S4_TCR2)	32	R/W	0000_0000h	22.4.3/1052
2B8_000C	SAI Transmit Configuration 3 Register (I2S4_TCR3)	32	R/W	0000_0000h	22.4.4/1053

Table continues on the next page...

I2S memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
2B8_0010	SAI Transmit Configuration 4 Register (I2S4_TCR4)	32	R/W	0000_0000h	22.4.5/1054
2B8_0014	SAI Transmit Configuration 5 Register (I2S4_TCR5)	32	R/W	0000_0000h	22.4.6/1056
2B8_0020	SAI Transmit Data Register (I2S4_TDR0)	32	W (always reads 0)	0000_0000h	22.4.7/1056
2B8_0040	SAI Transmit FIFO Register (I2S4_TFR0)	32	R	0000_0000h	22.4.8/1057
2B8_0060	SAI Transmit Mask Register (I2S4_TMR)	32	R/W	0000_0000h	22.4.9/1057
2B8_0080	SAI Receive Control Register (I2S4_RCSR)	32	R/W	0000_0000h	22.4.10/ 1059
2B8_0084	SAI Receive Configuration 1 Register (I2S4_RCR1)	32	R/W	0000_0000h	22.4.11/ 1062
2B8_0088	SAI Receive Configuration 2 Register (I2S4_RCR2)	32	R/W	0000_0000h	22.4.12/ 1062
2B8_008C	SAI Receive Configuration 3 Register (I2S4_RCR3)	32	R/W	0000_0000h	22.4.13/ 1064
2B8_0090	SAI Receive Configuration 4 Register (I2S4_RCR4)	32	R/W	0000_0000h	22.4.14/ 1065
2B8_0094	SAI Receive Configuration 5 Register (I2S4_RCR5)	32	R/W	0000_0000h	22.4.15/ 1066
2B8_00A0	SAI Receive Data Register (I2S4_RDR0)	32	R	0000_0000h	22.4.16/ 1067
2B8_00C0	SAI Receive FIFO Register (I2S4_RFR0)	32	R	0000_0000h	22.4.17/ 1068
2B8_00E0	SAI Receive Mask Register (I2S4_RMR)	32	R/W	0000_0000h	22.4.18/ 1068
2B9_0000	SAI Transmit Control Register (I2S5_TCSR)	32	R/W	0000_0000h	22.4.1/1048
2B9_0004	SAI Transmit Configuration 1 Register (I2S5_TCR1)	32	R/W	0000_0000h	22.4.2/1051
2B9_0008	SAI Transmit Configuration 2 Register (I2S5_TCR2)	32	R/W	0000_0000h	22.4.3/1052
2B9_000C	SAI Transmit Configuration 3 Register (I2S5_TCR3)	32	R/W	0000_0000h	22.4.4/1053
2B9_0010	SAI Transmit Configuration 4 Register (I2S5_TCR4)	32	R/W	0000_0000h	22.4.5/1054
2B9_0014	SAI Transmit Configuration 5 Register (I2S5_TCR5)	32	R/W	0000_0000h	22.4.6/1056
2B9_0020	SAI Transmit Data Register (I2S5_TDR0)	32	W (always reads 0)	0000_0000h	22.4.7/1056
2B9_0040	SAI Transmit FIFO Register (I2S5_TFR0)	32	R	0000_0000h	22.4.8/1057
2B9_0060	SAI Transmit Mask Register (I2S5_TMR)	32	R/W	0000_0000h	22.4.9/1057
2B9_0080	SAI Receive Control Register (I2S5_RCSR)	32	R/W	0000_0000h	22.4.10/ 1059
2B9_0084	SAI Receive Configuration 1 Register (I2S5_RCR1)	32	R/W	0000_0000h	22.4.11/ 1062
2B9_0088	SAI Receive Configuration 2 Register (I2S5_RCR2)	32	R/W	0000_0000h	22.4.12/ 1062

Table continues on the next page...

I2S memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
2B9_008C	SAI Receive Configuration 3 Register (I2S5_RCR3)	32	R/W	0000_0000h	22.4.13/ 1064
2B9_0090	SAI Receive Configuration 4 Register (I2S5_RCR4)	32	R/W	0000_0000h	22.4.14/ 1065
2B9_0094	SAI Receive Configuration 5 Register (I2S5_RCR5)	32	R/W	0000_0000h	22.4.15/ 1066
2B9_00A0	SAI Receive Data Register (I2S5_RDR0)	32	R	0000_0000h	22.4.16/ 1067
2B9_00C0	SAI Receive FIFO Register (I2S5_RFR0)	32	R	0000_0000h	22.4.17/ 1068
2B9_00E0	SAI Receive Mask Register (I2S5_RMR)	32	R/W	0000_0000h	22.4.18/ 1068

22.4.1 SAI Transmit Control Register (I2Sx_TCSR)

Address: Base address + 0h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TE	0	0	BCE	0		0	SR	0			WSF	SEF	FEF	FWF	FRF
W							FR					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			WSIE	SEIE	FEIE	FWIE	FRIE	0			0			FWDE	FRDE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

I2Sx_TCSR field descriptions

Field	Description
31 TE	Transmitter Enable Enables/disables the transmitter. When software clears this field, the transmitter remains enabled, and this bit remains set, until the end of the current frame. 0 Transmitter is disabled. 1 Transmitter is enabled, or transmitter has been disabled and has not yet reached end of frame.
30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 BCE	Bit Clock Enable Enables the transmit bit clock, separately from the TE. This field is automatically set whenever TE is set. When software clears this field, the transmit bit clock remains enabled, and this bit remains set, until the end of the current frame. 0 Transmit bit clock is disabled. 1 Transmit bit clock is enabled.
27–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 FR	FIFO Reset Resets the FIFO pointers. Reading this field will always return zero. FIFO pointers should only be reset when the transmitter is disabled or the FIFO error flag is set. 0 No effect. 1 FIFO reset.
24 SR	Software Reset When set, resets the internal transmitter logic including the FIFO pointers. Software-visible registers are not affected, except for the status registers. 0 No effect. 1 Software reset.
23–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 WSF	Word Start Flag Indicates that the start of the configured word has been detected. Write a logic 1 to this field to clear this flag. 0 Start of word not detected. 1 Start of word detected.
19 SEF	Sync Error Flag Indicates that an error in the externally-generated frame sync has been detected. Write a logic 1 to this field to clear this flag. 0 Sync error not detected. 1 Frame sync error detected.

Table continues on the next page...

I2Sx_TCSR field descriptions (continued)

Field	Description
18 FEF	FIFO Error Flag Indicates that an enabled transmit FIFO has underrun. Write a logic 1 to this field to clear this flag. 0 Transmit underrun not detected. 1 Transmit underrun detected.
17 FWF	FIFO Warning Flag Indicates that an enabled transmit FIFO is empty. 0 No enabled transmit FIFO is empty. 1 Enabled transmit FIFO is empty.
16 FRF	FIFO Request Flag Indicates that the number of words in an enabled transmit channel FIFO is less than or equal to the transmit FIFO watermark. 0 Transmit FIFO watermark has not been reached. 1 Transmit FIFO watermark has been reached.
15–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 WSIE	Word Start Interrupt Enable Enables/disables word start interrupts. 0 Disables interrupt. 1 Enables interrupt.
11 SEIE	Sync Error Interrupt Enable Enables/disables sync error interrupts. 0 Disables interrupt. 1 Enables interrupt.
10 FEIE	FIFO Error Interrupt Enable Enables/disables FIFO error interrupts. 0 Disables the interrupt. 1 Enables the interrupt.
9 FWIE	FIFO Warning Interrupt Enable Enables/disables FIFO warning interrupts. 0 Disables the interrupt. 1 Enables the interrupt.
8 FRIE	FIFO Request Interrupt Enable Enables/disables FIFO request interrupts. 0 Disables the interrupt. 1 Enables the interrupt.

Table continues on the next page...

I2Sx_TCSR field descriptions (continued)

Field	Description
7–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 FWDE	FIFO Warning DMA Enable Enables/disables DMA requests. 0 Disables the DMA request. 1 Enables the DMA request.
0 FRDE	FIFO Request DMA Enable Enables/disables DMA requests. 0 Disables the DMA request. 1 Enables the DMA request.

22.4.2 SAI Transmit Configuration 1 Register (I2Sx_TCR1)

Address: Base address + 4h 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																											TFW				
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

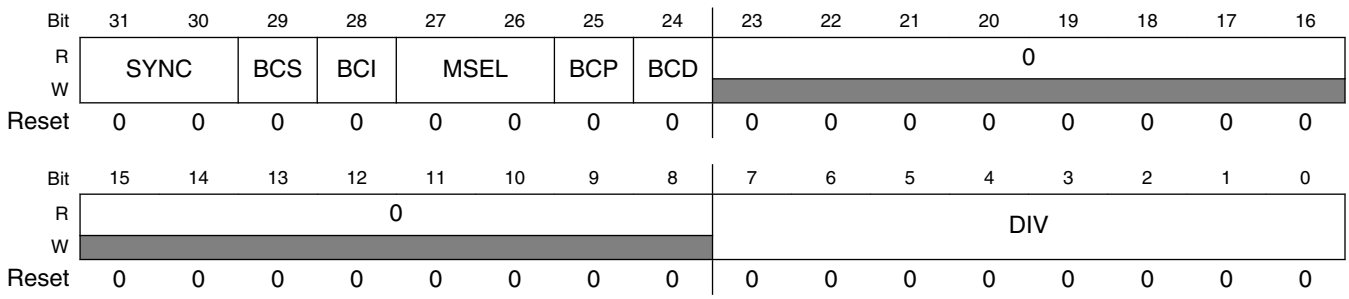
I2Sx_TCR1 field descriptions

Field	Description
31–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
TFW	Transmit FIFO Watermark Configures the watermark level for all enabled transmit channels.

22.4.3 SAI Transmit Configuration 2 Register (I2Sx_TCR2)

This register must not be altered when TCSR[TE] is set.

Address: Base address + 8h offset



I2Sx_TCR2 field descriptions

Field	Description
31–30 SYNC	Synchronous Mode Configures between asynchronous and synchronous modes of operation. When configured for a synchronous mode of operation, the receiver must be configured for asynchronous operation. 00 Asynchronous mode. 01 Synchronous with receiver.
29 BCS	Bit Clock Swap This field swaps the bit clock used by the transmitter. When the transmitter is configured in asynchronous mode and this bit is set, the transmitter is clocked by the receiver bit clock (SAI_RX_BCLK). This allows the transmitter and receiver to share the same bit clock, but the transmitter continues to use the transmit frame sync (SAI_TX_SYNC). When the transmitter is configured in synchronous mode, the transmitter BCS field and receiver BCS field must be set to the same value. When both are set, the transmitter and receiver are both clocked by the transmitter bit clock (SAI_TX_BCLK) but use the receiver frame sync (SAI_RX_SYNC). 0 Use the normal bit clock source. 1 Swap the bit clock source.
28 BCI	Bit Clock Input When this field is set and using an internally generated bit clock in either synchronous or asynchronous mode, the bit clock actually used by the transmitter is delayed by the pad output delay (the transmitter is clocked by the pad input as if the clock was externally generated). This has the effect of decreasing the data input setup time, but increasing the data output valid time. The slave mode timing from the datasheet should be used for the transmitter when this bit is set. In synchronous mode, this bit allows the transmitter to use the slave mode timing from the datasheet, while the receiver uses the master mode timing. This field has no effect when configured for an externally generated bit clock . 0 No effect. 1 Internal logic is clocked as if bit clock was externally generated.
27–26 MSEL	MCLK Select

Table continues on the next page...

I2Sx_TCR2 field descriptions (continued)

Field	Description
	Selects the audio Master Clock option used to generate an internally generated bit clock. This field has no effect when configured for an externally generated bit clock. **NOTE:** Depending on the device, some Master Clock options might not be available. See the chip-specific information for the meaning of each option. 00 Master Clock (MCLK) 1 option selected. 01 Master Clock (MCLK) 1 option selected. 10 Master Clock (MCLK) 2 option selected. 11 Master Clock (MCLK) 3 option selected.
25 BCP	Bit Clock Polarity Configures the polarity of the bit clock. 0 Bit clock is active high with drive outputs on rising edge and sample inputs on falling edge. 1 Bit clock is active low with drive outputs on falling edge and sample inputs on rising edge.
24 BCD	Bit Clock Direction Configures the direction of the bit clock. 0 Bit clock is generated externally in Slave mode. 1 Bit clock is generated internally in Master mode.
23–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
DIV	Bit Clock Divide Divides down the audio master clock to generate the bit clock when configured for an internal bit clock. The division value is (DIV + 1) * 2.

22.4.4 SAI Transmit Configuration 3 Register (I2Sx_TCR3)

This register must not be altered when TCSR[TE] is set.

Address: Base address + Ch offset

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

I2Sx_TCR3 field descriptions

Field	Description
31–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 TCE	Transmit Channel Enable Enables the corresponding data channel for transmit operation. A channel must be enabled before its FIFO is accessed. The width of TCE field = the number of transmit channels (call it N). For example, if TCE field is 2 bits wide, then bit position 16 refers to transmit channel 1 and bit position 17 refers to transmit channel 2. Setting bit 16 enables transmit channel 1, and setting bit 17 enables transmit channel 2. Setting bit N will enable transmit channel N. 0 Transmit data channel N is disabled. 1 Transmit data channel N is enabled.
15–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
WDFL	Word Flag Configuration Configures which word sets the start of word flag. The value written must be one less than the word number. For example, writing 0 configures the first word in the frame. When configured to a value greater than TCR4[FRSZ], then the start of word flag is never set.

22.4.5 SAI Transmit Configuration 4 Register (I2Sx_TCR4)

This register must not be altered when TCSR[TE] is set.

Address: Base address + 10h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0			0	0		0		0			FRSZ				
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			SYWD					0			MF	FSE	0	FSP	FSD
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

I2Sx_TCR4 field descriptions

Field	Description
31–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

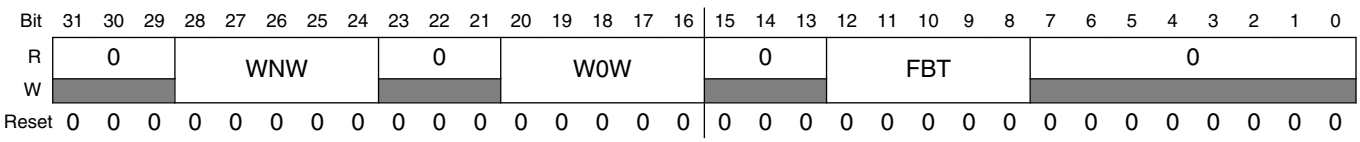
I2Sx_TCR4 field descriptions (continued)

Field	Description
25–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–16 FRSZ	Frame size Configures the number of words in each frame. The value written must be one less than the number of words in the frame. For example, write 0 for one word per frame. The maximum supported frame size is 32 words.
15–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–8 SYWD	Sync Width Configures the length of the frame sync in number of bit clocks. The value written must be one less than the number of bit clocks. For example, write 0 for the frame sync to assert for one bit clock only. The sync width cannot be configured longer than the first word of the frame.
7–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 MF	MSB First Configures whether the LSB or the MSB is transmitted first. 0 LSB is transmitted first. 1 MSB is transmitted first.
3 FSE	Frame Sync Early 0 Frame sync asserts with the first bit of the frame. 1 Frame sync asserts one bit before the first bit of the frame.
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 FSP	Frame Sync Polarity Configures the polarity of the frame sync. 0 Frame sync is active high. 1 Frame sync is active low.
0 FSD	Frame Sync Direction Configures the direction of the frame sync. 0 Frame sync is generated externally in Slave mode. 1 Frame sync is generated internally in Master mode.

22.4.6 SAI Transmit Configuration 5 Register (I2Sx_TCR5)

This register must not be altered when TCSR[TE] is set.

Address: Base address + 14h offset

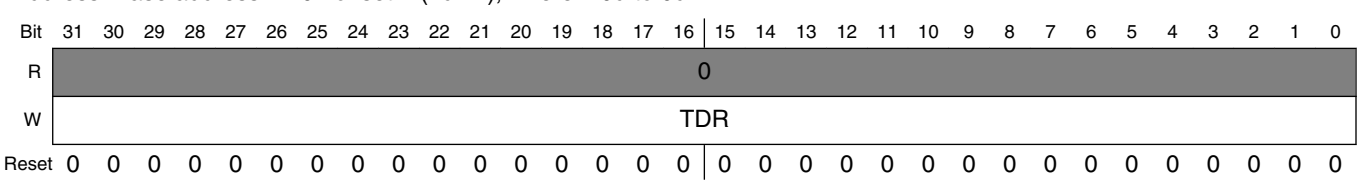


I2Sx_TCR5 field descriptions

Field	Description
31–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–24 WNW	Word N Width Configures the number of bits in each word, for each word except the first in the frame. The value written must be one less than the number of bits per word. Word width of less than 8 bits is not supported.
23–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–16 W0W	Word 0 Width Configures the number of bits in the first word in each frame. The value written must be one less than the number of bits in the first word. Word width of less than 8 bits is not supported if there is only one word per frame.
15–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–8 FBT	First Bit Shifted Configures the bit index for the first bit transmitted for each word in the frame. If configured for MSB First, the index of the next bit transmitted is one less than the current bit transmitted. If configured for LSB First, the index of the next bit transmitted is one more than the current bit transmitted. The value written must be greater than or equal to the word width when configured for MSB First. The value written must be less than or equal to 31-word width when configured for LSB First.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

22.4.7 SAI Transmit Data Register (I2Sx_TDR*n*)

Address: Base address + 20h offset + (4d × i), where i=0d to 0d



I2S*x*_TDR*n* field descriptions

Field	Description
TDR	Transmit Data Register The corresponding TCR3[TCE] bit must be set before accessing the channel's transmit data register. Writes to this register when the transmit FIFO is not full will push the data written into the transmit data FIFO. Writes to this register when the transmit FIFO is full are ignored.

22.4.8 SAI Transmit FIFO Register (I2S*x*_TFR*n*)

The MSB of the read and write pointers is used to distinguish between FIFO full and empty conditions. If the read and write pointers are identical, then the FIFO is empty. If the read and write pointers are identical except for the MSB, then the FIFO is full.

Address: Base address + 40h offset + (4d × i), where i=0d to 0d

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

I2S*x*_TFR*n* field descriptions

Field	Description
31 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
30–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–16 WFP	Write FIFO Pointer FIFO write pointer for transmit data channel.
15–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
RFP	Read FIFO Pointer FIFO read pointer for transmit data channel.

22.4.9 SAI Transmit Mask Register (I2S*x*_TMR)

This register is double-buffered and updates:

1. When TCSR[TE] is first set
2. At the end of each frame.

This allows the masked words in each frame to change from frame to frame.

Address: Base address + 60h offset

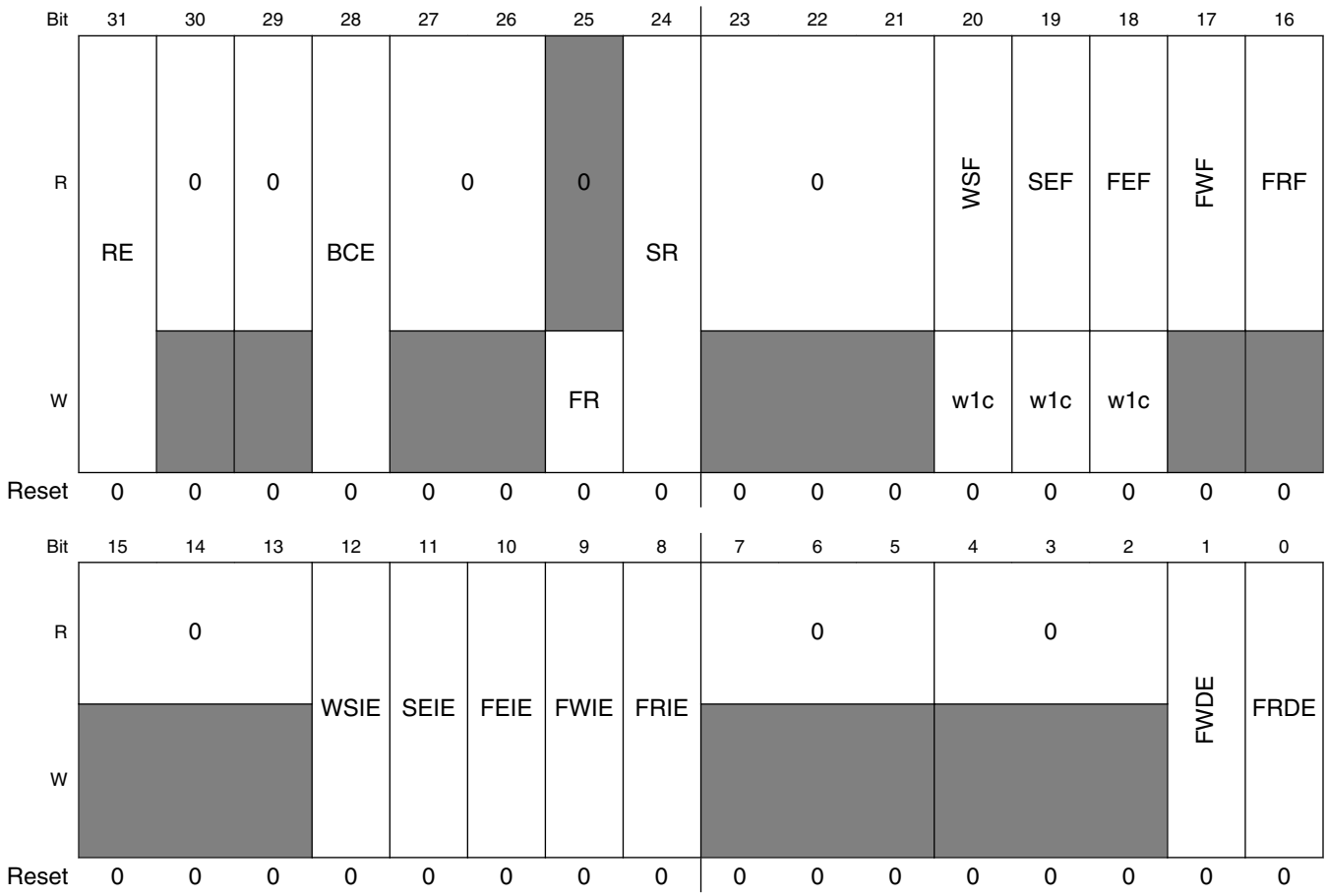
Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	TWM																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

I2Sx_TMR field descriptions

Field	Description
TWM	Transmit Word Mask Configures whether the transmit word is masked (transmit data pin tristated and transmit data not read from FIFO) for the corresponding word in the frame. 0 Word N is enabled. 1 Word N is masked. The transmit data pins are tri-stated when masked.

22.4.10 SAI Receive Control Register (I2Sx_RCSR)

Address: Base address + 80h offset



I2Sx_RCSR field descriptions

Field	Description
31 RE	Receiver Enable Enables/disables the receiver. When software clears this field, the receiver remains enabled, and this bit remains set, until the end of the current frame. 0 Receiver is disabled. 1 Receiver is enabled, or receiver has been disabled and has not yet reached end of frame.
30 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 BCE	Bit Clock Enable Enables the receive bit clock, separately from RE. This field is automatically set whenever RE is set. When software clears this field, the receive bit clock remains enabled, and this field remains set, until the end of the current frame.

Table continues on the next page...

I2Sx_RCSR field descriptions (continued)

Field	Description
	0 Receive bit clock is disabled. 1 Receive bit clock is enabled.
27–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25 FR	FIFO Reset Resets the FIFO pointers. Reading this field will always return zero. FIFO pointers should only be reset when the receiver is disabled or the FIFO error flag is set. 0 No effect. 1 FIFO reset.
24 SR	Software Reset Resets the internal receiver logic including the FIFO pointers. Software-visible registers are not affected, except for the status registers. 0 No effect. 1 Software reset.
23–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20 WSF	Word Start Flag Indicates that the start of the configured word has been detected. Write a logic 1 to this field to clear this flag. 0 Start of word not detected. 1 Start of word detected.
19 SEF	Sync Error Flag Indicates that an error in the externally-generated frame sync has been detected. Write a logic 1 to this field to clear this flag. 0 Sync error not detected. 1 Frame sync error detected.
18 FEF	FIFO Error Flag Indicates that an enabled receive FIFO has overflowed. Write a logic 1 to this field to clear this flag. 0 Receive overflow not detected. 1 Receive overflow detected.
17 FWF	FIFO Warning Flag Indicates that an enabled receive FIFO is full. 0 No enabled receive FIFO is full. 1 Enabled receive FIFO is full.
16 FRF	FIFO Request Flag Indicates that the number of words in an enabled receive channel FIFO is greater than the receive FIFO watermark.

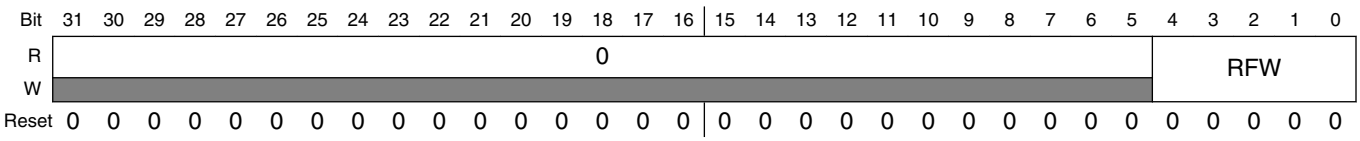
Table continues on the next page...

I2Sx_RCSR field descriptions (continued)

Field	Description
	0 Receive FIFO watermark not reached. 1 Receive FIFO watermark has been reached.
15–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 WSIE	Word Start Interrupt Enable Enables/disables word start interrupts. 0 Disables interrupt. 1 Enables interrupt.
11 SEIE	Sync Error Interrupt Enable Enables/disables sync error interrupts. 0 Disables interrupt. 1 Enables interrupt.
10 FEIE	FIFO Error Interrupt Enable Enables/disables FIFO error interrupts. 0 Disables the interrupt. 1 Enables the interrupt.
9 FWIE	FIFO Warning Interrupt Enable Enables/disables FIFO warning interrupts. 0 Disables the interrupt. 1 Enables the interrupt.
8 FRIE	FIFO Request Interrupt Enable Enables/disables FIFO request interrupts. 0 Disables the interrupt. 1 Enables the interrupt.
7–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4–2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 FWDE	FIFO Warning DMA Enable Enables/disables DMA requests. 0 Disables the DMA request. 1 Enables the DMA request.
0 FRDE	FIFO Request DMA Enable Enables/disables DMA requests. 0 Disables the DMA request. 1 Enables the DMA request.

22.4.11 SAI Receive Configuration 1 Register (I2Sx_RCR1)

Address: Base address + 84h offset



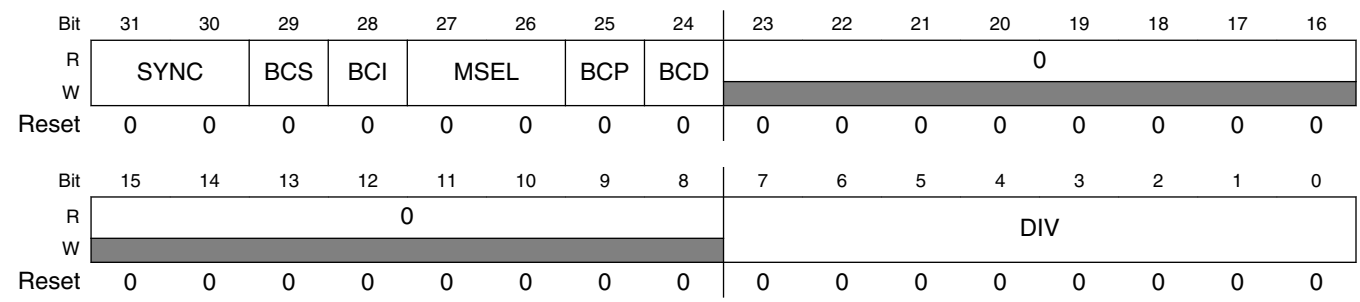
I2Sx_RCR1 field descriptions

Field	Description
31–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
RFW	Receive FIFO Watermark Configures the watermark level for all enabled receiver channels.

22.4.12 SAI Receive Configuration 2 Register (I2Sx_RCR2)

This register must not be altered when RCSR[RE] is set.

Address: Base address + 88h offset



I2Sx_RCR2 field descriptions

Field	Description
31–30 SYNC	Synchronous Mode Configures between asynchronous and synchronous modes of operation. When configured for a synchronous mode of operation, the transmitter must be configured for asynchronous operation. 00 Asynchronous mode. 01 Synchronous with transmitter.
29 BCS	Bit Clock Swap This field swaps the bit clock used by the receiver. When the receiver is configured in asynchronous mode and this bit is set, the receiver is clocked by the transmitter bit clock (SAI_TX_BCLK). This allows the transmitter and receiver to share the same bit clock, but the receiver continues to use the receiver frame sync (SAI_RX_SYNC).

Table continues on the next page...

I2Sx_RCR2 field descriptions (continued)

Field	Description
	When the receiver is configured in synchronous mode, the transmitter BCS field and receiver BCS field must be set to the same value. When both are set, the transmitter and receiver are both clocked by the receiver bit clock (SAI_RX_BCLK) but use the transmitter frame sync (SAI_TX_SYNC). 0 Use the normal bit clock source. 1 Swap the bit clock source.
28 BCI	Bit Clock Input When this field is set and using an internally generated bit clock in either synchronous or asynchronous mode, the bit clock actually used by the receiver is delayed by the pad output delay (the receiver is clocked by the pad input as if the clock was externally generated). This has the effect of decreasing the data input setup time, but increasing the data output valid time. The slave mode timing from the datasheet should be used for the receiver when this bit is set. In synchronous mode, this bit allows the receiver to use the slave mode timing from the datasheet, while the transmitter uses the master mode timing. This field has no effect when configured for an externally generated bit clock . 0 No effect. 1 Internal logic is clocked as if bit clock was externally generated.
27–26 MSEL	MCLK Select Selects the audio Master Clock option used to generate an internally generated bit clock. This field has no effect when configured for an externally generated bit clock. **NOTE:** Depending on the device, some Master Clock options might not be available. See the chip-specific information for the availability and chip-specific meaning of each option. 00 Bus Clock selected. 01 Master Clock (MCLK) 1 option selected. 10 Master Clock (MCLK) 2 option selected. 11 Master Clock (MCLK) 3 option selected.
25 BCP	Bit Clock Polarity Configures the polarity of the bit clock. 0 Bit Clock is active high with drive outputs on rising edge and sample inputs on falling edge. 1 Bit Clock is active low with drive outputs on falling edge and sample inputs on rising edge.
24 BCD	Bit Clock Direction Configures the direction of the bit clock. 0 Bit clock is generated externally in Slave mode. 1 Bit clock is generated internally in Master mode.
23–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
DIV	Bit Clock Divide Divides down the audio master clock to generate the bit clock when configured for an internal bit clock. The division value is (DIV + 1) * 2.

22.4.13 SAI Receive Configuration 3 Register (I2Sx_RCR3)

This register must not be altered when RCSR[RE] is set.

Address: Base address + 8Ch offset

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

I2Sx_RCR3 field descriptions

Field	Description
31–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23–17 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
16 RCE	Receive Channel Enable Enables the corresponding data channel for receive operation. A channel must be enabled before its FIFO is accessed. The width of RCE field = the number of receive channels (call it N). For example, if RCE field is 2 bits wide, then bit position 16 refers to receive channel 1 and bit position 17 refers to receive channel 2. Setting bit 16 enables receive channel 1, and setting bit 17 enables receive channel 2. Setting bit N will enable receive channel N. 0 Receive data channel N is disabled. 1 Receive data channel N is enabled.
15–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
WDFL	Word Flag Configuration Configures which word the start of word flag is set. The value written should be one less than the word number (for example, write zero to configure for the first word in the frame). When configured to a value greater than the Frame Size field, then the start of word flag is never set.

22.4.14 SAI Receive Configuration 4 Register (I2Sx_RCR4)

This register must not be altered when RCSR[RE] is set.

Address: Base address + 90h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0			0	0		0		0			FRSZ				
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			SYWD					0			MF	FSE	0	FSP	FSD
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

I2Sx_RCR4 field descriptions

Field	Description
31–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
27–26 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
25–24 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
23–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–16 FRSZ	Frame Size Configures the number of words in each frame. The value written must be one less than the number of words in the frame. For example, write 0 for one word per frame. The maximum supported frame size is 32 words.
15–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–8 SYWD	Sync Width Configures the length of the frame sync in number of bit clocks. The value written must be one less than the number of bit clocks. For example, write 0 for the frame sync to assert for one bit clock only. The sync width cannot be configured longer than the first word of the frame.
7–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 MF	MSB First Configures whether the LSB or the MSB is received first. 0 LSB is received first. 1 MSB is received first.
3 FSE	Frame Sync Early

Table continues on the next page...

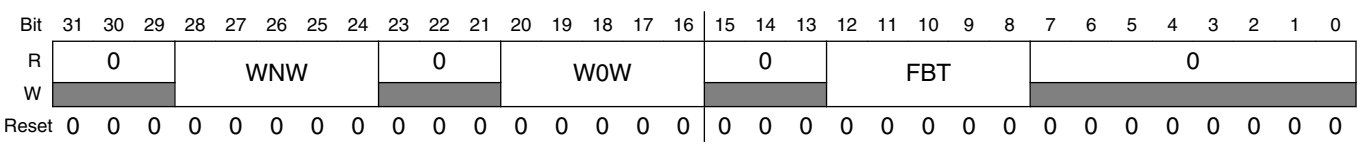
I2Sx_RCR4 field descriptions (continued)

Field	Description
	0 Frame sync asserts with the first bit of the frame. 1 Frame sync asserts one bit before the first bit of the frame.
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
1 FSP	Frame Sync Polarity Configures the polarity of the frame sync. 0 Frame sync is active high. 1 Frame sync is active low.
0 FSD	Frame Sync Direction Configures the direction of the frame sync. 0 Frame Sync is generated externally in Slave mode. 1 Frame Sync is generated internally in Master mode.

22.4.15 SAI Receive Configuration 5 Register (I2Sx_RCR5)

This register must not be altered when RCSR[RE] is set.

Address: Base address + 94h offset



I2Sx_RCR5 field descriptions

Field	Description
31–29 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
28–24 WNW	Word N Width Configures the number of bits in each word, for each word except the first in the frame. The value written must be one less than the number of bits per word. Word width of less than 8 bits is not supported.
23–21 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
20–16 W0W	Word 0 Width Configures the number of bits in the first word in each frame. The value written must be one less than the number of bits in the first word. Word width of less than 8 bits is not supported if there is only one word per frame.
15–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12–8 FBT	First Bit Shifted

Table continues on the next page...

I2S*x*_RCR5 field descriptions (continued)

Field	Description
	Configures the bit index for the first bit received for each word in the frame. If configured for MSB First, the index of the next bit received is one less than the current bit received. If configured for LSB First, the index of the next bit received is one more than the current bit received. The value written must be greater than or equal to the word width when configured for MSB First. The value written must be less than or equal to 31-word width when configured for LSB First.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

22.4.16 SAI Receive Data Register (I2S*x*_RDR*n*)

Reading this register introduces one additional peripheral clock wait state on each read.

Address: Base address + A0h offset + (4d × 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	RDR																															
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

I2S*x*_RDR*n* field descriptions

Field	Description
RDR	Receive Data Register The corresponding RCR3[RCE] bit must be set before accessing the channel's receive data register. Reads from this register when the receive FIFO is not empty will return the data from the top of the receive FIFO. Reads from this register when the receive FIFO is empty are ignored.

22.4.17 SAI Receive FIFO Register (I2Sx_RFRn)

The MSB of the read and write pointers is used to distinguish between FIFO full and empty conditions. If the read and write pointers are identical, then the FIFO is empty. If the read and write pointers are identical except for the MSB, then the FIFO is full.

Address: Base address + C0h offset + (4d × i), where i=0d to 0d

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

I2Sx_RFRn field descriptions

Field	Description
31–22 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
21–16 WFP	Write FIFO Pointer FIFO write pointer for receive data channel.
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
RFP	Read FIFO Pointer FIFO read pointer for receive data channel.

22.4.18 SAI Receive Mask Register (I2Sx_RMR)

This register is double-buffered and updates:

1. When RCSR[RE] is first set
2. At the end of each frame

This allows the masked words in each frame to change from frame to frame.

Address: Base address + E0h 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	RWM																															
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

I2Sx_RMR field descriptions

Field	Description
RWM	Receive Word Mask Configures whether the receive word is masked (received data ignored and not written to receive FIFO) for the corresponding word in the frame. 0 Word N is enabled. 1 Word N is masked.

22.5 Functional description

This section provides a complete functional description of the block.

22.5.1 SAI clocking

The SAI clocks include:

- The audio master clock
- The bit clock
- The bus clock

22.5.1.1 Audio master clock

The audio master clock is used to generate the bit clock when the receiver or transmitter is configured for an internally generated bit clock. The transmitter and receiver can independently select between the bus clock and up to three audio master clocks to generate the bit clock.

The audio master clock generation and selection is chip-specific. Refer to chip-specific clocking information about how the audio master clocks are generated.

22.5.1.2 Bit clock

The SAI transmitter and receiver support asynchronous free-running bit clocks that can be generated internally from an audio master clock or supplied externally. There is also the option for synchronous bit clock and frame sync operation between the receiver and transmitter or between multiple SAI peripherals.

- If both transmitter and receiver are configured for asynchronous operation, then the transmitter and receiver will each use *their own* bit clock and frame sync.
- If the *transmitter* is configured for asynchronous mode and the receiver is configured for synchronous mode, then both transmitter and receiver will use the *transmitter* bit clock and frame sync.
- If the *receiver* is configured for asynchronous mode and the transmitter is configured for synchronous mode, then both transmitter and receiver will use the *receiver* bit clock and frame sync.

Note that the software configures synchronous or asynchronous mode, and that choice selects the bit clock/frame sync used.

Externally generated bit clocks must be:

- Enabled before the SAI transmitter or receiver is enabled
- Disabled after the SAI transmitter or receiver is disabled and completes its current frames

22.5.1.3 Bus clock

The bus clock is used by the control and configuration registers and to generate synchronous interrupts and DMA requests.

NOTE

Although there is no specific minimum bus clock frequency specified, the bus clock frequency must be fast enough (relative to the bit clock frequency) to ensure that the FIFOs can be serviced, without generating either a transmitter FIFO underrun or receiver FIFO overflow condition.

22.5.2 SAI resets

The SAI is asynchronously reset on system reset. The SAI has a software reset and a FIFO reset.

22.5.2.1 Software reset

The SAI transmitter includes a software reset that resets all transmitter internal logic, including the bit clock generation, status flags, and FIFO pointers. It does not reset the configuration registers. The software reset remains asserted until cleared by software.

The SAI receiver includes a software reset that resets all receiver internal logic, including the bit clock generation, status flags and FIFO pointers. It does not reset the configuration registers. The software reset remains asserted until cleared by software.

22.5.2.2 FIFO reset

The SAI transmitter includes a FIFO reset that synchronizes the FIFO write pointer to the same value as the FIFO read pointer. This empties the FIFO contents and is to be used after TCSR[FEF] is set, and before the FIFO is re-initialized and TCSR[FEF] is cleared. The FIFO reset is asserted for one cycle only.

The SAI receiver includes a FIFO reset that synchronizes the FIFO read pointer to the same value as the FIFO write pointer. This empties the FIFO contents and is to be used after the RCSR[FEF] is set and any remaining data has been read from the FIFO, and before the RCSR[FEF] is cleared. The FIFO reset is asserted for one cycle only.

22.5.3 Synchronous modes

The SAI transmitter and receiver can operate synchronously to each other.

22.5.3.1 Synchronous mode

The SAI transmitter and receiver can be configured to operate with synchronous bit clock and frame sync.

If the transmitter bit clock and frame sync are to be used by both the transmitter and receiver:

- The transmitter must be configured for asynchronous operation and the receiver for synchronous operation.
- In synchronous mode, the receiver is enabled only when both the transmitter and receiver are enabled.
- It is recommended that the transmitter is the last enabled and the first disabled.

If the receiver bit clock and frame sync are to be used by both the transmitter and receiver:

- The receiver must be configured for asynchronous operation and the transmitter for synchronous operation.
- In synchronous mode, the transmitter is enabled only when both the receiver and transmitter are both enabled.
- It is recommended that the receiver is the last enabled and the first disabled.

When operating in synchronous mode, only the bit clock, frame sync, and transmitter/receiver enable are shared. The transmitter and receiver otherwise operate independently, although configuration registers must be configured consistently across both the transmitter and receiver.

22.5.4 Frame sync configuration

When enabled, the SAI continuously transmits and/or receives frames of data. Each frame consists of a fixed number of words and each word consists of a fixed number of bits. Within each frame, any given word can be masked causing the receiver to ignore that word and the transmitter to tri-state for the duration of that word.

The frame sync signal is used to indicate the start of each frame. A valid frame sync requires a rising edge (if active high) or falling edge (if active low) to be detected and the transmitter or receiver cannot be busy with a previous frame. A valid frame sync is also ignored (slave mode) or not generated (master mode) for the first four bit clock cycles after enabling the transmitter or receiver.

The transmitter and receiver frame sync can be configured independently with any of the following options:

- Externally generated or internally generated
- Active high or active low
- Assert with the first bit in frame or asserts one bit early
- Assert for a duration between 1 bit clock and the first word length
- Frame length from 1 to 32 words per frame
- Word length to support 8 to 32 bits per word
    - First word length and remaining word lengths can be configured separately
- Words can be configured to transmit/receive MSB first or LSB first

These configuration options cannot be changed after the SAI transmitter or receiver is enabled.

22.5.5 Data FIFO

Each transmit and receive channel includes a FIFO of size 32 × 32-bit. The FIFO data is accessed using the SAI Transmit/Receive Data Registers.

22.5.5.1 Data alignment

Data in the FIFO can be aligned anywhere within the 32-bit wide register through the use of the First Bit Shifted configuration field, which selects the bit index (between 31 and 0) of the first bit shifted.

Examples of supported data alignment and the required First Bit Shifted configuration are illustrated in Figure 22-2 for LSB First configurations and Figure 22-3 for MSB First configurations.

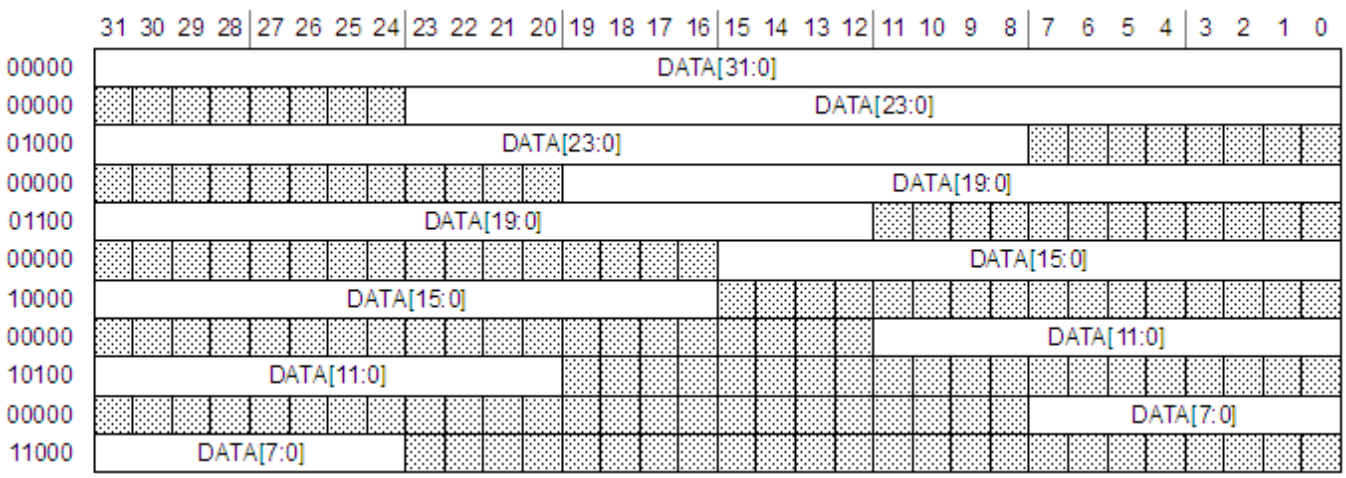


Figure 22-2. SAI first bit shifted, LSB first

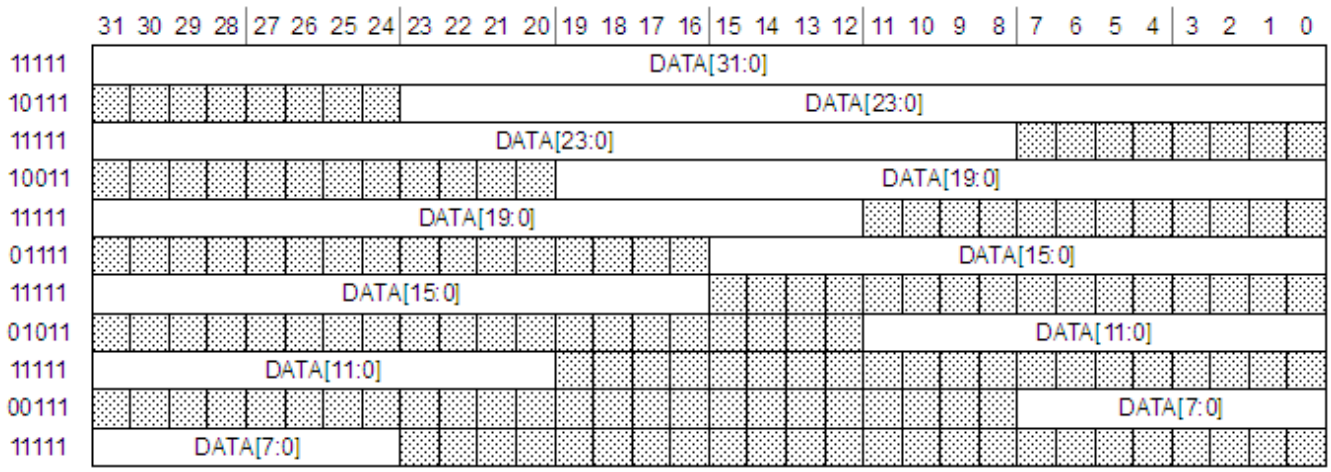


Figure 22-3. SAI first bit shifted, MSB first

22.5.5.2 FIFO pointers

When writing to a TDR, the WFP of the corresponding TFR increments after each valid write. The SAI supports 8-bit, 16-bit and 32-bit writes to the TDR and the FIFO pointer will increment after each individual write. Note that 8-bit writes should only be used when transmitting up to 8-bit data and 16-bit writes should only be used when transmitting up to 16-bit data.

Writes to a TDR are ignored if the corresponding bit of TCR3[TCE] is clear or if the FIFO is full. If the Transmit FIFO is empty, the TDR must be written at least three bit clocks before the start of the next unmasked word to avoid a FIFO underrun.

When reading an RDR, the RFP of the corresponding RFR increments after each valid read. The SAI supports 8-bit, 16-bit and 32-bit reads from the RDR and the FIFO pointer will increment after each individual read. Note that 8-bit reads should only be used when receiving up to 8-bit data and 16-bit reads should only be used when receiving up to 16-bit data.

Reads from an RDR are ignored if the corresponding bit of RCR3[RCE] is clear or if the FIFO is empty. If the Receive FIFO is full, the RDR must be read at least three bit clocks before the end of an unmasked word to avoid a FIFO overrun.

22.5.6 Word mask register

The SAI transmitter and receiver each contain a word mask register, namely TMR and RMR, that can be used to mask any word in the frame. Because the word mask register is double buffered, software can update it before the end of each frame to mask a particular word in the next frame.

The TMR causes the Transmit Data pin to be tri-stated for the length of each selected word and the transmit FIFO is not read for masked words.

The RMR causes the received data for each selected word to be discarded and not written to the receive FIFO.

22.5.7 Interrupts and DMA requests

The SAI transmitter and receiver generate separate interrupts and separate DMA requests, but support the same status flags.

22.5.7.1 FIFO request flag

The FIFO request flag is set based on the number of entries in the FIFO and the FIFO watermark configuration.

The transmit FIFO request flag is set when the number of entries in any of the enabled transmit FIFOs is less than or equal to the transmit FIFO watermark configuration and is cleared when the number of entries in each enabled transmit FIFO is greater than the transmit FIFO watermark configuration.

The receive FIFO request flag is set when the number of entries in any of the enabled receive FIFOs is greater than the receive FIFO watermark configuration and is cleared when the number of entries in each enabled receive FIFO is less than or equal to the receive FIFO watermark configuration.

The FIFO request flag can generate an interrupt or a DMA request.

22.5.7.2 FIFO warning flag

The FIFO warning flag is set based on the number of entries in the FIFO.

The transmit warning flag is set when the number of entries in any of the enabled transmit FIFOs is empty and is cleared when the number of entries in each enabled transmit FIFO is not empty.

The receive warning flag is set when the number of entries in any of the enabled receive FIFOs is full and is cleared when the number of entries in each enabled receive FIFO is not full.

The FIFO warning flag can generate an Interrupt or a DMA request.

22.5.7.3 FIFO error flag

The transmit FIFO error flag is set when the any of the enabled transmit FIFOs underflow. After it is set, all enabled transmit channels repeat the last valid word read from the transmit FIFO until TCSR[FEF] is cleared and the next transmit frame starts. All enabled transmit FIFOs must be reset and initialized with new data before TCSR[FEF] is cleared.

RCSR[FEF] is set when the any of the enabled receive FIFOs overflow. After it is set, all enabled receive channels discard received data until RCSR[FEF] is cleared and the next next receive frame starts. All enabled receive FIFOs should be emptied before RCSR[FEF] is cleared.

The FIFO error flag can generate only an interrupt.

22.5.7.4 Sync error flag

The sync error flag, TCSR[SEF] or RCSR[SEF], is set when configured for an externally generated frame sync and the external frame sync asserts when the transmitter or receiver is busy with the previous frame. The external frame sync assertion is ignored and the sync error flag is set. When the sync error flag is set, the transmitter or receiver continues checking for frame sync assertion when idle or at the end of each frame.

The sync error flag can generate an interrupt only.

22.5.7.5 Word start flag

The word start flag is set at the start of the second bit clock for the selected word, as configured by the Word Flag register field.

The word start flag can generate an interrupt only.

Chapter 23
Multi Mode DDR Controller (MMDC)

23.1 The MMDC module as implemented on the chip

This section provides details about how the MMDC module is implemented on the chip.

23.1.1 LS1012A MMDC module integration

The following table describes the MMDC module integration into this chip:

Table 23-1. MMDC module integration

Module	Module Base address
DDR controller	108_0000

23.1.2 LS1012A MMDC signals

The following table lists the SoC signal names and their corresponding MMDC module signal names used in this chapter:

Table 23-2. LS1012A MMDC signals

LS1012A signal name	MMDC module signal
D1_MDQ[15:0]	DRAM_SDQ[15:0]
D1_MDM[1:0]	DRAM_DQM[1:0]
D1_MDQS[1:0]	DRAM_SDQS[1:0]
D1_MDQS_B[1:0]	DRAM_SDQS_B[1:0]
D1_MBA[2:0]	DRAM_SDBA[2:0]
D1_MA[15:0]	DRAM_ADDR[15:0]
D1_MWE_B	DRAM_SDWE

Table continues on the next page...

Table 23-2. LS1012A MMDC signals (continued)

LS1012A signal name	MMDC module signal
D1_MRAS_B	DRAM_RAS
D1_MCAS_B	DRAM_CAS
D1_MCS_B	DRAM_CS
D1_MCKE	DRAM_SDCKE
D1_MCK	DRAM_SDCLK_P
D1_MCK_B	DRAM_SDCLK_N
D1_MODT	DRAM_ODT
D1_MDIC	Unused
Unused	DRAM_RESET

23.1.3 LS1012A MMDC module special consideration

The MMDC module implements the following parameter settings in the chip:

Table 23-3. LS1012A MMDC parameter settings

MMDC parameters	LS1012A parameter value
	DDR1
Data bus width	16-bit (supports one x16 or two x8 devices totaling to 16-bit data)
ECC support	No
No. of Chip select	1
DDR mode	DDR3L (DDR3 is not supported)
LPDDR2 support	No
Debug mode support	No
Discrete memory	Only discrete memories are supported in the chip.

NOTE

For the DDR controller, in case a location is accessed outside the total available system memory, error will not be reported. Software needs to ensure that no access is made to unconnected chip select. In case of an access to a non-initialized or unconnected chip select, the behavior might be unexpected (but no error indicated).

NOTE

Any reference to the features that are not supported in the chip should be ignored.

23.2 Overview

MMDC is a multi-mode DDR controller that supports DDR3L x16 memory type. MMDC is configurable and optimized for high performance.

The following figure shows the MMDC block diagram.

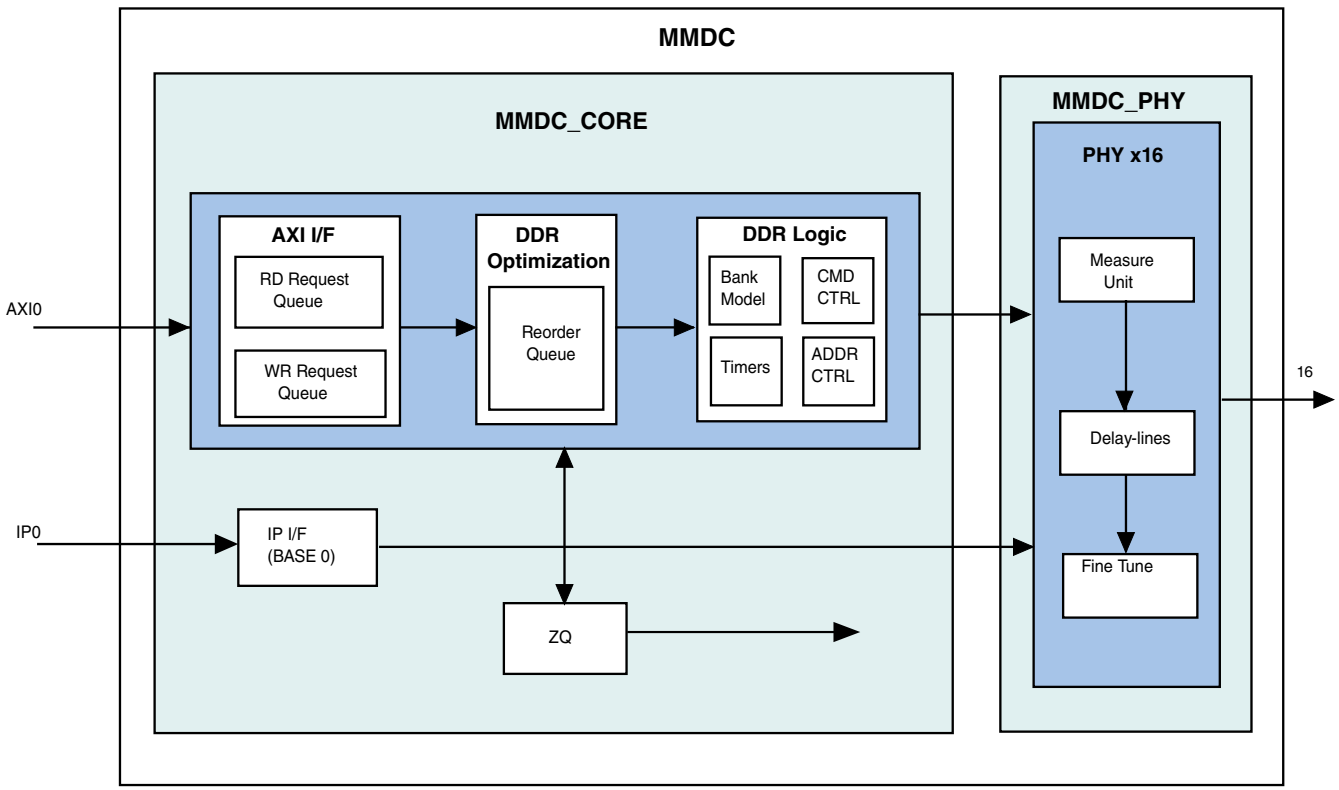


Figure 23-1. MMDC block diagram

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 500 MHz.

The internal memory map(configuration registers) of the MMDC can be configured through an IP channel (IP0).

23.2.1 MMDC feature summary

The table found here summarizes the MMDC features.

Table 23-4. MMDC feature summary

Feature	Details
DDR standards	• DDR3L x16 • Does not support LPDDR1 or DDR2

DDR interface	• x16 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
	• One chip select
	• Up to 2 Gbytes of address space
	• Supports burst length of 8 (aligned) for DDR3L

DDR performance	• MMDC running at up to 500 MHz (1000 MT/s). • 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 • Special optimization in case of non-aligned wrap accesses in 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.

DDR calibration and delay-lines.	• Supports various calibration processes which can be performed either automatically (hardware) or manually (software) towards CS0 The following calibration processes are supported:
	  • ZQ calibration for external DDR device in (DDR3L through ZQ calibration command and 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 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 DDR3L 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.

Table continues on the next page...

Table 23-4. MMDC feature summary (continued)

Feature	Details

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.
	• While CS (chip-select) is inactive (high) the command and address buses are not toggling 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

23.3 Functional Description

This section provides a complete functional description of the block.

23.3.1 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_SDCKE signals are kept low during the power-up and reset sequences of the chip (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. Configure the duty cycle control register (MPDCCR) to 0x249244A4 to tune the clock timing parameters.
7. Perform a ZQ calibration of the MMDC module to correctly initialize drive strengths.
8. 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_SDCKE as defined by JEDEC.
9. 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.
10. Program the DDR mode registers by configuring the appropriate command and address at the MDSCR register.
11. Configure the power down and self-refresh entry and exit parameters at the MDPDC and MAPSR registers.
12. Configure the ZQ scheme at the MPZQHWCTRL and MPZQLP2CTRL registers.
13. Configure and activate the periodic refresh scheme at the MDREF register.
14. 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 process. Before starting any calibration process, you must disable the periodic refresh scheme (MDREF[REF_SEL] = 11) and then issue a manual refresh command by configuring MDSCR[CMD] to 2h. For further information, see Calibration Process.

23.3.2 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.

23.3.3 MMDC Address Space

23.3.3.1 Address decoding

MMDC supports up to two consecutive chip selects, each with the same density.

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 8 banks in the DDR device
- MDCTL[DSIZ]—Defines the DDR data bus width of x16
- 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 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 assignment for the column. The total density is 256 MWords (512 Mbytes for x16).

Table 23-5. Address decoding—bank interleaving off

AXI ADDRESS	x16 DDR
A29	—
A28	BANK[2]
A27	BANK[1]
A26	BANK[0]
A25	ROW[14]
A24	ROW[13]
A23	ROW[12]
A22	ROW[11]
A21	ROW[10]
A20	ROW[9]
A19	ROW[8]
A18	ROW[7]
A17	ROW[6]
A16	ROW[5]
A15	ROW[4]
A14	ROW[3]
A13	ROW[2]
A12	ROW[1]
A11	ROW[0]
A10	COL[9]
A9	COL[8]
A8	COL[7]
A7	COL[6]
A6	COL[5]
A5	COL[4]
A4	COL[3]
A3	COL[2]
A2	COL[1]
A1	COL[0]
A0	—

Table 23-6. Address decoding—bank interleaving on

AXI ADDRESS	x16 DDR
A29	—

Table continues on the next page...

Table 23-6. Address decoding—bank interleaving on (continued)

AXI ADDRESS	x16 DDR
A28	ROW[14]
A27	ROW[13]
A26	ROW[12]
A25	ROW[11]
A24	ROW[10]
A23	ROW[9]
A22	ROW[8]
A21	ROW[7]
A20	ROW[6]
A19	ROW[5]
A18	ROW[4]
A17	ROW[3]
A16	ROW[2]
A15	ROW[1]
A14	ROW[0]
A13	BANK[2]
A12	BANK[1]
A11	BANK[0]
A10	COL[9]
A9	COL[8]
A8	COL[7]
A7	COL[6]
A6	COL[5]
A5	COL[4]
A4	COL[3]
A3	COL[2]
A2	COL[1]
A1	COL[0]
A0	—

NOTE

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

23.3.4 Power Saving and Clock Frequency Change modes

23.3.4.1 Power saving general

MMDC supports multiple DDR power saving modes.

NOTE

By 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 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)
     - 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 MDPDC register:
   - PWDT_0 - defines the number of idle cycles before entering power down.
   - SLOW_PD - In case of slow precharge power down then this bit should be set as well.
   - Few paramters must be configured in addition:
     - Timing parameters at MDCFG0[tXP and tXPDLL].
     - ODT timing at MDOTC[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 MDPDC field: PRCT_0.

23.3.4.2 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 treats 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 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:

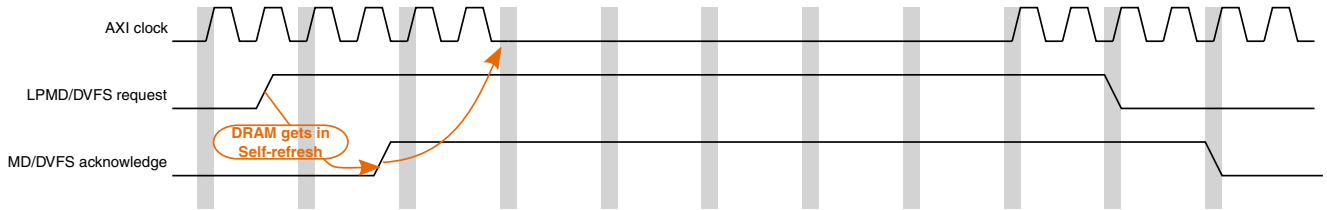


Figure 23-2. LPMD/DVFS Hardware/Software Handshaking

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.

23.3.5 Reset

23.3.5.1 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.

23.3.5.2 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.

23.3.6 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:

- 32 KHz clock
- 64 KHz 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 3.9µs (tREFI, refresh command every 3.9µs).

Table 23-7. 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)	0x0 (64 KHz)	not needed	tRFC * 8
2	Issue 4 refresh commands every 15,625ns	0x3 (4 refreshes)	0x1(32 KHz)	not needed	tRFC * 4
3	Issue 2 refresh commands every 7800ns	0x1(2 refreshes)	0x2 (REF_CNT)	7800/2.5 = 3120 (0xC30)	tRFC * 2
4	Issue 1 refresh command every 3900 ns	0x0 (1 refresh)	0x2 (REF_CNT)	3900/2.5 = 1560(0x618)	tRFC

23.3.7 Burst Length options towards DDR

The MMDC supports burst lengths which can be configured through MD+CTL[BL] as follows:

- In DDR3L mode, only burst length 8 can be used.

In DDR3L mode read/write accesses to the DDR are always 8 words (x16) and aligned in according to JEDEC standards.

In case of AXI INCREMENT, accesses that are not aligned the irrelevant data is masked in write accesses 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.

23.3.8 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.

23.3.9 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

23.4 Performance

23.4.1 Arbitration and reordering mechanism

23.4.1.1 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.

23.4.1.2 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.

23.4.1.3 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

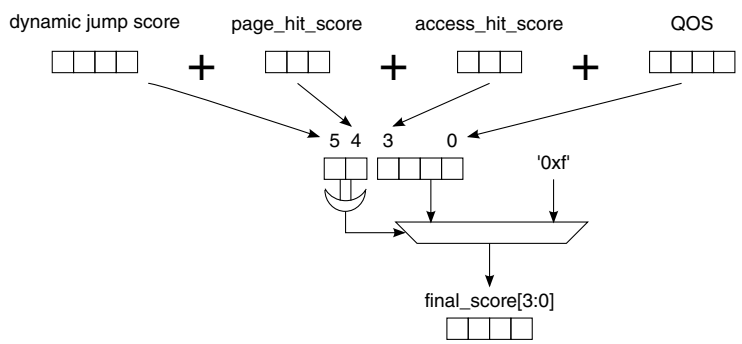


Figure 23-3. Dynamic score/priority calculation

23.4.1.4 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.

23.4.2 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

23.4.3 Special Optimization for accesses towards DDR3L

In case an AXI read/write wrap non-aligned access is acknowledged in DDR3L 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 DDR3L must be aligned.

For example: AXI write access with size of 128 bits (awsize[2:0]=3'b100), length of 4 (awlen[3:0]=4'b0011) towards DDR3L x64 (burst length 8). In that case the AXI wrap boundary is 16Bx4=64B (0x40) and the DDR3L 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 DDR3L 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.

23.5 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

23.6 DLL Switching

23.6.1 DLL Off mode

DLL Off mode is supported only in DDR3L and allows operation of the DDR in low frequency (i.e. below 125 MHz 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.

23.7 ODT Configuration

The MMDC supports one DRAM_ODT signal (DRAM_ODT for each DRAM_CS) in DDR3L 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 signal as well as configuration of several related timing.

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.

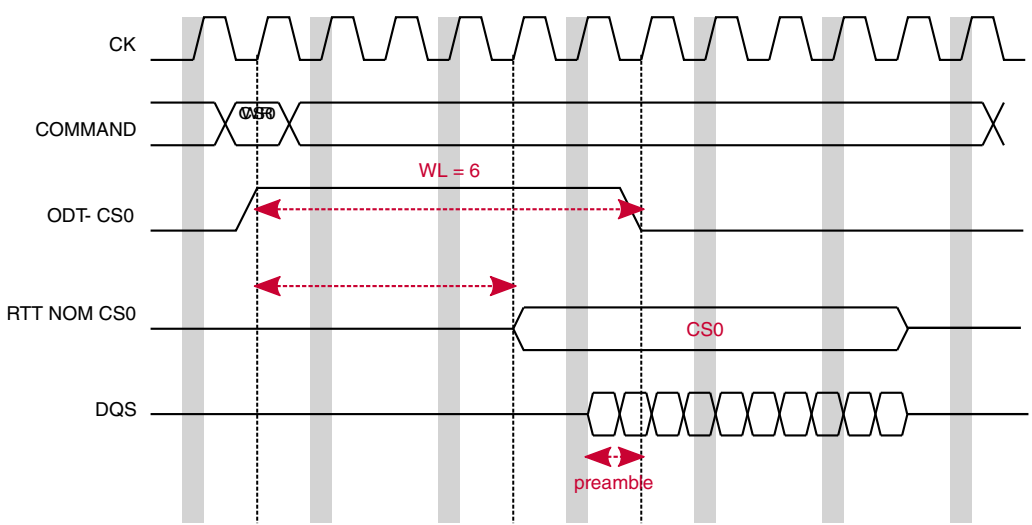


Figure 23-4. ODT - Timing Diagram DDR3L WL=6, BL=8

23.8 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 (
  - 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 LS1012A 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 DDR3L 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 DDR3L device MPR mode or write leveling calibration, the following should be done:

- Disable the periodic refresh scheme (i.e. setting MDREF[REF_SEL] = "11") 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").

23.8.1 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.

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

23.8.2 ZQ calibration

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

The first ZQ calibration (after booting the processor) is performed prior to turning on the MMDC. Subsequent 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 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 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].

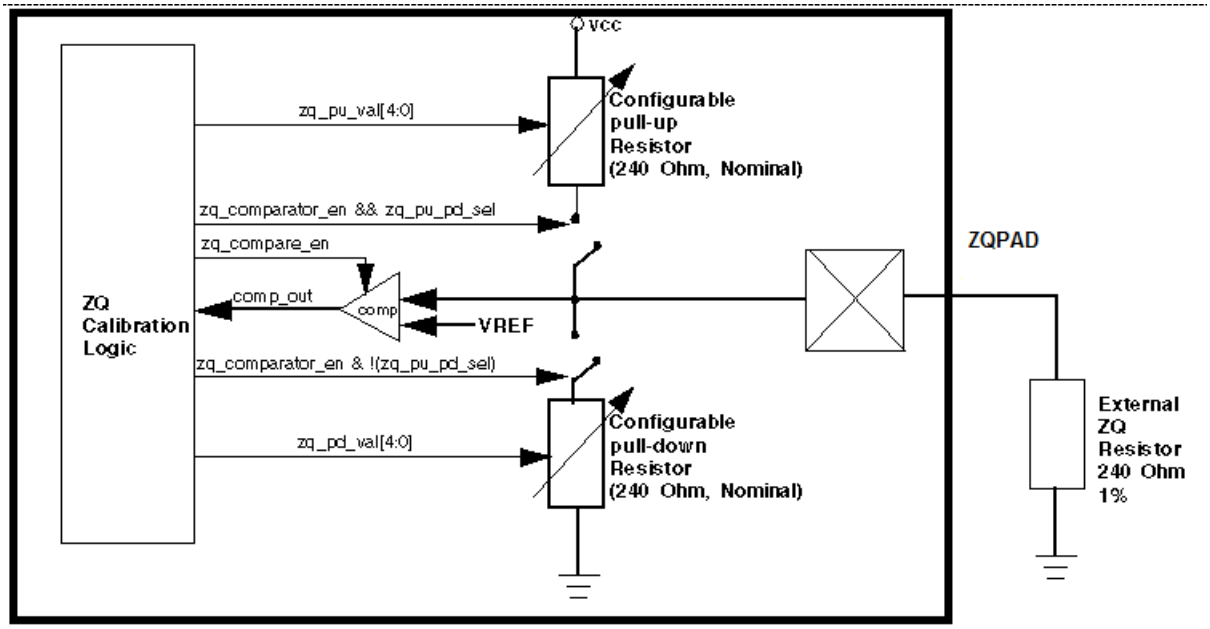


Figure 23-5. MMDC ZQ IF with PAD

23.8.2.1 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 process is as follows:

1. Assert the MPZQHWCTRL[ZQ_HW_FOR] bit.
2. Wait for ZQ_HW_FOR to be cleared by hardware.
3. The ZQ resistors are now calibrated. The pull-up and pull-down configured values can be read from the MPZQHWCTRL[ZQ_HW_PU_RES] and MPZQHWCTRL [ZQ_HW_PD_RES] bits.

NOTE

One time hardware ZQ calibration involves both the local MMDC PHY and the DDR device. ZQ_MODE should thus be set to one of the following modes:

- 0x1: ZQ calibration is issued to the chip's ZQ pad and external device (only when exiting self refresh).
- 0x3: ZQ calibration is issued to chip's ZQ pad and external device (both periodic and when exiting self refresh).

23.8.2.1.1 ZQ automatic Pull-up calibration

The MMDC automatically performs a handshaking mechanism with the 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

23.8.2.1.2 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

23.8.2.2 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.

The ZQ software calibration steps are as follows:

- Before starting the calibration, make sure that MMDC_MPZQHWCTRL[ZQ_MODE] bit is set to 00.

NOTE

Pull calibration steps must be performed first as it is the reference resistor during the PD calibration.

23.8.2.2.1 ZQ software Pull-up calibration

1. Set the MPZQSWCTRL[ZQ_SW_PD] bit to 0 (PU calibration).
2. Set the MPZQSWCTRL[ZQ_SW_PU_VAL] bit to a selected value.
3. Assert the MPZQSWCTRL[ZQ_SW_FOR] bit.
4. Wait for ZQ_SW_FOR to be cleared by hardware.
5. If ZQ_SW_RES == 1 is detected, decrease ZQ_SW_PU_VAL and repeat previous steps.
6. If ZQ_SW_RES == 0 is detected, stop the sequence and read ZQ_SW_PU_VAL.

23.8.2.2.2 ZQ software Pull-down calibration

1. Set the MPZQSWCTRL[ZQ_SW_PD] bit to 1 (PD calibration).
2. Set the MPZQSWCTRL[ZQ_SW_PD_VAL] bit to a selected value.
3. Assert the MPZQSWCTRL[ZQ_SW_FOR] bit.
4. Wait for ZQ_SW_FOR to be cleared by hardware.
5. If ZQ_SW_RES == 1 is detected, decrease ZQ_SW_PD_VAL and repeat previous steps.
6. If ZQ_SW_RES == 0 is detected, stop the sequence and read ZQ_SW_PD_VAL.

23.8.2.2.3 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).

23.8.3 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.

23.8.3.1 Hardware DQS Gating Calibration

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

23.8.3.1.1 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]

23.8.3.1.2 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.

    Execute the following steps manually to program the final delay:

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#] value and 3 MSB bits correspond to MPDGCTRL#[DG_HC_DEL#] value.

NOTE

Lower 10 bits are used in the final delay calculation, the 11^{th} bit is not used.

34. Set MPDGHWST#[HW_DG_UP#][6:0] (7 LSBs) to MPDGCTRL#[DG_DL_ABS_OFFSET#].
35. Set (MPDGHWST#[HW_DG_UP#][9:7] - 1) (3 MSBs) to MPDGCTRL#[DG_HC_DEL#]. (Set the DQS gating value to be the upper limit value minus 1 half cycle)

23.8.3.2 SW read DQS gating Calibration

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

23.8.3.2.1 SW read DQS gating 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

23.8.3.2.2 SW read DQS gating 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. Program the MDSCR register to all 0's.
3. Configure the pre-defined value, which reflects the value that will be written and compared through the read calicration, to MPPDCMPR1[PDV1, PDV2].
4. Issue write access to the external DDR device by setting MPSWDAR0[SW_DUMMY_WR] = 1.
5. 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 manually by the MMDC for both modes (MPR and Pre-defined value):

6. Configure the read DQS delay-line to issue zero delay by setting MPDGCTRL#[DG_DL_ABS_OFFSET#] = 0 and MPDGCTRL#[DG_HC_DEL#] = 0.
7. Force the delay line to measure itself and to issue the requested read delay by configuring MPMUR[FRC_MSR] = 1.
8. Wait 16 DDR cycles till the read DQS delay-line is updated with the absolute delay value for all bytes.
9. Issue read command to the external DDR device by setting MPSWDAR0[SW_DUMMY_RD] = 1.
10. Waits 16 or 32 cycles (according to MPDGCTRL0[DG_CMP_CYC]) assuming that the data has arrived from the DDR device.
11. 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.
12. MMDC resets the read FIFO (to the inverted pre-defined/MPR value) and it's pointers by setting MPDGCTRL[RST_RD_FIFO] = 1.
13. Increment the read DQS gating delay of each byte by half cycle (i.e. MPDGCTRL#[DG_HC_DEL#] + 1).
14. 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.
15. 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.
16. Store the temporary lower boundary and start searching the temporary upper boundary.
17. MMDC resets the read FIFO (to the inverted pre-defined/MPR value) and it's pointers by setting MPDGCTRL[RST_RD_FIFO] = 1.
18. Increment the read DQS gating delay of each byte by half cycle (i.e. MPDGCTRL#[DG_HC_DEL#] + 1).
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. 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).
21. 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.
22. Load the temporary low boundary minus half cycle into the associated MPDGCTRL#[DG_HC_DEL#].
23. Reset the read FIFO (to the inverted pre-defined/MPR value) and it's pointers by setting MPDGCTRL[RST_RD_FIFO] = 1.
24. 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.
25. 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.
26. 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).
27. 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.
28. Store the adequate lower boundary.
29. Load the temporary upper boundary minus half cycle into the associated MPDGCTRL#[DG_HC_DEL#].
30. Reset the read FIFO (to the inverted pre-defined/MPR value) and it's pointers by setting MPDGCTRL[RST_RD_FIFO] = 1.
31. 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.
32. 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.
33. 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).
34. If the comparison passes then it is needed to repeat steps 29-32. If the comparisons fails then advance to the next step.
35. Reset the read FIFO (to the inverted pre-defined/MPR value) and it's pointers by setting MPDGCTRL[RST_RD_FIFO] = 1.
36. Store the adequate upper boundary.
37. Keep the MPDGCTRL#[DG_DL_ABS_OFFSET#] value of the upper limit.
38. Set MPDGCTRL#[DG_HC_DEL#] = (MPDGCTRL#[DG_HC_DEL#] - 1). (Set the DQS gating value to be the upper limit value minus 1 half cycle).
39. Issue the requested read DQS delay by configuring MPMUR[FRC_MSR] = 1.
40. Exit the DDR device from MPR mode through MRS command.

NOTE

Even the lower boundary offset is found in above sequence, this would not be used in final programming.

23.8.4 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 DDR3L mode, the activation of the calibration is done by setting MPRDDLHWCTL[HW_RD_DL_EN].

23.8.4.1 Hardware (automatic) Read Calibration

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

23.8.4.1.1 Hardware (automatic) Calibration with MPR (DDR3L)

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/MRW 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 (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].

23.8.4.1.2 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

23.8.4.2 SW Read Calibration

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

23.8.4.2.1 Calibration with MPR(DDR3L)

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

23.8.4.2.2 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. Program MDSCR register to all 0's.
3. Configure the pre-defined value, which reflects the value that will be written and compared through the read calibration, to MPPDCMPR1[PDV1, PDV2].
4. Issue write access to the external DDR device by setting MPSWDAR0[SW_DUMMY_WR] to 1.
5. 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):

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 delay-line is updated with the absolute delay value for all bytes.
8. Issue read command to the external DDR device by setting MPSWDAR0[SW_DUMMY_RD] = 1.

9. 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.
10. Reset the rd fifo (to the inverted pre-defined/MPR value) and it's pointers by setting MPDGCTRL[RST_RD_FIFO] = 1.
11. Decrement the read delay line absolute offset of each byte by 1 (i.e. MPRDDLCTL[RD_DL_ABS_OFFSET#]).
12. Force the delay line to measure itself and to issue the requested read delay by configuring MPMUR[FRC_MSR] = 1.
13. Issue read command to the external DDR device by setting MPSWDAR0[SW_DUMMY_RD] = 1 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.
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 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.
15. 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.
16. Force the delay line to measure itself and to issue the requested read delay by configuring MPMUR[FRC_MSR] = 1.
17. Resets the rd fifo (to the inverted pre-defined/MPR value) and it's pointers by setting MPDGCTRL[RST_RD_FIFO] = 1
18. Issue read command to the external DDR device by setting MPSWDAR0[SW_DUMMY_RD] = 1.
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). 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#]).
20. Force the delay line to measure itself and to issue the requested read delay by configuring MPMUR[FRC_MSR] = 1.
21. If all read data comparisons fail then advance to the next step, else repeat steps 16-19.
22. 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#].
23. Force the delay line to measure itself and to issue the requested read delay by configuring MPMUR[FRC_MSR] = 1.

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

23.8.5 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.

23.8.5.1 HW (automatic) Write Calibration

The following steps should be executed:

1. Program MDSCR register to all 0's.
2. Clear the MMDC_MPPDCMPR2[MPR_CMP] bit, if already set.
3. 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.
4. Configure the pre-defined value, which reflects the value that will be written and compared through the write calibration, to MPPDCMPR1[PDV1, PDV2].
5. Set MPSWDAR0[SW_DUMMY_WR] to 1 to issue write access to the external DDR device (MMDC will generate internally write access without intervention of the system towards bank 0, row 0, column 0).
6. Wait for MPSWDAR0[SW_DUMMY_WR] = 0.
7. Assert MPWRDLHWCTL0[HW_WR_DL_EN].
8. MMDC indicates that the write data calibration had finished by setting MPWRDLHWCTL[HW_WR_DL_EN] to 0.

   The following steps will be executed automatically:

9. MMDC waits till the write delay-line is updated with the absolute delay value for all bytes at MPWRDCTL[WR_DL_ABS_OFFSET#]
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 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.

13. MMDC resets the rd fifo (to the inverted pre-defined value) and it's pointers by setting MPDGCTRL[RST_RD_FIFO] = 1
14. 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.
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 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
18. 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
19. MMDC resets the rd fifo (to the inverted pre-defined value) and its pointers by setting MPDGCTRL[RST_RD_FIFO] = 1
20. 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.
21. MMDC drives read command to the same address from the external DDR
22. 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.
23. MMDC repeats steps 14-17. If all data comparisons fail then the MMDC advances to the next step
24. .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.
25. MMDC indicates that the write data calibration had finished by setting MPWRDLHWCTL[HW_WR_DL_EN] = 0

23.8.5.2 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. Clear MPPDCMPR2[MPR_CMP] if already set.
2. Precharge all active banks (Can be done through MDSCR) as required by the standard.
3. Program MDSCR register to all 0's.
4. Configure the pre-defined value, which reflects the value that will be written and compared through the write calibration, to MPPDCMPR1[PDV1, PDV2].
5. 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.
6. Force the delay line to measure itself and to issue the requested write delay by configuring MPMUR[FRC_MSR] = 1.
7. Wait 16 DDR cycles till the write delay-line is updated with the absolute delay value for all bytes.
8. Issue write access to the external DDR device by setting MPSWDAR0[SW_DUMMY_WR] = 1.
9. Issue read command to the external DDR device by setting MPSWDAR0[SW_DUMMY_RD] = 1.
10. 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.
11. MMDC resets the rd fifo (to the inverted pre-defined value) and it's pointers by setting MPDGCTRL[RST_RD_FIFO] = 1.
12. Decrement the write delay line absolute offset of each byte by 1 (i.e. MPWRDLCTL[WR_DL_ABS_OFFSET#]).
13. Force the delay line to measure itself and to issue the requested write delay by configuring MPMUR[FRC_MSR] = 1.
14. Issue write access to the external DDR device by setting MPSWDAR0[SW_DUMMY_WR] = 1.
15. Issue read command to the external DDR device by setting MPSWDAR0[SW_DUMMY_RD] = 1.
16. 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.
17. Start seeking the upper boundary and set the write delay line absolute offset of each byte to the initial value + 1.
18. Force the delay line to measure itself and to issue the requested write delay by configuring MPMUR[FRC_MSR] = 1.
19. Reset the rd fifo (to the inverted pre-defined value) and it's pointers by setting MPDGCTRL[RST_RD_FIFO] = 1.
20. Issue write access to the external DDR device by setting MPSWDAR0[SW_DUMMY_WR] = 1.
21. Issue read command to the external DDR device by setting MPSWDAR0[SW_DUMMY_RD] = 1.
22. 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.
23. Force the delay line to measure itself and to issue the requested write delay by configuring MPMUR[FRC_MSR] = 1.
24. If all read data comparisons fail then advance to the next step else repeat steps 16-20.
25. 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#].
26. Force the delay line to measure itself and to issue the requested write delay by configuring MPMUR[FRC_MSR] = 1.

23.8.6 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 DDR3L 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 DDR3L 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 DDR3L memory to ensure less than 1 cycle skew between DQS and clock.

23.8.6.1 Hardware Write Leveling Calibration

The following steps should be executed:

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

   The following steps will be executed automatically by the MMDC:

3. 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
4. MMDC drives one DQS pulse to the DDR external device
5. 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])
6. 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
7. MMDC repeates steps 5-7 till the write leveling delay is 1 cycle
8. 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[WL_HW_ERR#]
9. 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
10. 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#]
11. MMDC perform measurement process in order to load the most accurate value to the associated delay-line
12. User should issue MRS command to exit write leveling mode
13. 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

14. Return the DQS output enable to functional mode by deasserting MDSCR[WL_EN]

23.8.6.2 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]

23.8.7 Write fine tuning

Write fine tuning is an additional circuit that provides the ability to fine tune the timing of each of the DQ/DM bits (relative to DQS) by up to 100 ps. To use the write fine tuning, select the number of delay units in each DQ/DM I/O (maximum 3 delay units of around 30-35 ps each). The delay can be configured independently for each DQ/DM. .

23.8.8 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.

23.8.9 ZQ Fine Tuning

An offset can be added to the PU /PD values determined by the ZQ calibration process. The offset range is programmable from -7 to +7, controlled by the MMDC_MPPDCMPR2[ZQ_PU_OFFSET] and MMDC_MPPDCMPR2[ZQ_PD_OFFSET] fields. The offset is enabled/disabled by MMDC_MPPDCMPR2[ZQ_OFFSET_EN]. See MMDC_MPPDCMPR2 register for more information.

23.8.10 Duty cycle adjustment

Duty cycle adjustments can be made to the SDCLKx and SDQSx signals, see the MMDCx_MPDCCR register for more information.

23.9 MMDC Memory Map/Register Definition

23.9.1 MMDC register descriptions

The Memory Map is shown below.

NOTE

The terms *clocks* and *cycles* are used interchangeably and refer to the clock period of the main ddr clock , commonly referred to as the DDR frequency.

23.9.1.1 MMDC Memory map

MMDC base address: 108_0000h

Offset	Register	Width (In bits)	Access	Reset value
0h	MMDC Core Control Register (MDCTL)	32	RW	0311_0000h
4h	MMDC Core Power Down Control Register (MDPDC)	32	RW	0003_0012h
8h	MMDC Core ODT Timing Control Register (MDOTC)	32	RW	1227_2000h
Ch	MMDC Core Timing Configuration Register 0 (MDCFG0)	32	RW	3236_22D3h

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
10h	MMDC Core Timing Configuration Register 1 (MDCFG1)	32	RW	B6B1_8A23h
14h	MMDC Core Timing Configuration Register 2 (MDCFG2)	32	RW	00C7_0092h
18h	MMDC Core Miscellaneous Register (MDMISC)	32	RW	0000_1600h
1Ch	MMDC Core Special Command Register (MDSCR)	32	RW	0000_0000h
20h	MMDC Core Refresh Control Register (MDREF)	32	RW	0000_C000h
2Ch	MMDC Core Read/Write Command Delay Register (MDRWD)	32	RW	0F9F_26D2h
30h	MMDC Core Out of Reset Delays Register (MDOR)	32	RW	009F_0E0Eh
400h	MMDC Core AXI Reordering Control Register (MAARCR)	32	RW	5142_01F0h
404h	MMDC Core Power Saving Control and Status Register (MAPSR)	32	RW	0000_1007h
408h	MMDC Core Exclusive ID Monitor Register0 (MAEXIDR0)	32	RW	0020_0000h
40Ch	MMDC Core Exclusive ID Monitor Register1 (MAEXIDR1)	32	RW	0060_0040h
410h	MMDC Core Debug and Profiling Control Register 0 (MADPCR0)	32	RW	0000_0000h
414h	MMDC Core Debug and Profiling Control Register 1 (MADPCR1)	32	RW	0000_0000h
418h	MMDC Core Debug and Profiling Status Register 0 (MADPSR0)	32	RO	0000_0000h
41Ch	MMDC Core Debug and Profiling Status Register 1 (MADPSR1)	32	RO	0000_0000h
420h	MMDC Core Debug and Profiling Status Register 2 (MADPSR2)	32	RO	0000_0000h
424h	MMDC Core Debug and Profiling Status Register 3 (MADPSR3)	32	RO	0000_0000h
428h	MMDC Core Debug and Profiling Status Register 4 (MADPSR4)	32	RO	0000_0000h
42Ch	MMDC Core Debug and Profiling Status Register 5 (MADPSR5)	32	RO	0000_0000h
430h	MMDC Core Step By Step Address Register (MASBS0)	32	RO	0000_0000h
434h	MMDC Core Step By Step Address Attributes Register (MASBS1)	32	RO	0000_0000h
440h	MMDC Core General Purpose Register (MAGENP)	32	RW	0000_0000h
800h	MMDC PHY ZQ HW control register (MPZQHWCTRL)	32	RW	A138_0000h
804h	MMDC PHY ZQ SW control register (MPZQSWCTRL)	32	RW	0000_0000h
808h	MMDC PHY Write Leveling Configuration and Error Status Register (MPWLGCR)	32	RW	0000_0000h
80Ch	MMDC PHY Write Leveling Delay Control Register 0 (MPWLDECTRL0)	32	RW	0000_0000h
814h	MMDC PHY Write Leveling delay-line Status Register (MPWLDLST)	32	RO	0000_0000h
818h	MMDC PHY ODT control register (MPODTCTRL)	32	RW	0000_0000h
81Ch	MMDC PHY Read DQ Byte0 Delay Register (MPRDDQBY0DL)	32	RW	0000_0000h
820h	MMDC PHY Read DQ Byte1 Delay Register (MPRDDQBY1DL)	32	RW	0000_0000h
82Ch	MMDC PHY Write DQ Byte0 Delay Register (MPWRDQBY0DL)	32	RW	0000_0000h
830h	MMDC PHY Write DQ Byte1 Delay Register (MPWRDQBY1DL)	32	RW	0000_0000h
83Ch	MMDC PHY Read DQS Gating Control Register 0 (MPDGCTRL0)	32	RW	0000_0000h
844h	MMDC PHY Read DQS Gating delay-line Status Register (MPDGDLST0)	32	RO	0000_0000h
848h	MMDC PHY Read delay-lines Configuration Register (MPRDDLCTL)	32	RW	4040_4040h
84Ch	MMDC PHY Read delay-lines Status Register (MPRDDLST)	32	RO	0000_0000h
850h	MMDC PHY Write delay-lines Configuration Register (MPWRDLCTL)	32	RW	4040_4040h

Table continues on the next page...

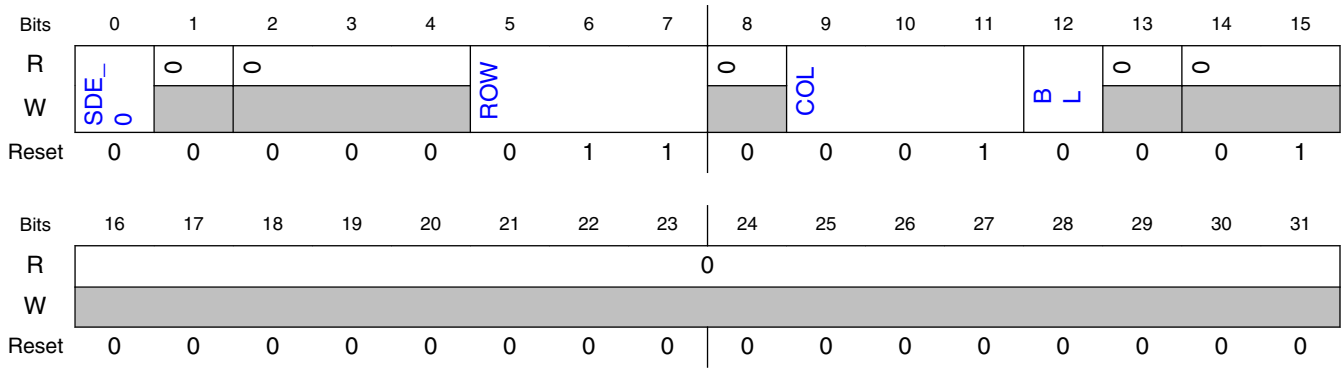
Offset	Register	Width (In bits)	Access	Reset value
854h	MMDC PHY Write delay-lines Status Register (MPWRDLST)	32	RO	0000_0000h
858h	MMDC PHY CK Control Register (MPSDCTRL)	32	RW	0000_0000h
860h	MMDC PHY Read Delay HW Calibration Control Register (MPRD DLHWCTL)	32	RW	0000_0000h
864h	MMDC PHY Write Delay HW Calibration Control Register (MPWR DLHWCTL)	32	RW	0000_0000h
868h	MMDC PHY Read Delay HW Calibration Status Register 0 (MPRD DLHWST0)	32	RO	0000_0000h
870h	MMDC PHY Write Delay HW Calibration Status Register 0 (MPWR DLHWST0)	32	RO	0000_0000h
878h	MMDC PHY Write Leveling HW Error Register (MPWLHWERR)	32	RO	0000_0000h
87Ch	MMDC PHY Read DQS Gating HW Status Register 0 (MPDGHWST 0)	32	RO	0000_0000h
880h	MMDC PHY Read DQS Gating HW Status Register 1 (MPDGHWST 1)	32	RO	0000_0000h
88Ch	MMDC PHY Pre-defined Compare Register 1 (MPPDCMPR1)	32	RW	0000_0000h
890h	MMDC PHY Pre-defined Compare and CA delay-line Configuration Register (MPPDCMPR2)	32	RW	0040_0000h
894h	MMDC PHY SW Dummy Access Register (MPSWDAR0)	32	RW	0000_0000h
898h	MMDC PHY SW Dummy Read Data Register 0 (MPSWDRDR0)	32	RO	FFFF_FFFFh
89Ch	MMDC PHY SW Dummy Read Data Register 1 (MPSWDRDR1)	32	RO	FFFF_FFFFh
8A0h	MMDC PHY SW Dummy Read Data Register 2 (MPSWDRDR2)	32	RO	FFFF_FFFFh
8A4h	MMDC PHY SW Dummy Read Data Register 3 (MPSWDRDR3)	32	RO	FFFF_FFFFh
8A8h	MMDC PHY SW Dummy Read Data Register 4 (MPSWDRDR4)	32	RO	FFFF_FFFFh
8ACh	MMDC PHY SW Dummy Read Data Register 5 (MPSWDRDR5)	32	RO	FFFF_FFFFh
8B0h	MMDC PHY SW Dummy Read Data Register 6 (MPSWDRDR6)	32	RO	FFFF_FFFFh
8B4h	MMDC PHY SW Dummy Read Data Register 7 (MPSWDRDR7)	32	RO	FFFF_FFFFh
8B8h	MMDC PHY Measure Unit Register (MPMUR0)	32	RW	0000_0000h
8C0h	MMDC Duty Cycle Control Register (MPDCCR)	32	RW	2492_2492h

23.9.1.2 MMDC Core Control Register (MDCTL)

23.9.1.2.1 Offset

Register	Offset
MDCTL	0h

23.9.1.2.2 Diagram



23.9.1.2.3 Fields

Field	Function
0 SDE_0	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. 0b - Disabled 1b - Enabled
1 —	Reserved.
2-4 —	- Reserved
5-7 ROW	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 000b - 11 bits Row 001b - 12 bits Row 010b - 13 bits Row 011b - 14 bits Row 100b - 15 bits Row 101b - 16 bits Row
8 —	- Reserved
9-11 COL	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. 001b - 10 bits column 010b - 11 bits column

Table continues on the next page...

Field	Function
	011-111b - Reserved
12 BL	BL Burst Length. This field determines the burst length of the DDR device. In DDR3L mode the MMDC supports burst length 8. 0b - Burst Length 4 is used 1b - Burst Length 8 is used
13 —	- Reserved
14-15 —	Reserved.
16-31 —	- Reserved

23.9.1.3 MMDC Core Power Down Control Register (MDPDC)

23.9.1.3.1 Offset

Register	Offset
MDPDC	4h

23.9.1.3.2 Function

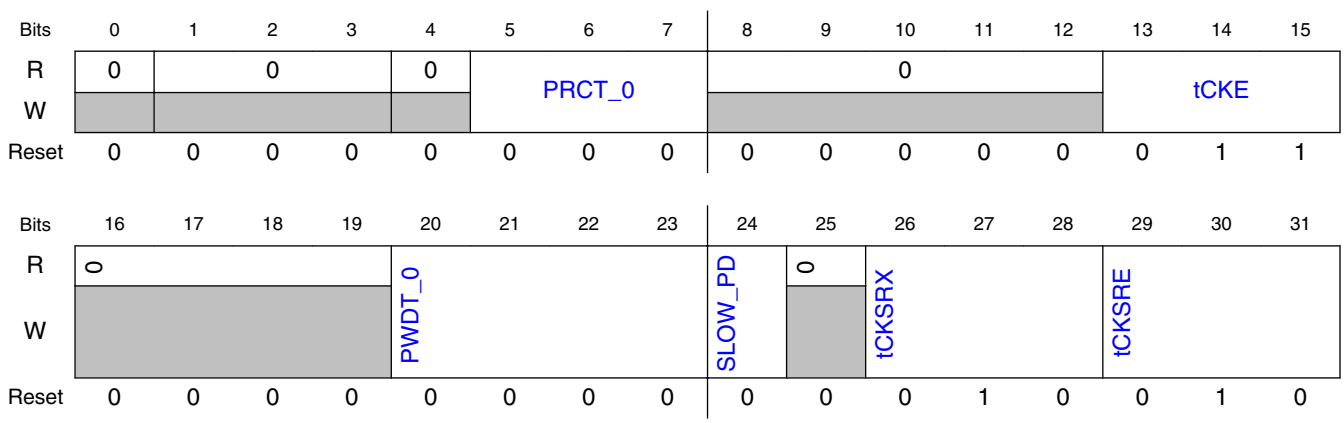
Table 23-8. PRCT field encoding

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 23-9. 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

23.9.1.3.3 Diagram



23.9.1.3.4 Fields

Field	Function
0 —	- Reserved
1-3 —	Reserved.
4 —	- Reserved

Table continues on the next page...

Field	Function
5-7 PRCT_0	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 above.
8-12 —	- Reserved
13-15 tCKE	tCKE CKE minimum pulse width. This field determines the minimum pulse width of CKE. 000b - 1 cycle 001b - 2 cycles 110b - 7 cycles 111b - 8 cycles
16-19 —	Reserved.
20-23 PWDT_0	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.
24 SLOW_PD	SLOW_PD Slow/fast power down. In DDR3L mode this field is referred to slow precharge power-down. **NOTE:** Memory should be configured the same. 0b - Fast mode. 1b - Slow mode.
25 —	Reserved.
26-28 tCKSRX	tCKSRX Valid clock cycles before self-refresh exit. This field determines the amount of clock cycles before self-refresh exit 000b - 0 cycle 001b - 1 cycles 110b - 6 cycles 111b - 7 cycles
29-31 tCKSRE	tCKSRE Valid clock cycles after self-refresh entry. This field determines the amount of clock cycles after self-refresh entry 000b - 0 cycle 001b - 1 cycles 110b - 6cycles 111b - 7cycles

23.9.1.4 MMDC Core ODT Timing Control Register (MDOTC)

23.9.1.4.1 Offset

Register	Offset
MDOTC	8h

23.9.1.4.2 Function

For further information see ODT Configuration .

23.9.1.4.3 Diagram

Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0		tAOFPD			tAONPD			tANPD				tAXPD			
W																
Reset	0	0	0	1	0	0	1	0	0	0	1	0	0	1	1	1

Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
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

23.9.1.4.4 Fields

Field	Function
0-1 —	- Reserved
2-4 tAOFPD	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. 000b - 1 cycle 001b - 2 cycles 110b - 7 cycles 111b - 8 cycles
5-7 tAONPD	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. 000b - 1 cycle 001b - 2 cycles

Table continues on the next page...

Field	Function
	110b - 7 cycles 111b - 8 cycles
8-11 tANPD	tANPD Asynchronous ODT to power down entry delay. In DDR3L should be set to tCWL-1 0000b - 1 clock 0001b - 2 clocks 0010b - 3 clocks 1110b - 15 clocks 1111b - 16 clocks
12-15 tAXPD	tAXPD Asynchronous ODT to power down exit delay. In DDR3L should be set to tCWL-1 0000b - 1 clock 0001b - 2 clocks 0010b - 3 clocks 1110b - 15 clocks 1111b - 16 clocks
16 —	- Reserved
17-19 tODTLon	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] 000b - - 0x1 Reserved 010b - 2 cycles 011b - 3 cycles 100b - 4 cycles 101b - 5 cycles 110b - 6 cycles 111b - Reserved
20-22 —	- Reserved
23-27 tODT_idle_off	tODT_idle_off ODT turn off latency. This field determines the Idle period before turning memory ODT off. 00000b - 0 cycle (turned off at the earliest possible time) 00001b - 1 cycle 00010b - 2 cycles 11110b - 30 cycles 11111b - 31 cycles
28-31 —	- Reserved

23.9.1.5 MMDC Core Timing Configuration Register 0 (MDCFG0)

23.9.1.5.1 Offset

Register	Offset
MDCFG0	Ch

23.9.1.5.2 Diagram

Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	tRFC								tXS							
Reset	0	0	1	1	0	0	1	0	0	0	1	1	0	1	1	0

Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	tXP			tXPDLL					tFAW				tCL			
Reset	0	0	1	0	0	0	1	0	1	1	0	1	0	0	1	1

23.9.1.5.3 Fields

Field	Function
0-7 tRFC	tRFC Refresh command to Active or Refresh command time. See DDR3 SDRAM Specification JESD79-3E (July 2010) for a detailed description of this parameter. 00000000b - 1 clock 00000001b - 2 clocks 00000010b - 3 clocks 11111110b - 255 clocks 11111111b - 256 clocks
8-15 tXS	tXS Exit self refresh to non READ command. See DDR3 SDRAM Specification JESD79-3E (July 2010) for a detailed description of this parameter. 00000000b - - 0x15 reserved 00010110b - 23 clocks 00010111b - 24 clocks 11111110b - 255 clocks 11111111b - 256 clocks
16-18 tXP	tXP Exit power down with DLL-on to any valid command. Exit power down with DLL-frozen to commands not requiring a locked DLL See DDR3 SDRAM Specification JESD79-3E (July 2010) for a detailed description of this parameter. 000b - 1 cycle 001b - 2 cycles

Table continues on the next page...

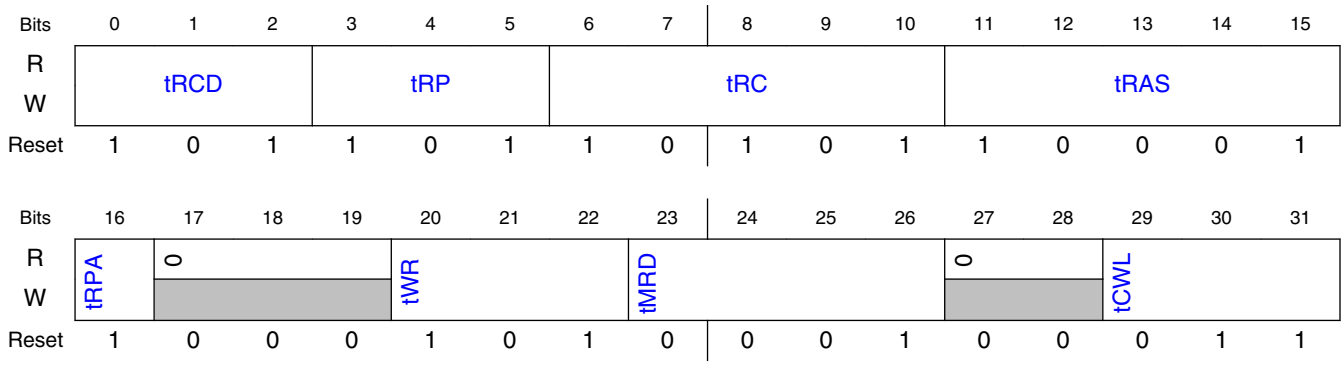
Field	Function
	110b - 7 cycles 111b - 8 cycles
19-22 tXPDLL	tXPDLL Exit precharge power down with DLL frozen to commands requiring DLL. See DDR3 SDRAM Specification JESD79-3E (July 2010) for a detailed description of this parameter. 0000b - 1 clock 0001b - 2 clocks 0010b - 3 clocks 1110b - 15 clocks 1111b - 16 clocks
23-27 tFAW	tFAW Four Active Window (all banks). See DDR3 SDRAM Specification JESD79-3E (July 2010) for a detailed description of this parameter. 00000b - 1 clock 00001b - 2 clocks 00010b - 3 clocks 11110b - 31 clocks 11111b - 32 clocks
28-31 tCL	tCL CAS Read Latency. In DDR3L mode this field is referred to CL. See DDR3 SDRAM Specification JESD79-3E (July 2010) for a detailed description of this parameter. 0000b - 3 cycles 0001b - 4 cycles 0010b - 5 cycles 0011b - 6 cycles 0100b - 7 cycles 0101b - 8 cycles 0110b - 9 cycles 0111b - 10 cycles 1000b - 11 cycles 1001b - - 0xF Reserved

23.9.1.6 MMDC Core Timing Configuration Register 1 (MDCFG1)

23.9.1.6.1 Offset

Register	Offset
MDCFG1	10h

23.9.1.6.2 Diagram



23.9.1.6.3 Fields

Field	Function
0-2 tRCD	tRCD Active command to internal read or write delay time (same bank). This field is valid only for DDR3L memories See DDR3 SDRAM Specification JESD79-3E (July 2010) for a detailed description of this parameter. 000b - 1 clock 001b - 2 clocks 010b - 3 clocks 011b - 4 clocks 100b - 5 clocks 101b - 6 clocks 110b - 7 clocks 111b - 8 clocks
3-5 tRP	tRP Precharge command period (same bank). This field is valid only for DDR3L memories See DDR3 SDRAM Specification JESD79-3E (July 2010) for a detailed description of this parameter. 000b - 1 clock 001b - 2 clocks 010b - 3 clocks 011b - 4 clocks 100b - 5 clocks 101b - 6 clocks 110b - 7 clocks 111b - 8 clocks
6-10 tRC	tRC Active to Active or Refresh command period (same bank). This field is valid only for DDR3L memories See DDR3 SDRAM Specification JESD79-3E (July 2010) for a detailed description of this parameter. 00000b - 1 clock 00001b - 2 clocks

Table continues on the next page...

Field	Function
	00010b - 3 clocks 11110b - 31 clocks 11111b - 32 clocks
11-15 tRAS	tRAS Active to Precharge command period (same bank). See DDR3 SDRAM Specification JESD79-3E (July 2010) for a detailed description of this parameter. 00000b - 1 clock 00001b - 2 clocks 00010b - 3 clocks 11110b - 31 clocks 11111b - Reserved
16 tRPA	tRPA Precharge-all command period. This field is valid only for DDR3L memories See DDR3 SDRAM Specification JESD79-3E (July 2010) for a detailed description of this parameter. 0b - Will be equal to: tRP. 1b - Will be equal to: tRP+1.
17-19 —	- Reserved
20-22 tWR	tWR WRITE recovery time (same bank). See DDR3 SDRAM Specification JESD79-3E (July 2010) for a detailed description of this parameter. 000b - 1cycle 001b - 2cycles 010b - 3cycles 011b - 4cycles 100b - 5cycles 101b - 6cycles 110b - 7cycles 111b - 8 cycles
23-26 tMRD	tMRD Mode Register Set command cycle (all banks). In DDR3L mode this field shoud be set to max (tMRD,tMOD). See DDR3 SDRAM Specification JESD79-3E (July 2010) for a detailed description of this parameter. 0000b - 1 clock 0001b - 2 clocks 0010b - 3 clocks 1110b - 15 clocks 1111b - 16 clocks
27-28 —	- Reserved
29-31 tCWL	tCWL CAS Write Latency. In DDR3L mode this field is referred to CWL. 000b - 2 cycles (DDR3L)

Field	Function
	001b - 3 cycles (DDR3L) 010b - 4 cycles (DDR3L) 011b - 5 cycles (DDR3L) 100b - 6 cycles (DDR3L) 101b - 7 cycles (DDR3L) 110b - 8 cycles (DDR3L) 111b - Reserved

23.9.1.7 MMDC Core Timing Configuration Register 2 (MDCFG2)

23.9.1.7.1 Offset

Register	Offset
MDCFG2	14h

23.9.1.7.2 Diagram

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

Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0								tRTP		tWTR			tRRD		
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	1	0	0	1	0

23.9.1.7.3 Fields

Field	Function
0-6 —	- Reserved
7-15 tDLLK	tDLLK DLL locking time. See DDR3 SDRAM Specification JESD79-3E (July 2010) for a detailed description of this parameter. 000000000b - 1 cycle. 000000001b - 2 cycles.

Table continues on the next page...

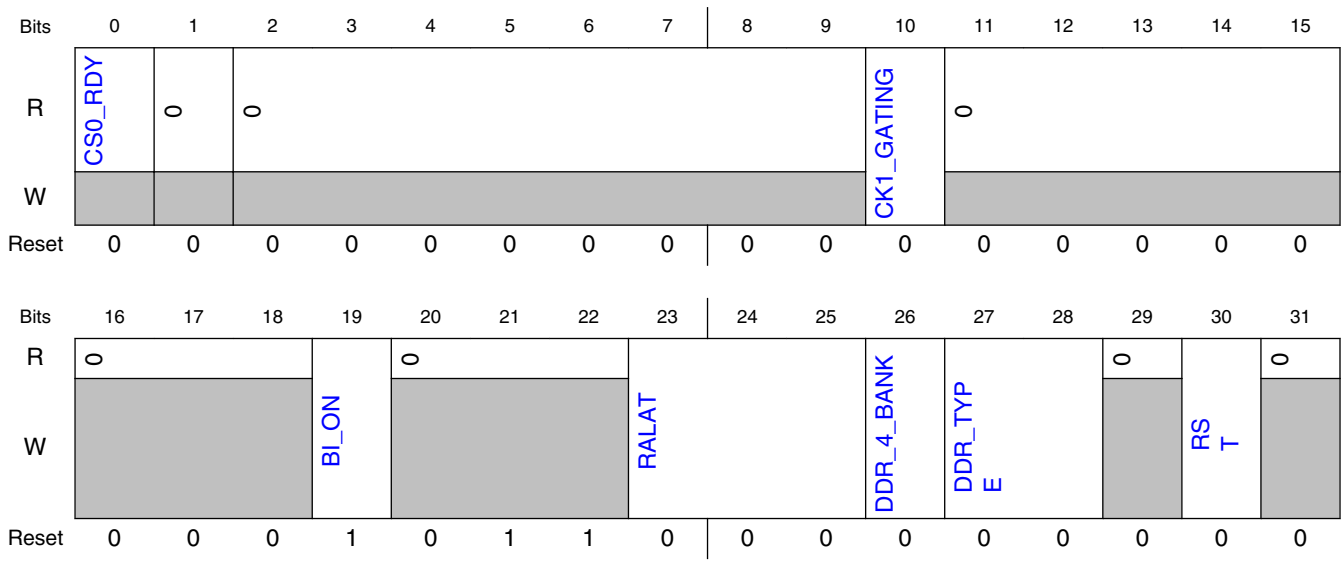
Field	Function
	000000010b - 3 cycles. 011000111b - 200 cycles 111111110b - 511 cycles. 111111111b - 512 cycles (JEDEC value for DDR3L).
16-22 —	- Reserved
23-25 tRTP	tRTP Internal READ command to Precharge command delay (same bank). See DDR3 SDRAM Specification JESD79-3E (July 2010) for a detailed description of this parameter. 000b - 1cycle 001b - 2cycles 010b - 3cycles 011b - 4cycles 100b - 5cycles 101b - 6cycles 110b - 7cycles 111b - 8 cycles
26-28 tWTR	tWTR Internal WRITE to READ command delay (same bank). See DDR3 SDRAM Specification JESD79-3E (July 2010) for a detailed description of this parameter. 000b - 1cycle 001b - 2cycles 010b - 3cycles 011b - 4cycles 100b - 5cycles 101b - 6cycles 110b - 7cycles 111b - 8 cycles
29-31 tRRD	tRRD Active to Active command period (all banks). See DDR3 SDRAM Specification JESD79-3E (July 2010) for a detailed description of this parameter. 000b - 1cycle 001b - 2cycles 010b - 3cycles 011b - 4cycles 100b - 5cycles 101b - 6cycles 110b - 7cycles 111b - Reserved

23.9.1.8 MMDC Core Miscellaneous Register (MDMISC)

23.9.1.8.1 Offset

Register	Offset
MDMISC	18h

23.9.1.8.2 Diagram



23.9.1.8.3 Fields

Field	Function
0 CS0_RDY	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. 0b - Device in wake-up period. 1b - Device is ready for initialization.
1 —	Reserved.
2-9 —	- Reserved
10 CK1_GATING	CK1_GATING Gating the secondary DDR clock. When this bit is asserted then the MMDC will disable the secondary DDR clock 0b - MMDC drives two clocks toward the DDR memory 1b - MMDC drives only one clock toward the DDR memory (CK0)
11-15	Reserved.

Table continues on the next page...

Field	Function
—	
16-18 —	- Reserved
19 BI_ON	BI_ON Bank Interleaving On. This bit controls the organization of the bank, row and column address bits. For further information see Address decoding . 0b - Banks are not interleaved, and address will be decoded as bank-row-column 1b - Banks are interleaved, and address will be decoded as row-bank-column
20-22 —	Reserved.
23-25 RALAT	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. 000b - no additional latency. 001b - 1 cycle additional latency. 010b - 2 cycles additional latency. 011b - 3 cycles additional latency. 100b - 4 cycles additional latency. 101b - 5 cycles additional latency. 110b - 6 cycles additional latency. 111b - 7 cycles additional latency.
26 DDR_4_BANK	DDR_4_BANK Number of banks per DDR device. 0b - 8 banks device is being used. (Default)
27-28 DDR_TYPE	DDR_TYPE DDR TYPE. This field determines the type of the external DDR device. 00b - DDR3L device is used. 01b - Reserved 10b - Reserved 11b - Reserved
29 —	- Reserved
30 RST	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. 0b - Do nothing. 1b - Assert reset to the MMDC.
31 —	- Reserved

23.9.1.9 MMDC Core Special Command Register (MDSCR)

23.9.1.9.1 Offset

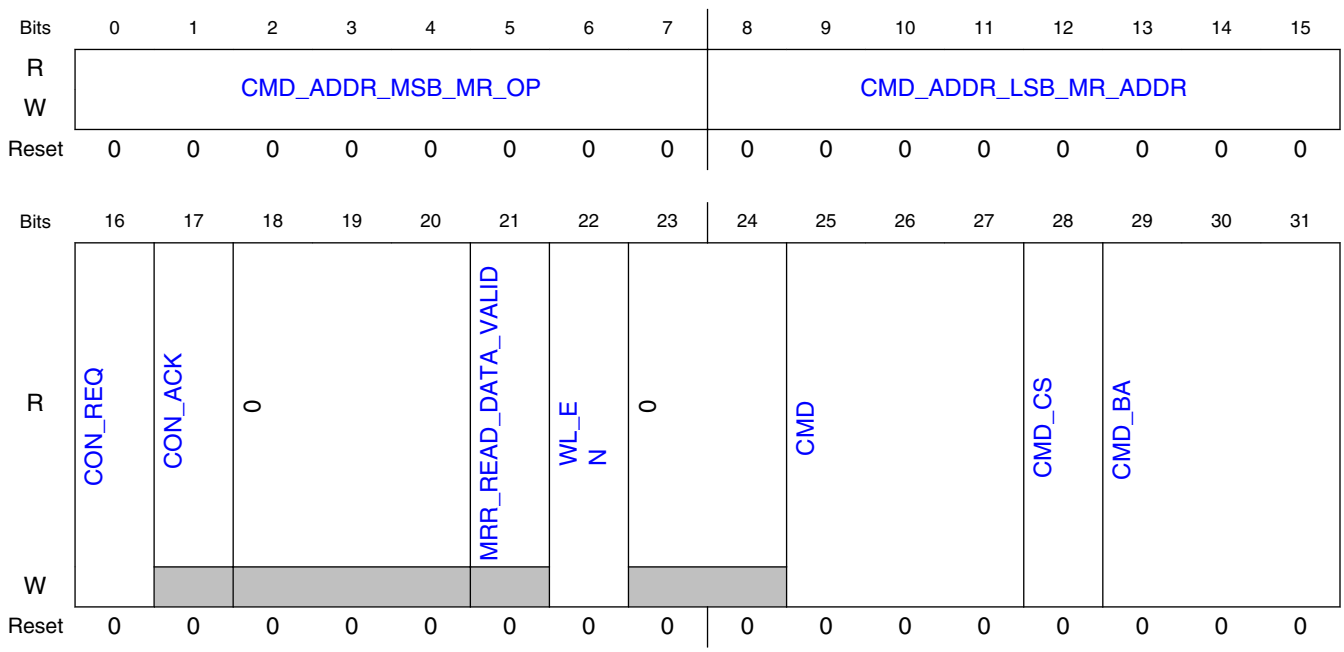
Register	Offset
MDSCR	1Ch

23.9.1.9.2 Function

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.

23.9.1.9.3 Diagram



23.9.1.9.4 Fields

Field	Function
0-7	CMD_ADDR_MSB_MR_OP

Table continues on the next page...

Field	Function
CMD_ADDR_MSB_MR_OP	Command/Address MSB. This field indicates the MSB of the command/Address.
8-15 CMD_ADDR_LSB_MR_ADDR	CMD_ADDR_LSB_MR_ADDR Command/Address LSB. This field indicates the LSB of the command/Address
16 CON_REQ	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] 0b - No request to configure MMDC. 1b - A request to configure MMDC is valid
17 CON_ACK	CON_ACK Configuration acknowledge. Whenever this bit is set, it is permitted to configure MMDC IP registers. 0b - Configuration of MMDC registers is forbidden. 1b - Configuration of MMDC registers is permitted.
18-20 —	- Reserved
21 MRR_READ_DATA_VALID	MRR_READ_DATA_VALID MRR read data valid. This field indicates that read data is valid at MDMRR register 0b - Cleared upon the assertion of MRR command 1b - Set after MRR data is valid and stored at MDMRR register.
22 WL_EN	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. 0b - Exit write leveling mode or stay in normal mode. 1b - Write leveling entry command was sent.
23-24 —	- Reserved.
25-27 CMD	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. **NOTE:** Only the 0x5 Precharge all command should be used, memory operation is unpredictable when using the 0x1 command 000b - Normal operation

Table continues on the next page...

Field	Function
	001b - Reserved 010b - Auto-Refresh Command (set correct CMD_CS). 011b - Load Mode Register Command DDR3L, set correct CMD_CS, CMD_BA, CMD_ADDR_LSB, CMD_ADDR_MSB), MRW Command 100b - ZQ calibration (DDR3L, set correct CMD_CS, {CMD_ADDR_MSB,CMD_ADDR_LSB} = 0x400 or 0x0) 101b - Precharge all, only if banks open (set correct CMD_CS). 110b - MRR command 111b - Reserved
28 CMD_CS	CMD_CS Chip Select. This field determines which chip select the command is targeted to: 0b - Chip-select 0 1b - Reserved
29-31 CMD_BA	CMD_BA Bank Address. This field determines the address of the bank within the selected chip-select where the command is targetted to. 000b - bank address 0 001b - bank address 1 010b - bank address 2 111b - bank address 7

23.9.1.10 MMDC Core Refresh Control Register (MDREF)

23.9.1.10.1 Offset

Register	Offset
MDREF	20h

23.9.1.10.2 Function

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 23-10. 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

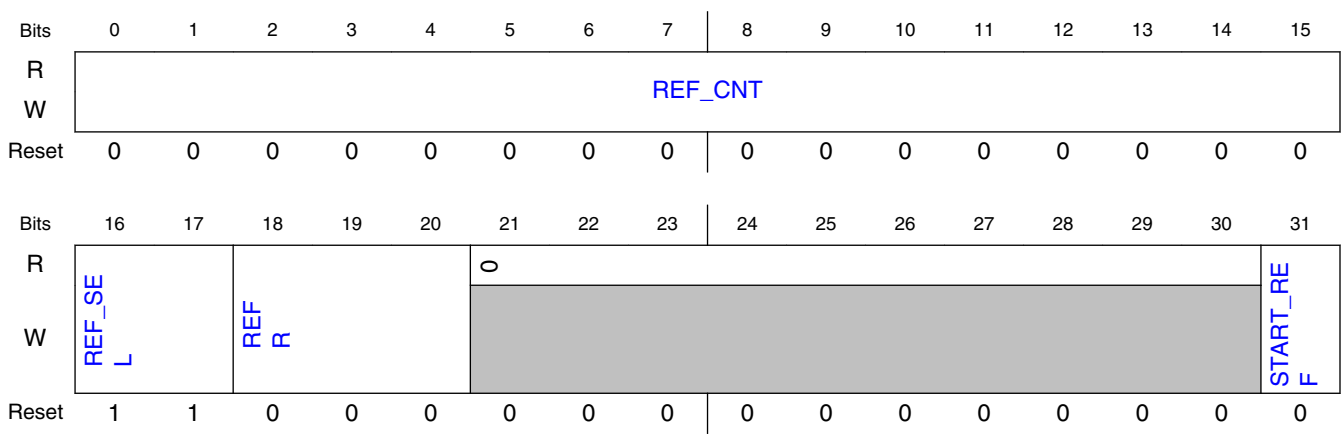
Table 23-11. Refresh rate example for REF_SEL = 2@ 400MHz

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.

23.9.1.10.3 Diagram



23.9.1.10.4 Fields

Field	Function
0-15 REF_CNT	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. 0000000000000000b - Reserved. 0000000000000001b - 1 cycle. 1111111111111110b - 65534 cycles. 1111111111111111b - 65535 cycles.
16-17 REF_SEL	REF_SEL Refresh Selector. This bit selects the source of the clock that will trigger each refresh cycle: 00b - Periodic refresh cycles will be triggered in frequency of 64KHz. 01b - Periodic refresh cycles will be triggered in frequency of 32KHz. 10b - Periodic refresh cycles will be triggered every amount of cycles that are configured in REF_CNT field. 11b - No refresh cycles will be triggered.
18-20 REFR	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 000b - 1 refresh 001b - 2 refreshes 010b - 3 refreshes 011b - 4 refreshes 100b - 5 refreshes 101b - 6 refreshes 110b - 7 refreshes 111b - 8 refreshes
21-30 —	- Reserved
31 START_REF	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. 0b - Do nothing. 1b - Start a refresh cycle.

23.9.1.11 MMDC Core Read/Write Command Delay Register (MDRWD)

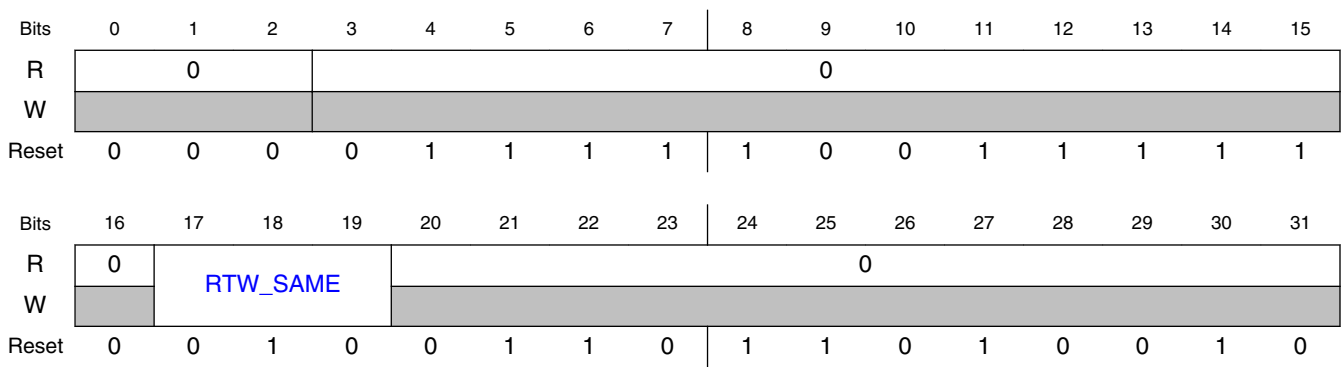
23.9.1.11.1 Offset

Register	Offset
MDRWD	2Ch

23.9.1.11.2 Function

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.

23.9.1.11.3 Diagram



23.9.1.11.4 Fields

Field	Function
0-2 —	- Reserved
3-15 —	Reserved.
16 —	- Reserved
17-19 RTW_SAME	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 000b - 0 cycle 001b - 1 cycle 010b - 2 cycles (Default) 011b - 3 cycles

Table continues on the next page...

Field	Function
	100b - 4 cycles 101b - 5 cycles 110b - 6 cycles 111b - 7 cycles
20-31 —	Reserved.

23.9.1.12 MMDC Core Out of Reset Delays Register (MDOR)

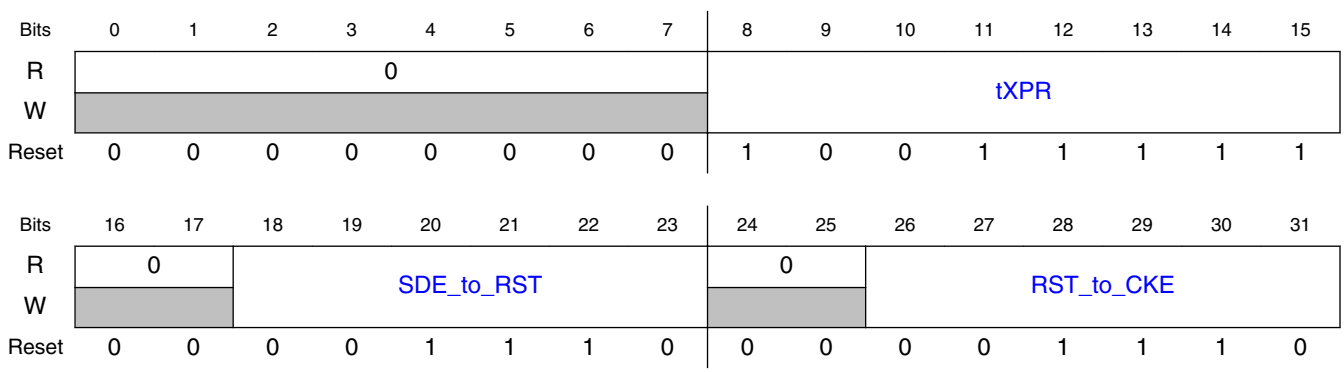
23.9.1.12.1 Offset

Register	Offset
MDOR	30h

23.9.1.12.2 Function

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

23.9.1.12.3 Diagram



23.9.1.12.4 Fields

Field	Function
0-7 —	- Reserved
8-15	tXPR

Table continues on the next page...

Field	Function
tXPR	DDR3L: CKE HIGH to a valid command. DDR3L: As defined in timing parameter table. 00000000b - Reserved 00000001b - 2 cycles 00000010b - 3 cycles 11111110b - 255 cycles 11111111b - 256 cycles
16-17 —	- Reserved
18-23 SDE_to_RST	SDE_to_RST DDR3L: Time from SDE enable until DDR reset is high. **NOTE:** Each cycle in this field is 15.258 µs. 000000b - Reserved 000001b - Reserved 000010b - Reserved 000011b - 1 cycles 000100b - 2 cycles 010000b - 14 cycles (JEDEC value for DDR3L) - total of 200 µs 111110b - 60 cycles 111111b - 61 cycles
24-25 —	- Reserved
26-31 RST_to_CKE	RST_to_CKE DDR3L: Time from SDE enable to CKE rise. In case that DDR reset is low, will wait until it's high and then wait this period until rising CKE. (JEDEC value is 500 µs) **NOTE:** Each cycle in this field is 15.258 µs. 000000b - Reserved 000001b - Reserved 000010b - Reserved 000011b - 1 cycles 010000b - 14 cycles 100011b - 33 cycles (JEDEC value for DDR3L) - total of 500 µs 111110b - 60 cycles 111111b - 61 cycles

23.9.1.13 MMDC Core AXI Reordering Control Register (MAARCR)

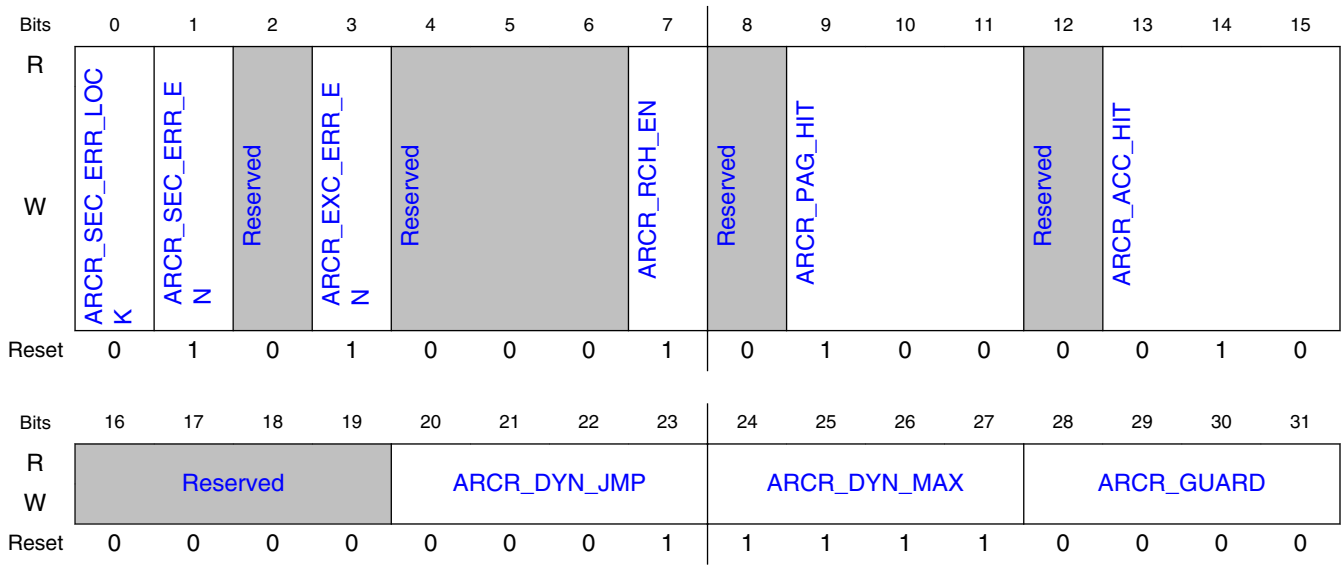
23.9.1.13.1 Offset

Register	Offset
MAARCR	400h

23.9.1.13.2 Function

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

23.9.1.13.3 Diagram



23.9.1.13.4 Fields

Field	Function
0 ARCR_SEC_ERR_LOCK	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) 0b - ARCR_SEC_ERR_EN is unlocked, so can be updated any moment 1b - ARCR_SEC_ERR_EN is locked, so it can't be updated
1 ARCR_SEC_ERR_EN	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) 0b - security violation results in OKAY response (rresp/bresp=2'b00) 1b - security violation results in SLAVE Error response (rresp/bresp=2'b10)
2 —	- Reserved
3 ARCR_EXC_ERR_EN	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

Table continues on the next page...

Field	Function
	Default value is 0x1 - encoding 1(response is SLV Error) 0b - violation of AXI exclusive rules (6.2.4) result in OKAY response (rresp/bresp=2'b00) 1b - violation of AXI exclusive rules (6.2.4) result in SLAVE Error response (rresp/bresp=2'b10)
4-6 —	- Reserved
7 ARCR_RCH_EN	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 priority in the optimization/reordering mechanism Default value is 0x1 - encoding 1 (Enabled) 0b - normal prioritization, no bypassing 1b - accesses with QoS=='F' bypass the arbitration
8 —	- Reserved
9-11 ARCR_PAG_HIT	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.
12 —	- Reserved
13-15 ARCR_ACC_HIT	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.
16-19 —	- Reserved
20-23 ARCR_DYN_JMP	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
24-27 ARCR_DYN_MAX	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. 0000b - 0 0001b - 1 1111b - 15 (default)
28-31 ARCR_GUARD	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. 0000b - 15 (default) 0001b - 16 1111b - 30

23.9.1.14 MMDC Core Power Saving Control and Status Register (MAPSR)

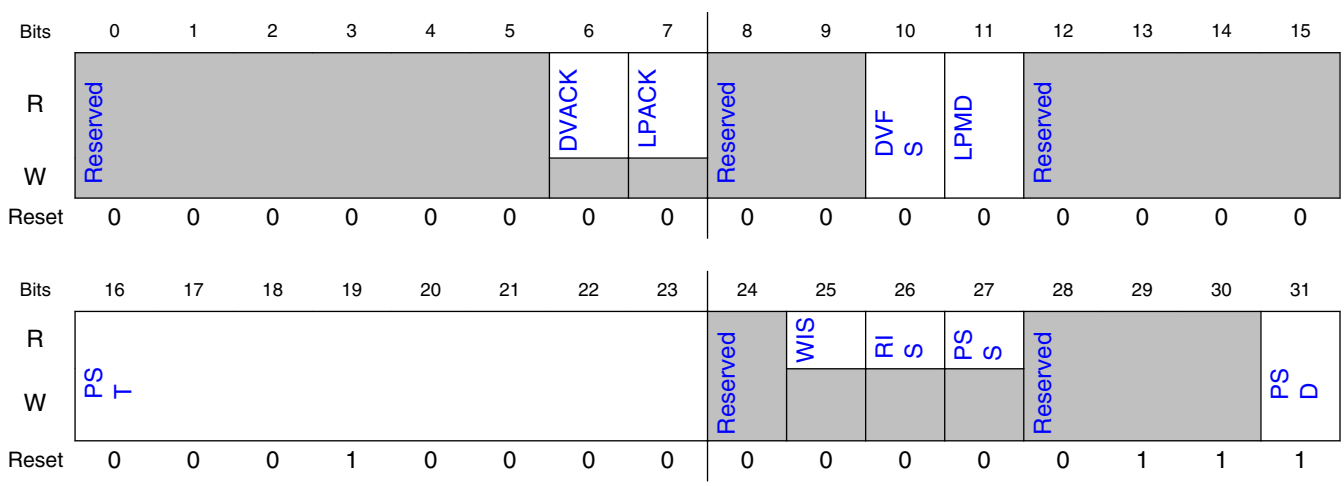
23.9.1.14.1 Offset

Register	Offset
MAPSR	404h

23.9.1.14.2 Function

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

23.9.1.14.3 Diagram



23.9.1.14.4 Fields

Field	Function
0-5 —	- Reserved
6 DVACK	DVACK General DVFS acknowledge. This read only bit indicates whether a dvfs acknowledge was asserted and that MMDC is in self-refresh mode
7	LPACK

Table continues on the next page...

Field	Function
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
8-9 —	- Reserved
10 DVFS	DVFS General DVFS request. SW request for DVFS. Assertion of this bit will yield in self-refresh entry sequence 0b - no dvfs request 1b - dvfs request
11 LPMD	LPMD General LPMD request. SW request for LPMD. Assertion of this bit will yield in self-refresh entry sequence 0b - no lpmd request 1b - lpmd request
12-15 —	- Reserved
16-23 PST	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. 00000000b - Reserved - this value is forbidden. 00000001b - timer is configured to 64 clock cycles. 00000010b - timer is configured to 128 clock cycles. 00010000b - (Default)- 1024 clock cycles. 11111111b - timer clock is configured to 16320 clock cycles.
24 —	- Reserved.
25 WIS	WIS Write Idle Status. This read only bit indicates whether write request buffer is idle (empty) or not. 0b - idle 1b - not idle
26 RIS	RIS Read Idle Status. This read only bit indicates whether read request buffer is idle (empty) or not. 0b - idle 1b - not idle
27 PSS	PSS Power Saving Status. This read only bit indicates whether the MMDC is in automatic power saving mode. 0b - not in power saving 1b - power saving
28-30 —	- Reserved.
31	PSD

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

23.9.1.15 MMDC Core Exclusive ID Monitor Register0 (MAEXIDR0)

23.9.1.15.1 Offset

Register	Offset
MAEXIDR0	408h

23.9.1.15.2 Function

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

23.9.1.15.3 Diagram

Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	EXC_ID_MONITOR1															
Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	EXC_ID_MONITOR0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

23.9.1.15.4 Fields

Field	Function
0-15 EXC_ID_MONITOR1	EXC_ID_MONITOR1 This field defines ID for Exclusive monitor#1. Default value is 0x0020

Table continues on the next page...

Field	Function
16-31 EXC_ID_MONITOR0	EXC_ID_MONITOR0 This field defines ID for Exclusive monitor#0. Default value is 0x0000

23.9.1.16 MMDC Core Exclusive ID Monitor Register1 (MAEXIDR1)

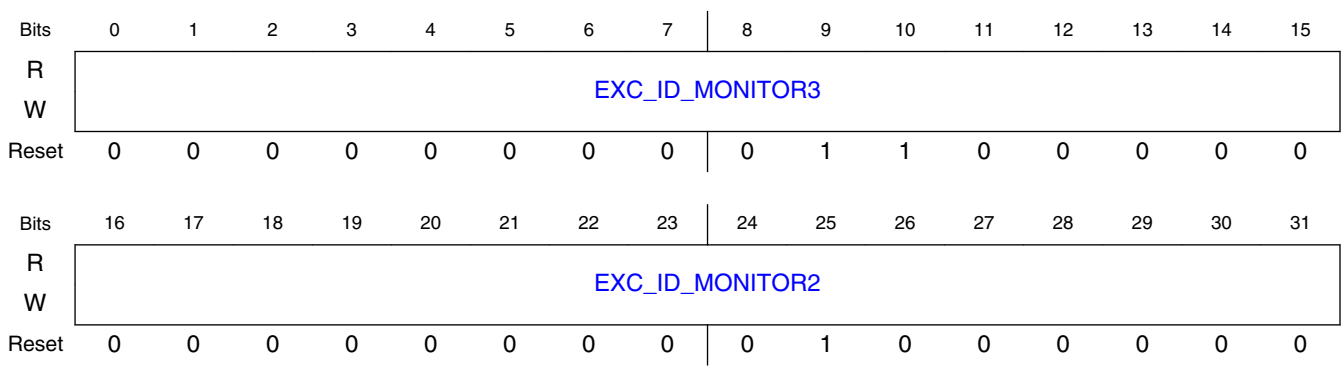
23.9.1.16.1 Offset

Register	Offset
MAEXIDR1	40Ch

23.9.1.16.2 Function

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

23.9.1.16.3 Diagram



23.9.1.16.4 Fields

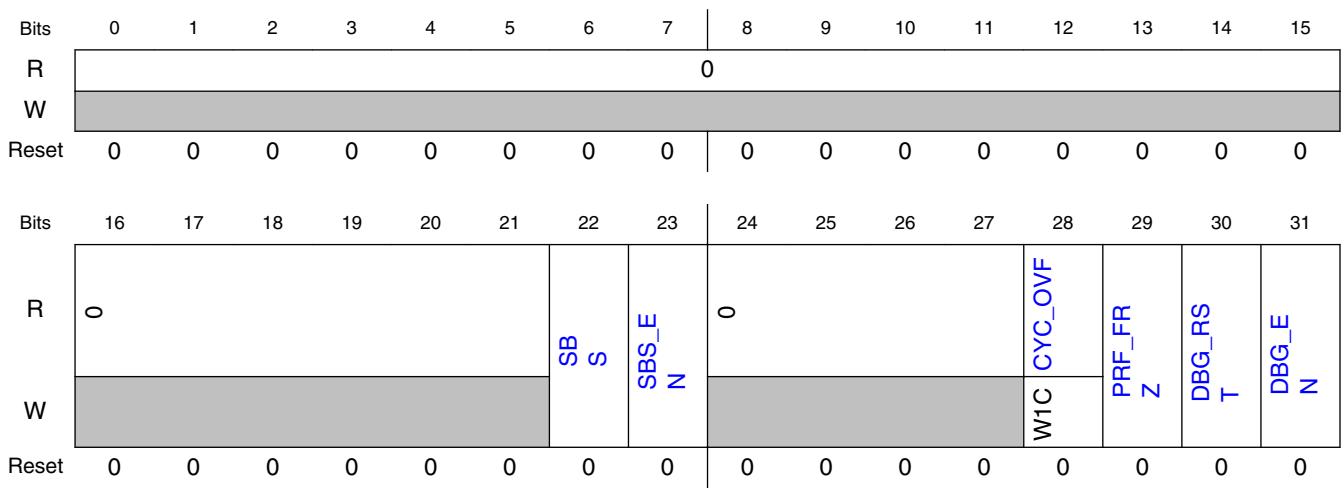
Field	Function
0-15 EXC_ID_MONITOR3	EXC_ID_MONITOR3 This field defines ID for Exclusive monitor#3. Default value is 0x0060
16-31 EXC_ID_MONITOR2	EXC_ID_MONITOR2 This field defines ID for Exclusive monitor#2. Default value is 0x0040

23.9.1.17 MMDC Core Debug and Profiling Control Register 0 (MADPCR0)

23.9.1.17.1 Offset

Register	Offset
MADPCR0	410h

23.9.1.17.2 Diagram



23.9.1.17.3 Fields

Field	Function
0-21 —	- Reserved.
22 SBS	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. 0b - No access will be launched toward the DDR 1b - Launch AXI pending access toward the DDR
23 SBS_EN	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).

Table continues on the next page...

Field	Function
	0b - disable 1b - enable
24-27 —	- Reserved.
28 CYC_OVF	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. 0b - no overflow 1b - overflow
29 PRF_FRZ	PRF_FRZ Profiling freeze. When this bit is asserted then the profiling mechanism will be froze and the associated status registers (MADPSR0-MADPSR5) will hold the the current profiling values. 0b - profiling counters are not frozen 1b - profiling counters are frozen
30 DBG_RST	DBG_RST Debug and Profiling Reset. Reset all debug and profiling counters and components. 0b - no reset 1b - reset
31 DBG_EN	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 froze and the profiling results will be sampled to the status registers (MADPSR0-MADPSR5). For further information see MMDC Profiling. default is "disable" 0b - disable 1b - enable

23.9.1.18 MMDC Core Debug and Profiling Control Register 1 (MADPCR1)

23.9.1.18.1 Offset

Register	Offset
MADPCR1	414h

23.9.1.18.2 Diagram

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

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

23.9.1.18.3 Fields

Field	Function
0-15 PRF_AXI_ID_MASK	PRF_AXI_ID_MASK Profiling AXI ID Mask. AXI ID bits which masked by this value are chosen for profiling. 0000000000000000b - AXI ID specific bit is ignored (don't care) 0000000000000001b - AXI ID specific bit is chosen for profiling
16-31 PRF_AXI_ID	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

23.9.1.19 MMDC Core Debug and Profiling Status Register 0 (MADPSR0)

23.9.1.19.1 Offset

Register	Offset
MADPSR0	418h

23.9.1.19.2 Diagram

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

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

23.9.1.19.3 Fields

Field	Function
0-31 CYC_COUNT	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

23.9.1.20 MMDC Core Debug and Profiling Status Register 1 (MADP SR1)

23.9.1.20.1 Offset

Register	Offset
MADPSR1	41Ch

23.9.1.20.2 Function

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.

23.9.1.20.3 Diagram

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

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

23.9.1.20.4 Fields

Field	Function
0-31 BUSY_COUNT	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.

23.9.1.21 MMDC Core Debug and Profiling Status Register 2 (MADP SR2)

23.9.1.21.1 Offset

Register	Offset
MADPSR2	420h

23.9.1.21.2 Function

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

23.9.1.21.3 Diagram

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

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

23.9.1.21.4 Fields

Field	Function
0-31	RD_ACC_COUNT
RD_ACC_COUNT	Profiling Read Access Count. This register reflects the total number of read accesses (per AXI ID) toward MMDC.

23.9.1.22 MMDC Core Debug and Profiling Status Register 3 (MADPSR3)

23.9.1.22.1 Offset

Register	Offset
MADPSR3	424h

23.9.1.22.2 Function

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

23.9.1.22.3 Diagram

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

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

23.9.1.22.4 Fields

Field	Function
0-31 WR_ACC_COUNT	WR_ACC_COUNT Profiling Write Access Count. This register reflects the total number of write accesses (per AXI ID) toward MMDC.

23.9.1.23 MMDC Core Debug and Profiling Status Register 4 (MADPSR4)

23.9.1.23.1 Offset

Register	Offset
MADPSR4	428h

23.9.1.23.2 Function

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

23.9.1.23.3 Diagram

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

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

23.9.1.23.4 Fields

Field	Function
0-31 RD_BYTES_COUNT	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.

23.9.1.24 MMDC Core Debug and Profiling Status Register 5 (MADPSR5)

23.9.1.24.1 Offset

Register	Offset
MADPSR5	42Ch

23.9.1.24.2 Function

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

23.9.1.24.3 Diagram

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

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

23.9.1.24.4 Fields

Field	Function
0-31	WR_BYTES_COUNT
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.

23.9.1.25 MMDC Core Step By Step Address Register (MASBS0)

23.9.1.25.1 Offset

Register	Offset
MASBS0	430h

23.9.1.25.2 Diagram

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

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

23.9.1.25.3 Fields

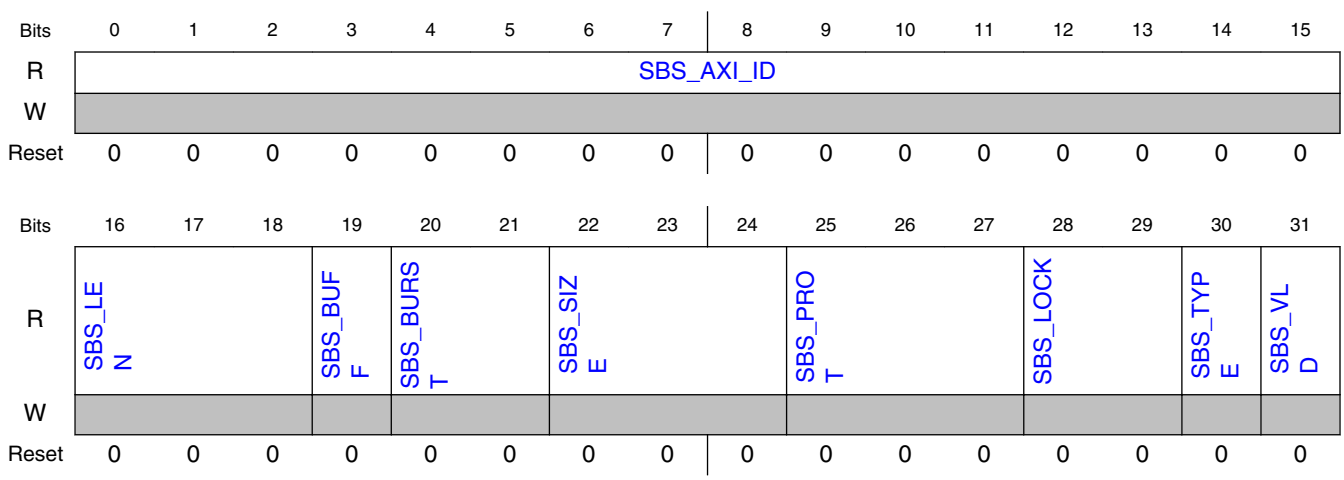
Field	Function
0-31 SBS_ADDR	SBS_ADDR Step By Step Address. These bits reflect the address of the pending request in case of step by step mode.

23.9.1.26 MMDC Core Step By Step Address Attributes Register (MASBS1)

23.9.1.26.1 Offset

Register	Offset
MASBS1	434h

23.9.1.26.2 Diagram



23.9.1.26.3 Fields

Field	Function
0-15 SBS_AXI_ID	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.
16-18	SBS_LEN

Table continues on the next page...

Field	Function
SBS_LEN	Step By Step Length. These bits reflect the AXI LENGTH of the pending request in case of step by step mode. 000b - burst of length 1 001b - burst of length 2 111b - burst of length 8
19 SBS_BUFF	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
20-21 SBS_BURST	SBS_BURST Step By Step Burst. These bits reflect the AXI BURST of the pending request in case of step by step mode. 00b - FIXED 01b - INCR burst 10b - WRAP burst 11b - reserved
22-24 SBS_SIZE	SBS_SIZE Step By Step Size. These bits reflect the AXI SIZE of the pending request in case of step by step mode. 000b - 8 bits 001b - 16 bits 010b - 32 bits 011b - 64 bits 100b - 128bits 101-111b - Reserved
25-27 SBS_PROT	SBS_PROT Step By Step Protection. These bits reflect the AXI PROT of the pending request in case of step by step mode.
28-29 SBS_LOCK	SBS_LOCK Step By Step Lock. These bits reflect the AXI LOCK of the pending request in case of step by step mode.
30 SBS_TYPE	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. 0b - write 1b - read
31 SBS_VLD	SBS_VLD Step By Step Valid. This bit reflects whether there is a pending request in case of step by step mode. 0b - not valid 1b - valid

23.9.1.27 MMDC Core General Purpose Register (MAGENP)

23.9.1.27.1 Offset

Register	Offset
MAGENP	440h

23.9.1.27.2 Function

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

23.9.1.27.3 Diagram

Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	GP31_GP0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	GP31_GP0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

23.9.1.27.4 Fields

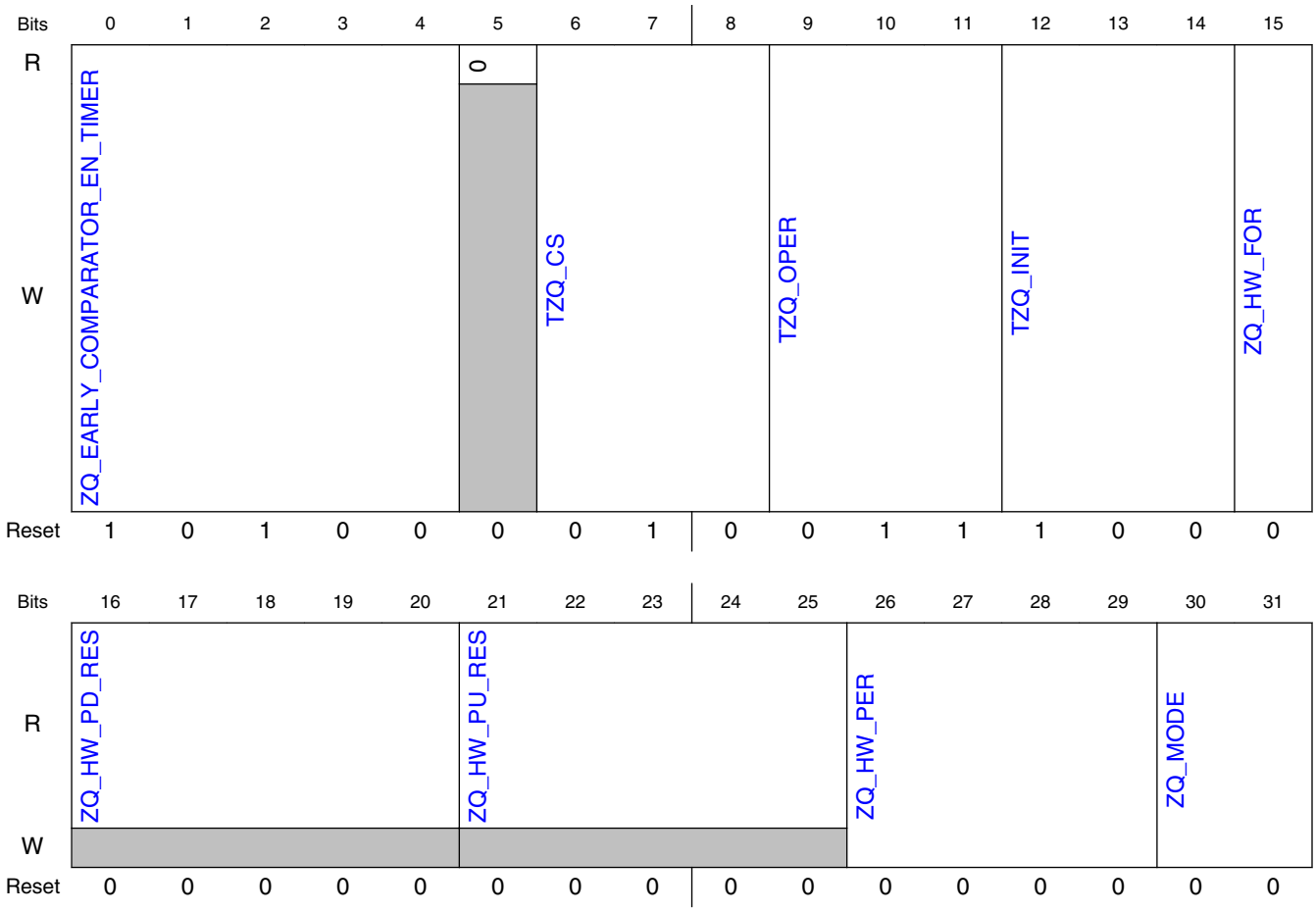
Field	Function
0-31 GP31_GP0	GP31_GP0 General purpose read/write bits.

23.9.1.28 MMDC PHY ZQ HW control register (MPZQHWCTRL)

23.9.1.28.1 Offset

Register	Offset
MPZQHWCTRL	800h

23.9.1.28.2 Diagram



23.9.1.28.3 Fields

Field	Function
0-4 ZQ_EARLY_COMPARATOR_EN_TIMER	ZQ_EARLY_COMPARATOR_EN_TIMER ZQ early comparator enable timer. This timer defines the interval between the warming up of the comparator of the ZQ calibration pad and the beginning of the ZQ calibration process with the pad 00000b - - 0x6 Reserved 00111b - 8 cycles 10100b - 21 cycles (Default) 11110b - 31 cycles 11111b - 32 cycles
5 —	- Reserved.
6-8 TZQ_CS	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.

Table continues on the next page...

Field	Function
	NOTE: This field should not be update during ZQ calibration. 000b - Reserved 001b - Reserved 010b - 128 cycles (Default) 011b - 256 cycles 100b - 512 cycles 101b - 1024 cycles 110-111b - Reserved
9-11 TZQ_OPER	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 issued 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:** This field should not be update during ZQ calibration. 000b - Reserved 001b - Reserved 010b - 128 cycles 011b - 256 cycles - Default (JEDEC value for DDR3L) 100b - 512 cycles 101b - 1024 cycles 110-111b - Reserved
12-14 TZQ_INIT	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:** This field should not be update during ZQ calibration. 000b - Reserved 001b - Reserved 010b - 128 cycles 011b - 256 cycles 100b - 512 cycles - Default (JEDEC value for DDR3L) 101b - 1024 cycles 110-111b - Reserved
15 ZQ_HW_FOR	ZQ_HW_FOR Force ZQ automatic calibration process with the ZQ calibration pad. When this bit is asserted then the MMDC will issue one ZQ automatic calibration process with the 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"
16-20 ZQ_HW_PD_RES	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 ZQ calibration pad. **NOTE:** An offset can be applied to this result, see MMDC_MPPDCMPR2[ZQ_PD_OFFSET]. 00000b - Max. resistance. 11111b - Min. resistance.
21-25 ZQ_HW_PU_RES	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 ZQ calibration pad.

Table continues on the next page...

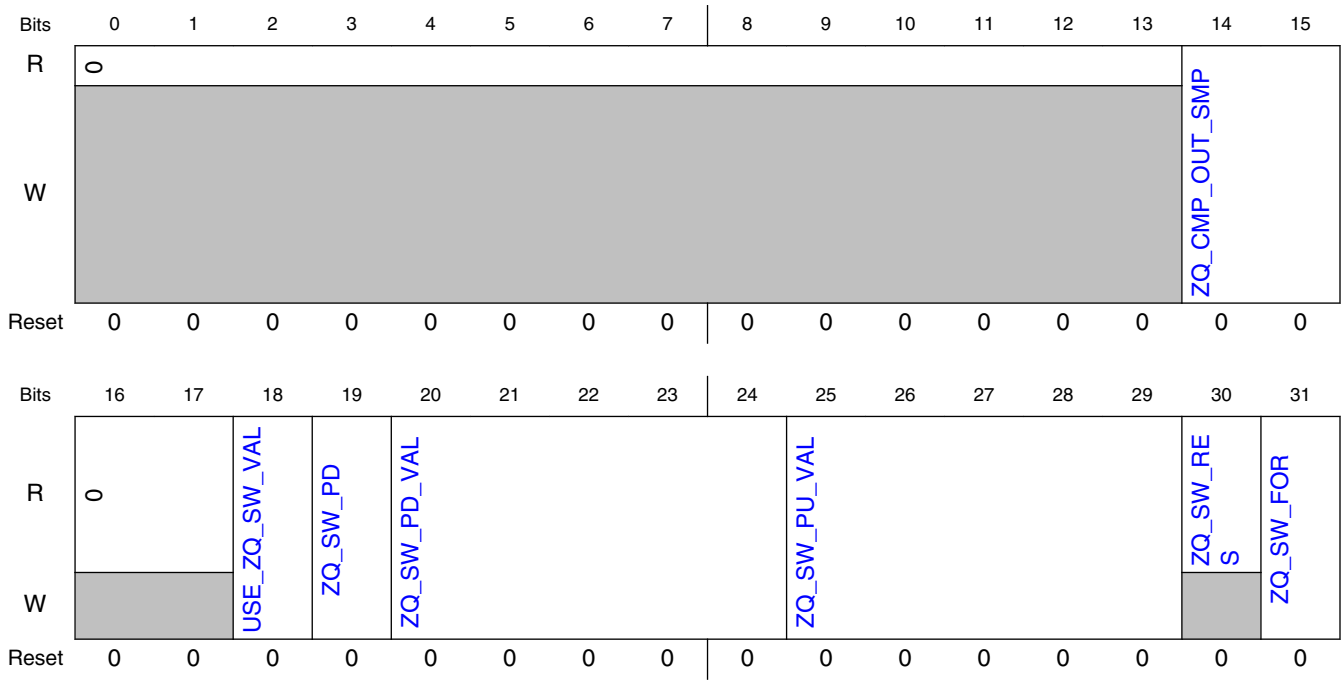
Field	Function
	NOTE: An offset can be applied to this result, see MMDC_MPPDCMPR2[ZQ_PU_OFFSET] 00000b - Min. resistance. 11111b - Max. resistance.
26-29 ZQ_HW_PER	ZQ_HW_PER ZQ periodic calibration time. This field determines how often the periodic ZQ calibration is performed. This field is applied for both ZQ short calibration and ZQ automatic calibration process with ZQ calibration pad. Whenever this timer is expired then according to ZQ_MODE the ZQ automatic calibration process with the 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" 0000b - ZQ calibration is performed every 1 ms. 0001b - ZQ calibration is performed every 2 ms. 0010b - ZQ calibration is performed every 4 ms. 1010b - ZQ calibration is performed every 1 sec. 1110b - ZQ calibration is performed every 16 sec. 1111b - ZQ calibration is performed every 32 sec.
30-31 ZQ_MODE	ZQ_MODE ZQ calibration mode: 00b - No ZQ calibration is issued. (Default) 01b - ZQ calibration is issued to ZQ calibration pad together with ZQ long command to the external DDR device only when exiting self refresh. 10b - ZQ calibration command long/short is issued only to the external DDR device periodically and when exiting self refresh 11b - ZQ calibration is issued to ZQ calibration pad together with ZQ calibration command long/short to the external DDR device periodically and when exiting self refresh

23.9.1.29 MMDC PHY ZQ SW control register (MPZQSWCTRL)

23.9.1.29.1 Offset

Register	Offset
MPZQSWCTRL	804h

23.9.1.29.2 Diagram



23.9.1.29.3 Fields

Field	Function
0-13 —	- Reserved
14-15 ZQ_CMP_OUT_SMP	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 ZQ calibration pad 00b - 7 cycles 01b - 15 cycles 10b - 23 cycles 11b - 31 cycles
16-17 —	- Reserved
18 USE_ZQ_SW_VAL	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. 0b - Fields ZQ_HW_PD_VAL & ZQ_HW_PU_VAL will be driven to I/O pads resistor controls. 1b - Fields ZQ_SW_PD_VAL & ZQ_SW_PU_VAL will be driven to I/O pads resistor controls.
19 ZQ_SW_PD	ZQ_SW_PD ZQ software PU/PD calibration. This bit determines the calibration stage (PU or PD).

Table continues on the next page...

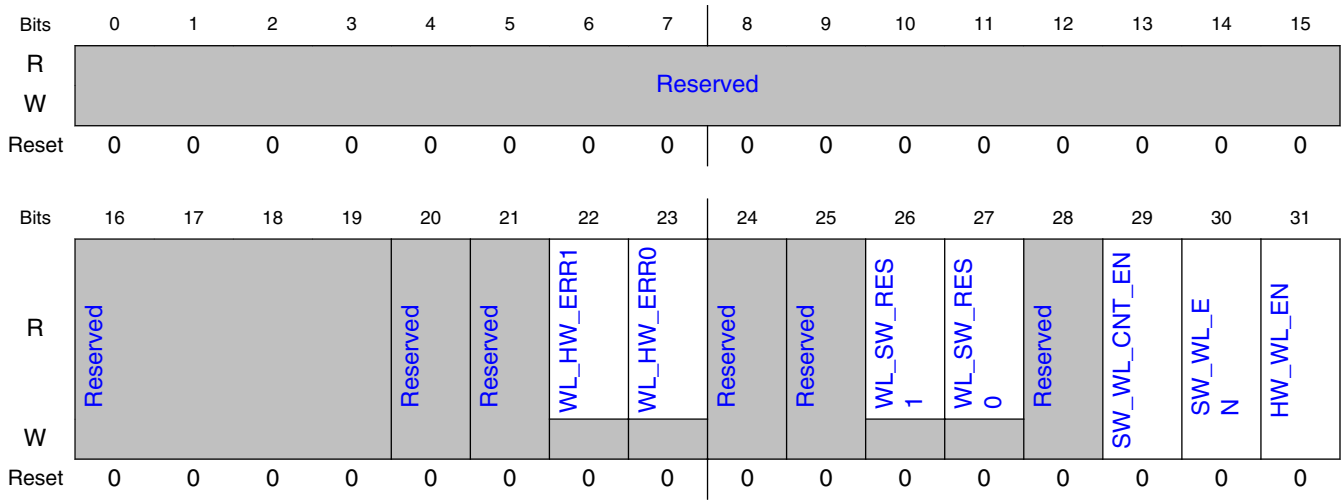
Field	Function
	0b - PU resistor calibration 1b - PD resistor calibration
20-24 ZQ_SW_PD_VAL	ZQ_SW_PD_VAL ZQ software pull-down resistence. This field determines the value of the PD resistor during SW ZQ calibration. 00000b - Max. resistance. 11111b - Min. resistance.
25-29 ZQ_SW_PU_VAL	ZQ_SW_PU_VAL ZQ software pull-up resistence. This field determines the value of the PU resistor during SW ZQ calibration. 00000b - Min. resistance. 11111b - Max. resistance.
30 ZQ_SW_RES	ZQ_SW_RES ZQ software calibration result. This bit reflects the ZQ calibration voltage comparator value. 0b - Current ZQ calibration voltage is less than VDD/2. 1b - Current ZQ calibration voltage is more than VDD/2
31 ZQ_SW_FOR	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

23.9.1.30 MMDC PHY Write Leveling Configuration and Error Status Register (MPWLGCR)

23.9.1.30.1 Offset

Register	Offset
MPWLGCR	808h

23.9.1.30.2 Diagram



23.9.1.30.3 Fields

Field	Function
0-19 —	- Reserved
20 —	- Reserved
21 —	- Reserved
22 WL_HW_ERR1	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. This bit is valid only upon completion of the write-leveling HW calibration (i.e. HW_WL_EN bit is de-asserted) 0b - No error was found on byte1 during write-leveling HW calibration. 1b - An error was found on byte1 during write-leveling HW calibration.
23 WL_HW_ERR0	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) 0b - No error was found on byte0 during write-leveling HW calibration. 1b - An error was found on byte0 during write-leveling HW calibration.
24 —	- Reserved
25 —	- Reserved

Table continues on the next page...

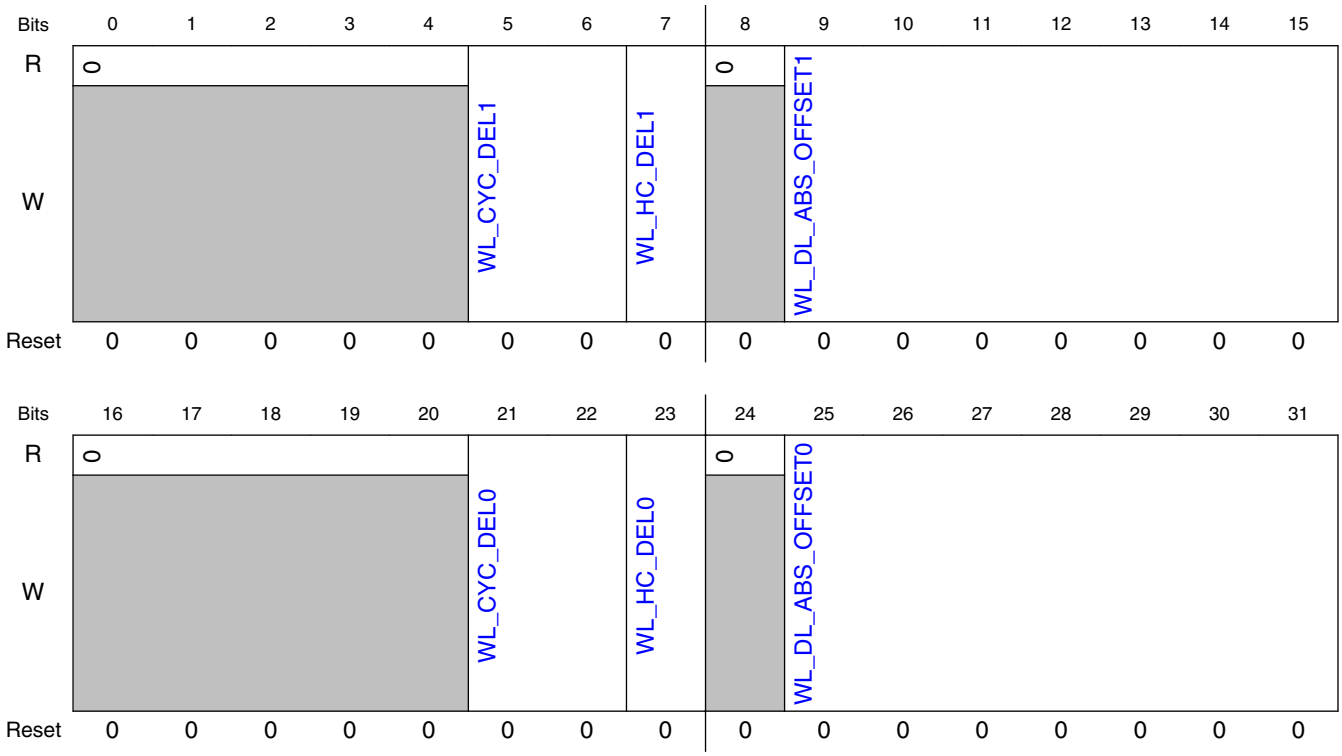
Field	Function
26 WL_SW_RES1	WL_SW_RES1 Byte1 write-leveling software result. This bit reflects the value that is driven by the DDR device on DQ8 during SW write-leveling. 0b - DQS1 sampled low CK during SW write-leveling. 1b - DQS1 sampled high CK during SW write-leveling.
27 WL_SW_RES0	WL_SW_RES0 Byte0 write-leveling software result. This bit reflects the value that is driven by the DDR device on DQ0 during SW write-leveling. 0b - DQS0 sampled low CK during SW write-leveling. 1b - DQS0 sampled high CK during SW write-leveling.
28 —	- Reserved
29 SW_WL_CNT_EN	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 0b - MMDC doesn't count 25+15 cycles before issuing write-leveling DQS. 1b - MMDC counts 25+15 cycles before issuing write-leveling DQS.
30 SW_WL_EN	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.
31 HW_WL_EN	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 **NOTE:** Before issuing the first DQS the MMDC counts 25 + 15 cycles automatically as required by the standard.

23.9.1.31 MMDC PHY Write Leveling Delay Control Register 0 (MPWLDECTRL0)

23.9.1.31.1 Offset

Register	Offset
MPWLDECTRL0	80Ch

23.9.1.31.2 Diagram



23.9.1.31.3 Fields

Field	Function
0-4 —	- Reserved
5-6 WL_CYC_DEL1	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. 00b - No delay is added. 01b - 1 cycle delay is added. 10b - 2 cycles delay is added. 11b - Reserved.
7 WL_HC_DEL1	WL_HC_DEL1

Table continues on the next page...

Field	Function
	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. 0b - No delay is added. 1b - Half cycle delay is added.
8 —	- Reserved
9-15 WL_DL_ABS_OFFSET1	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.
16-20 —	- Reserved
21-22 WL_CYC_DEL0	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. 00b - No delay is added. 01b - 1 cycle delay is added. 10b - 2 cycles delay is added. 11b - Reserved.
23 WL_HC_DEL0	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). 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.

Table continues on the next page...

Field	Function
	0b - No delay is added. 1b - Half cycle delay is added.
24 —	- Reserved
25-31 WL_DL_ABS_OFFSET0	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.

23.9.1.32 MMDC PHY Write Leveling delay-line Status Register (MPWLDLST)

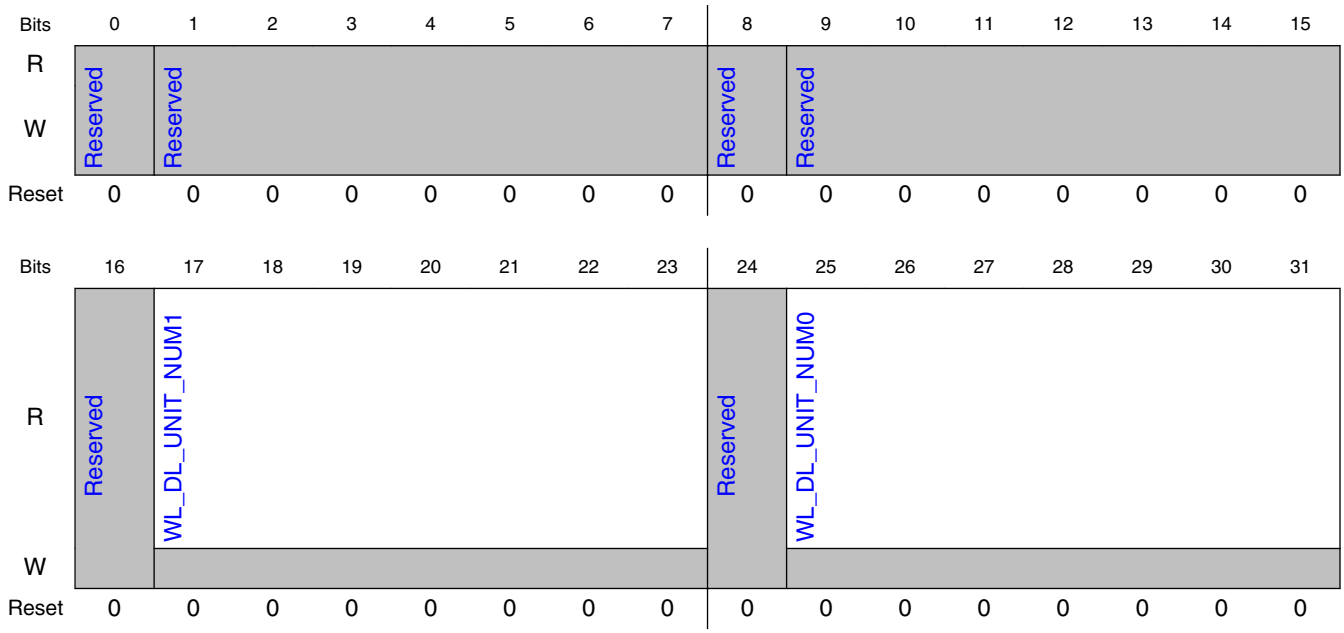
23.9.1.32.1 Offset

Register	Offset
MPWLDLST	814h

23.9.1.32.2 Function

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

23.9.1.32.3 Diagram



23.9.1.32.4 Fields

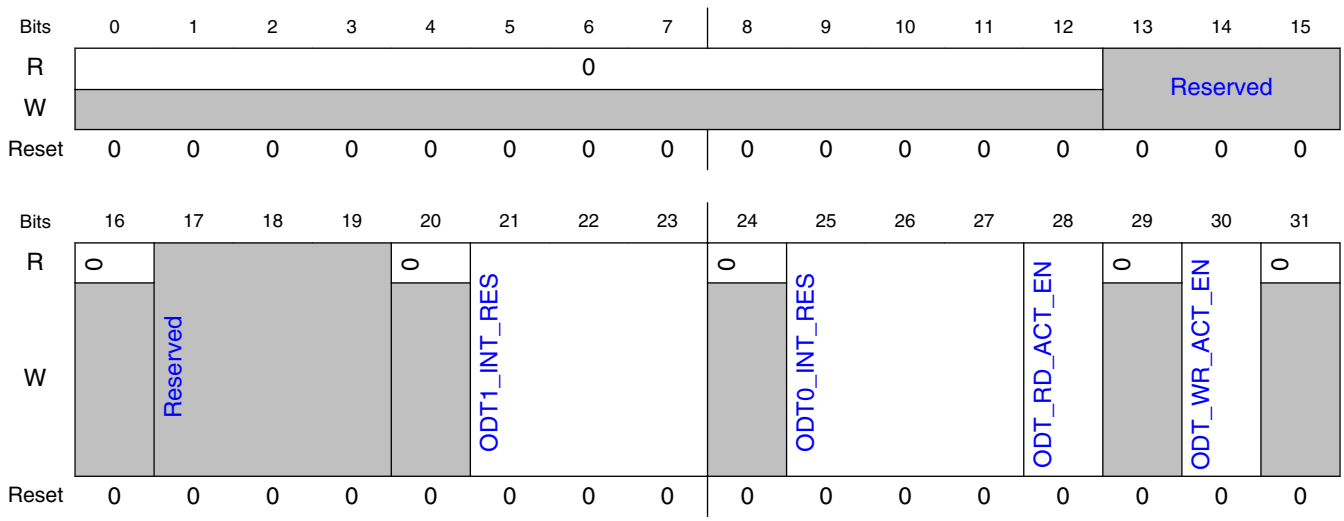
Field	Function
0 —	- Reserved
1-7 —	- Reserved
8 —	- Reserved
9-15 —	- Reserved
16 —	- Reserved
17-23 WL_DL_UNIT_NUM1	WL_DL_UNIT_NUM1 This field reflects the number of delay units that are actually used by write leveling delay-line 1.
24 —	- Reserved
25-31 WL_DL_UNIT_NUM0	WL_DL_UNIT_NUM0 This field reflects the number of delay units that are actually used by write leveling delay-line 0.

23.9.1.33 MMDC PHY ODT control register (MPODTCTRL)

23.9.1.33.1 Offset

Register	Offset
MPODTCTRL	818h

23.9.1.33.2 Diagram



23.9.1.33.3 Fields

Field	Function
0-12 —	- Reserved
13-15 —	- Reserved
16 —	- Reserved
17-19 —	- Reserved
20 —	- Reserved
21-23	ODT1_INT_RES

Table continues on the next page...

Field	Function
ODT1_INT_RES	On chip ODT byte1 resistor - This field determines the Rtt_Nom of the on chip ODT byte1 resistor during read accesses. 000b - Rtt_Nom Disabled. 001b - Rtt_Nom 120 Ohm 010b - Rtt_Nom 60 Ohm 011b - Rtt_Nom 40 Ohm 100b - Rtt_Nom 30 Ohm 101b - Rtt_Nom 24 Ohm 110b - Rtt_Nom 20 Ohm 111b - Rtt_Nom 17 Ohm
24 —	- Reserved
25-27 ODT0_INT_RES	ODT0_INT_RES On chip ODT byte0 resistor - This field determines the Rtt_Nom of the on chip ODT byte0 resistor during read accesses. 000b - Rtt_Nom Disabled. 001b - Rtt_Nom 120 Ohm 010b - Rtt_Nom 60 Ohm 011b - Rtt_Nom 40 Ohm 100b - Rtt_Nom 30 Ohm 101b - Rtt_Nom 24 Ohm 110b - Rtt_Nom 20 Ohm 111b - Rtt_Nom 17 Ohm
28 ODT_RD_ACT_EN	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. 0b - Active CS ODT pin is disabled during read access. 1b - Active CS ODT pin is enabled during read access.
29 —	Reserved.
30 ODT_WR_ACT_EN	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. 0b - Active CS ODT pin is disabled during write access. 1b - Active CS ODT pin is enabled during write access.
31 —	Reserved.

23.9.1.34 MMDC PHY Read DQ Byte0 Delay Register (MPRDDQBY 0DL)

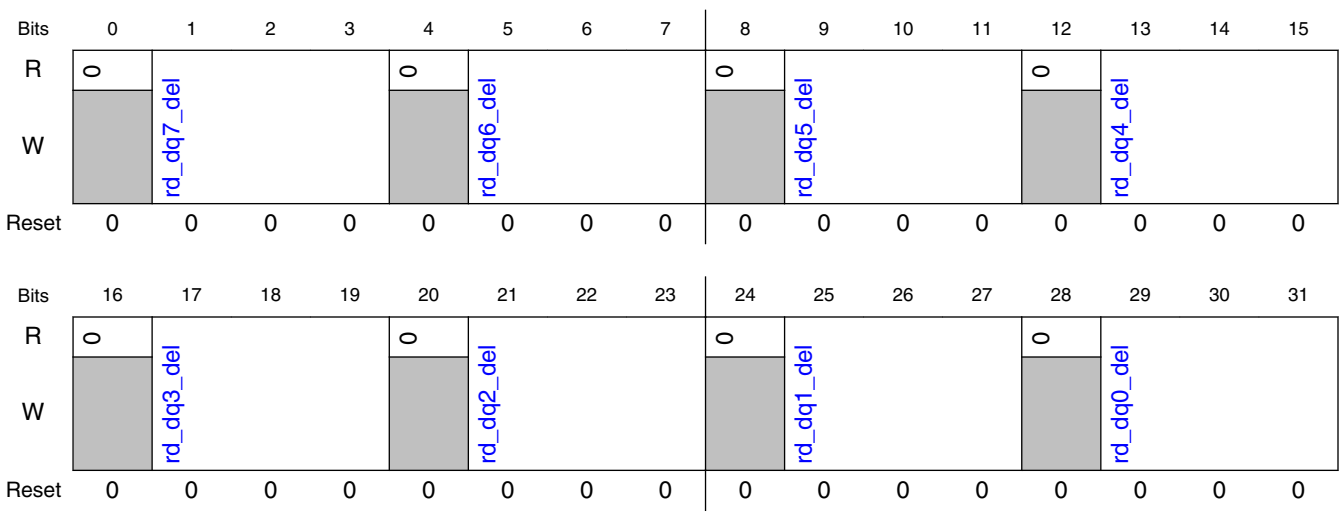
23.9.1.34.1 Offset

Register	Offset
MPRDDQBY0DL	81Ch

23.9.1.34.2 Function

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.

23.9.1.34.3 Diagram



23.9.1.34.4 Fields

Field	Function
0 —	- Reserved
1-3 rd_dq7_del	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. 000b - No change in dq7 delay 001b - Add dq7 delay of 1 delay unit 010b - Add dq7 delay of 2 delay units. 011b - Add dq7 delay of 3 delay units. 100b - Add dq7 delay of 4 delay units. 101b - Add dq7 delay of 5 delay units. 110b - Add dq7 delay of 6 delay units.

Table continues on the next page...

Field	Function
	111b - Add dq7 delay of 7 delay units.
4 —	- Reserved
5-7 rd_dq6_del	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. 000b - No change in dq6 delay 001b - Add dq6 delay of 1 delay unit 010b - Add dq6 delay of 2 delay units. 011b - Add dq6 delay of 3 delay units. 100b - Add dq6 delay of 4 delay units. 101b - Add dq6 delay of 5 delay units. 110b - Add dq6 delay of 6 delay units. 111b - Add dq6 delay of 7 delay units.
8 —	- Reserved
9-11 rd_dq5_del	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. 000b - No change in dq5 delay 001b - Add dq5 delay of 1 delay unit 010b - Add dq5 delay of 2 delay units. 011b - Add dq5 delay of 3 delay units. 100b - Add dq5 delay of 4 delay units. 101b - Add dq5 delay of 5 delay units. 110b - Add dq5 delay of 6 delay units. 111b - Add dq5 delay of 7 delay units.
12 —	- Reserved
13-15 rd_dq4_del	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. 000b - No change in dq4 delay 001b - Add dq4 delay of 1 delay unit 010b - Add dq4 delay of 2 delay units. 011b - Add dq4 delay of 3 delay units. 100b - Add dq4 delay of 4 delay units. 101b - Add dq4 delay of 5 delay units. 110b - Add dq4 delay of 6 delay units. 111b - Add dq4 delay of 7 delay units.
16 —	- Reserved
17-19 rd_dq3_del	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. 000b - No change in dq3 delay 001b - Add dq3 delay of 1 delay unit 010b - Add dq3 delay of 2 delay units.

Table continues on the next page...

Field	Function
	011b - Add dq3 delay of 3 delay units. 100b - Add dq3 delay of 4 delay units. 101b - Add dq3 delay of 5 delay units. 110b - Add dq3 delay of 6 delay units. 111b - Add dq3 delay of 7 delay units.
20 —	- Reserved
21-23 rd_dq2_del	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. 000b - No change in dq2 delay 001b - Add dq2 delay of 1 delay unit 010b - Add dq2 delay of 2 delay units. 011b - Add dq2 delay of 3 delay units. 100b - Add dq2 delay of 4 delay units. 101b - Add dq2 delay of 5 delay units. 110b - Add dq2 delay of 6 delay units. 111b - Add dq2 delay of 7 delay units.
24 —	- Reserved
25-27 rd_dq1_del	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. 000b - No change in dq1 delay 001b - Add dq1 delay of 1 delay unit 010b - Add dq1 delay of 2 delay units. 011b - Add dq1 delay of 3 delay units. 100b - Add dq1 delay of 4 delay units. 101b - Add dq1 delay of 5 delay units. 110b - Add dq1 delay of 6 delay units. 111b - Add dq1 delay of 7 delay units.
28 —	- Reserved
29-31 rd_dq0_del	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. 000b - No change in dq0 delay 001b - Add dq0 delay of 1 delay unit 010b - Add dq0 delay of 2 delay units. 011b - Add dq0 delay of 3 delay units. 100b - Add dq0 delay of 4 delay units. 101b - Add dq0 delay of 5 delay units. 110b - Add dq0 delay of 6 delay units. 111b - Add dq0 delay of 7 delay units.

23.9.1.35 MMDC PHY Read DQ Byte1 Delay Register (MPRDDQBY1DL)

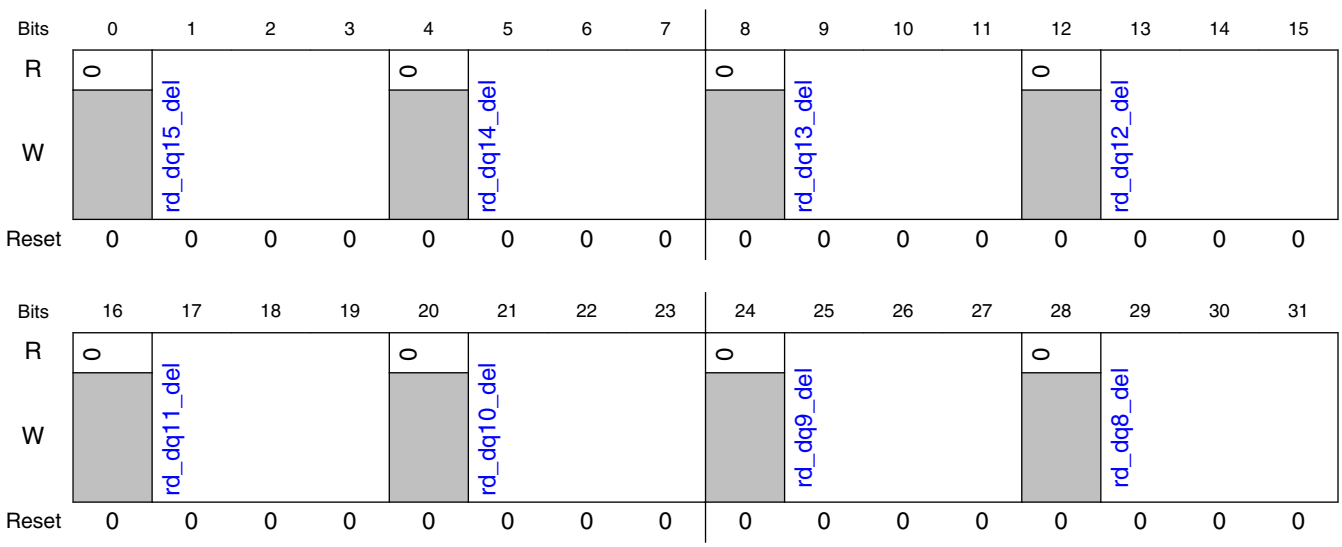
23.9.1.35.1 Offset

Register	Offset
MPRDDQBY1DL	820h

23.9.1.35.2 Function

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

23.9.1.35.3 Diagram



23.9.1.35.4 Fields

Field	Function
0 —	- Reserved
1-3 rd_dq15_del	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. 000b - No change in dq15 delay 001b - Add dq15 delay of 1 delay unit

Table continues on the next page...

Field	Function
	010b - Add dq15 delay of 2 delay units. 011b - Add dq15 delay of 3 delay units. 100b - Add dq15 delay of 4 delay units. 101b - Add dq15 delay of 5 delay units. 110b - Add dq15 delay of 6 delay units. 111b - Add dq15 delay of 7 delay units.
4 —	- Reserved
5-7 rd_dq14_del	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. 000b - No change in dq14 delay 001b - Add dq14 delay of 1 delay unit 010b - Add dq14 delay of 2 delay units. 011b - Add dq14 delay of 3 delay units. 100b - Add dq14 delay of 4 delay units. 101b - Add dq14 delay of 5 delay units. 110b - Add dq14 delay of 6 delay units. 111b - Add dq14 delay of 7 delay units.
8 —	- Reserved
9-11 rd_dq13_del	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. 000b - No change in dq13 delay 001b - Add dq13 delay of 1 delay unit 010b - Add dq13 delay of 2 delay units. 011b - Add dq13 delay of 3 delay units. 100b - Add dq13 delay of 4 delay units. 101b - Add dq13 delay of 5 delay units. 110b - Add dq13 delay of 6 delay units. 111b - Add dq13 delay of 7 delay units.
12 —	- Reserved
13-15 rd_dq12_del	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. 000b - No change in dq12 delay 001b - Add dq12 delay of 1 delay unit 010b - Add dq12 delay of 2 delay units. 011b - Add dq12 delay of 3 delay units. 100b - Add dq12 delay of 4 delay units. 101b - Add dq12 delay of 5 delay units. 110b - Add dq12 delay of 6 delay units. 111b - Add dq12 delay of 7 delay units.
16 —	- Reserved
17-19 rd_dq11_del	rd_dq11_del

Table continues on the next page...

Field	Function
	Read dqs1 to dq11 delay fine-tuning. This field holds the number of delay units that are added to dq11 relative to dqs1. 000b - No change in dq11 delay 001b - Add dq11 delay of 1 delay unit 010b - Add dq11 delay of 2 delay units. 011b - Add dq11 delay of 3 delay units. 100b - Add dq11 delay of 4 delay units. 101b - Add dq11 delay of 5 delay units. 110b - Add dq11 delay of 6 delay units. 111b - Add dq11 delay of 7 delay units.
20 —	- Reserved
21-23 rd_dq10_del	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. 000b - No change in dq10 delay 001b - Add dq10 delay of 1 delay unit 010b - Add dq10 delay of 2 delay units. 011b - Add dq10 delay of 3 delay units. 100b - Add dq10 delay of 4 delay units. 101b - Add dq10 delay of 5 delay unit 110b - Add dq10 delay of 6 delay units. 111b - Add dq10 delay of 7 delay units.
24 —	- Reserved
25-27 rd_dq9_del	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. 000b - No change in dq9 delay 001b - Add dq9 delay of 1 delay unit 010b - Add dq9 delay of 2 delay units. 011b - Add dq9 delay of 3 delay units. 100b - Add dq9 delay of 4 delay units. 101b - Add dq9 delay of 5 delay units. 110b - Add dq9 delay of 6 delay units. 111b - Add dq9 delay of 7 delay units.
28 —	- Reserved
29-31 rd_dq8_del	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. 000b - No change in dq8 delay 001b - Add dq8 delay of 1 delay unit 010b - Add dq8 delay of 2 delay units. 011b - Add dq8 delay of 3 delay units. 100b - Add dq8 delay of 4 delay units. 101b - Add dq8 delay of 5 delay units. 110b - Add dq8 delay of 6 delay units. 111b - Add dq8 delay of 7 delay units.

23.9.1.36 MMDC PHY Write DQ Byte0 Delay Register (MPWRDQBY0DL)

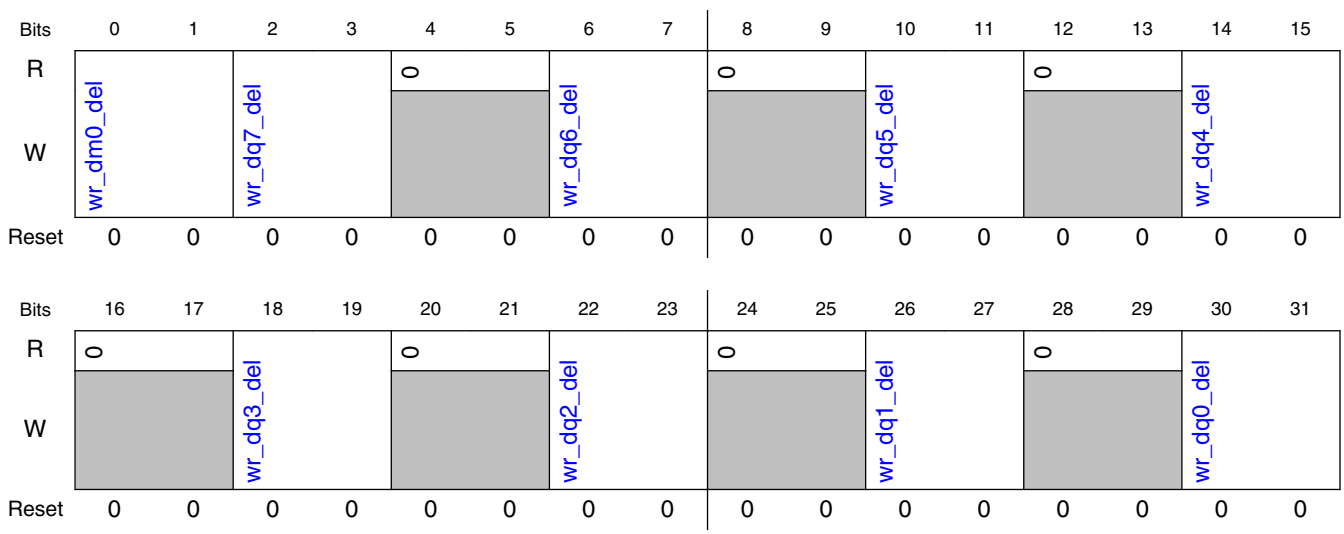
23.9.1.36.1 Offset

Register	Offset
MPWRDQBY0DL	82Ch

23.9.1.36.2 Function

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

23.9.1.36.3 Diagram



23.9.1.36.4 Fields

Field	Function
0-1 wr_dm0_del	wr_dm0_del Write dm0 delay fine-tuning. This field holds the number of delay units that are added to dm0 relative to dqs0. 00b - No change in dm0 delay 01b - Add dm0 delay of 1 delay unit. 10b - Add dm0 delay of 2 delay units.

Table continues on the next page...

Field	Function
	11b - Add dm0 delay of 3 delay units.
2-3 wr_dq7_del	wr_dq7_del Write dq7 delay fine-tuning. This field holds the number of delay units that are added to dq7 relative to dqs0. 00b - No change in dq7 delay 01b - Add dq7 delay of 1 delay unit. 10b - Add dq7 delay of 2 delay units. 11b - Add dq7 delay of 3 delay units.
4-5 —	- Reserved
6-7 wr_dq6_del	wr_dq6_del Write dq6 delay fine-tuning. This field holds the number of delay units that are added to dq6 relative to dqs0. 00b - No change in dq6 delay 01b - Add dq6 delay of 1 delay unit. 10b - Add dq6 delay of 2 delay units. 11b - Add dq6 delay of 3 delay units.
8-9 —	- Reserved
10-11 wr_dq5_del	wr_dq5_del Write dq5 delay fine-tuning. This field holds the number of delay units that are added to dq5 relative to dqs0. 00b - No change in dq5 delay 01b - Add dq5 delay of 1 delay unit. 10b - Add dq5 delay of 2 delay units. 11b - Add dq5 delay of 3 delay units.
12-13 —	- Reserved
14-15 wr_dq4_del	wr_dq4_del Write dq4 delay fine-tuning. This field holds the number of delay units that are added to dq4 relative to dqs0. 00b - No change in dq4 delay 01b - Add dq4 delay of 1 delay unit.. 10b - Add dq4 delay of 2 delay units. 11b - Add dq4 delay of 3 delay units.
16-17 —	- Reserved
18-19 wr_dq3_del	wr_dq3_del Write dq3 delay fine-tuning. This field holds the number of delay units that are added to dq3 relative to dqs0. 00b - No change in dq3 delay 01b - Add dq3 delay of 1 delay unit. 10b - Add dq3 delay of 2 delay units. 11b - Add dq3 delay of 3 delay units.
20-21 —	- Reserved

Table continues on the next page...

Field	Function
22-23 wr_dq2_del	wr_dq2_del Write dq2 delay fine-tuning. This field holds the number of delay units that are added to dq2 relative to dqs0. 00b - No change in dq2 delay 01b - Add dq2 delay of 1 delay unit. 10b - Add dq2 delay of 2 delay units. 11b - Add dq2 delay of 3 delay units.
24-25 —	- Reserved
26-27 wr_dq1_del	wr_dq1_del Write dq1 delay fine-tuning. This field holds the number of delay units that are added to dq1 relative to dqs0. 00b - No change in dq1 delay 01b - Add dq1 delay of 1 delay unit. 10b - Add dq1 delay of 2 delay units. 11b - Add dq1 delay of 3 delay units.
28-29 —	- Reserved
30-31 wr_dq0_del	wr_dq0_del Write dq0 delay fine-tuning. This field holds the number of delay units that are added to dq0 relative to dqs0. 00b - No change in dq0 delay 01b - Add dq0 delay of 1 delay unit. 10b - Add dq0 delay of 2 delay units. 11b - Add dq0 delay of 3 delay units.

23.9.1.37 MMDC PHY Write DQ Byte1 Delay Register (MPWRDQBY1DL)

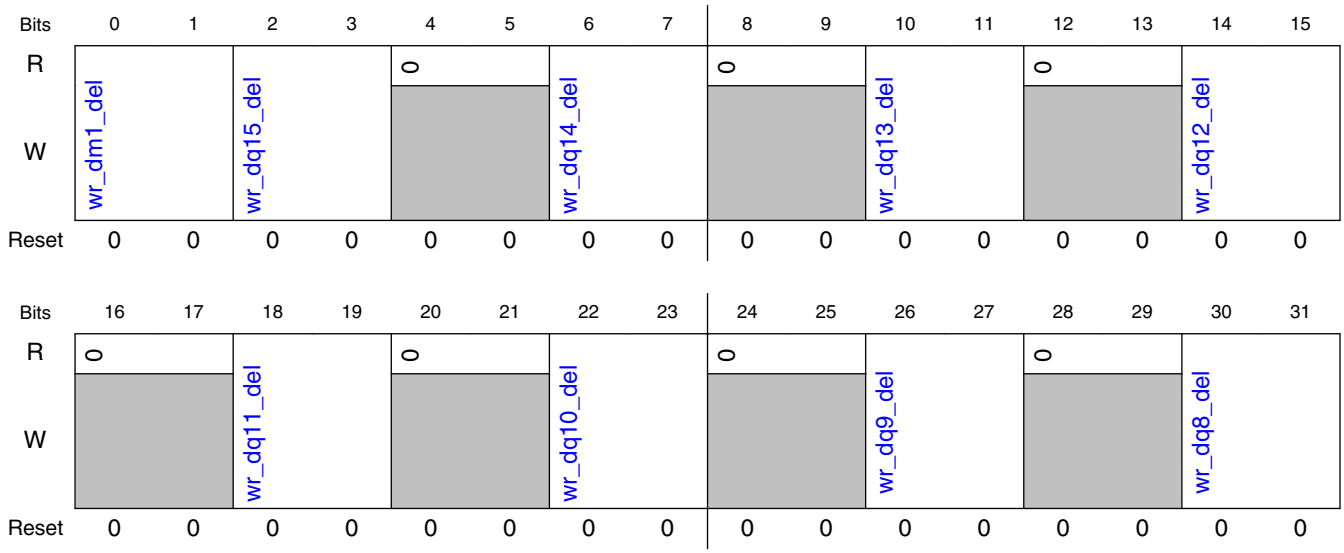
23.9.1.37.1 Offset

Register	Offset
MPWRDQBY1DL	830h

23.9.1.37.2 Function

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

23.9.1.37.3 Diagram



23.9.1.37.4 Fields

Field	Function
0-1 wr_dm1_del	wr_dm1_del Write dm1 delay fine-tuning. This field holds the number of delay units that are added to dm1 relative to dqs1. 00b - No change in dm1 delay 01b - Add dm1 delay of 1 delay unit. 10b - Add dm1 delay of 2 delay units. 11b - Add dm1 delay of 3 delay units.
2-3 wr_dq15_del	wr_dq15_del Write dq15 delay fine-tuning. This field holds the number of delay units that are added to dq15 relative to dqs1. 00b - No change in dq15 delay 01b - Add dq15 delay of 1 delay unit. 10b - Add dq15 delay of 2 delay units. 11b - Add dq15 delay of 3 delay units.
4-5 —	- Reserved
6-7 wr_dq14_del	wr_dq14_del Write dq14 delay fine-tuning. This field holds the number of delay units that are added to dq14 relative to dqs1. 00b - No change in dq14 delay 01b - Add dq14 delay of 1 delay unit. 10b - Add dq14 delay of 2 delay units. 11b - Add dq14 delay of 3 delay units.
8-9 —	- Reserved

Table continues on the next page...

Field	Function
10-11 wr_dq13_del	wr_dq13_del Write dq13 delay fine-tuning. This field holds the number of delay units that are added to dq13 relative to dqs1. 00b - No change in dq13 delay 01b - Add dq13 delay of 1 delay unit. 10b - Add dq13 delay of 2 delay units. 11b - Add dq13 delay of 3 delay units.
12-13 —	- Reserved
14-15 wr_dq12_del	wr_dq12_del Write dq12 delay fine-tuning. This field holds the number of delay units that are added to dq12 relative to dqs1. 00b - No change in dq12 delay 01b - Add dq12 delay of 1 delay unit. 10b - Add dq12 delay of 2 delay units. 11b - Add dq12 delay of 3 delay units.
16-17 —	- Reserved
18-19 wr_dq11_del	wr_dq11_del Write dq11 delay fine-tuning. This field holds the number of delay units that are added to dq11 relative to dqs1. 00b - No change in dq11 delay 01b - Add dq11 delay of 1 delay unit. 10b - Add dq11 delay of 2 delay units. 11b - Add dq11 delay of 3 delay units.
20-21 —	- Reserved
22-23 wr_dq10_del	wr_dq10_del Write dq10 delay fine-tuning. This field holds the number of delay units that are added to dq10 relative to dqs1. 00b - No change in dq10 delay 01b - Add dq10 delay of 1 delay unit. 10b - Add dq10 delay of 2 delay units. 11b - Add dq10 delay of 3 delay units.
24-25 —	- Reserved
26-27 wr_dq9_del	wr_dq9_del Write dq9 delay fine-tuning. This field holds the number of delay units that are added to dq9 relative to dqs1. 00b - No change in dq9 delay 01b - Add dq9 delay of 1 delay unit. 10b - Add dq9 delay of 2 delay units. 11b - Add dq9 delay of 3 delay units.
28-29 —	- Reserved
30-31	wr_dq8_del

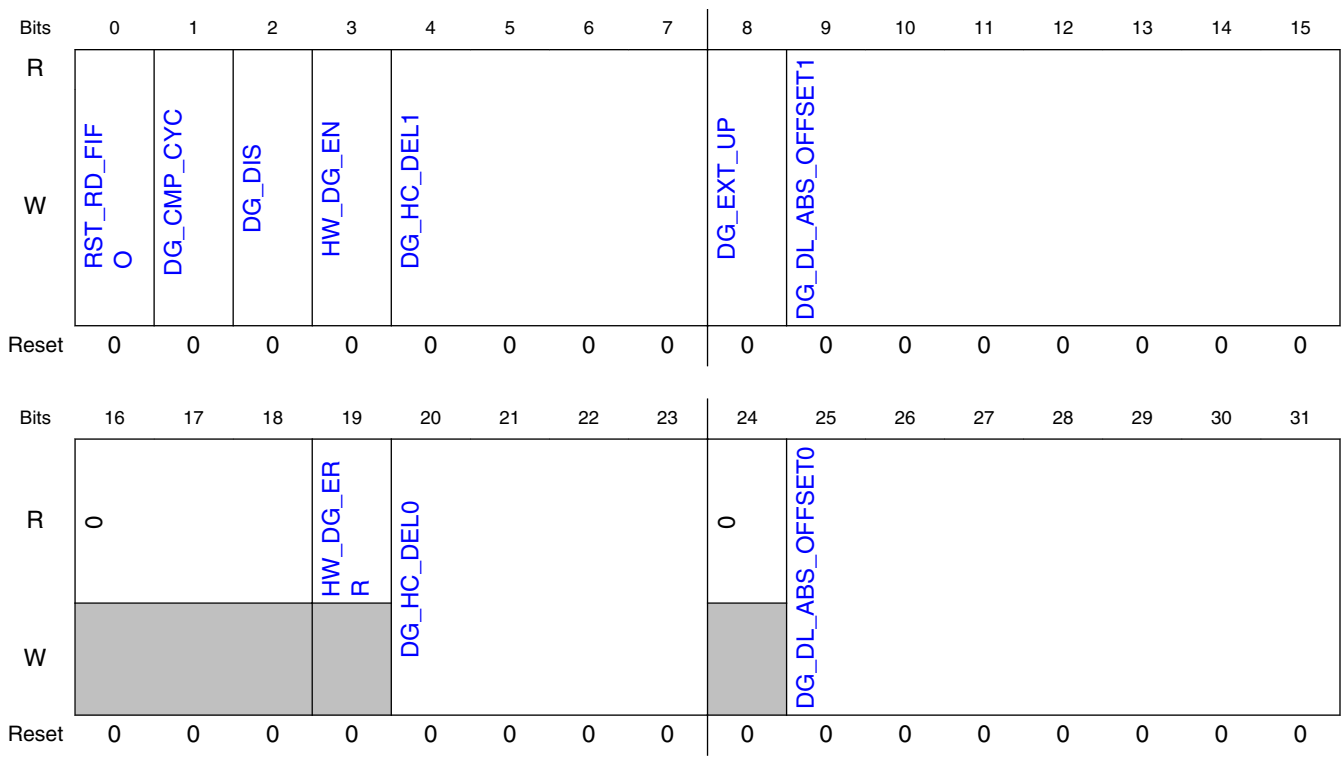
Field	Function
wr_dq8_del	Write dq8 delay fine-tuning. This field holds the number of delay units that are added to dq8 relative to dqs1. 00b - No change in dq8 delay 01b - Add dq8 delay of 1 delay unit. 10b - Add dq8 delay of 2 delay units. 11b - Add dq8 delay of 3 delay units.

23.9.1.38 MMDC PHY Read DQS Gating Control Register 0 (MPDG CTRL0)

23.9.1.38.1 Offset

Register	Offset
MPDGCTRL0	83Ch

23.9.1.38.2 Diagram



23.9.1.38.3 Fields

Field	Function
0 RST_RD_FIFO	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.
1 DG_CMP_CYC	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. 0b - MMDC waits 16 DDR cycles 1b - MMDC waits 32 DDR cycles
2 DG_DIS	DG_DIS Read DQS gating disable. If this bit is asserted then the MMDC disables the read DQS gating mechanism. 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 0b - Read DQS gating mechanism is enabled 1b - Read DQS gating mechanism is disabled
3 HW_DG_EN	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. 0b - Disable automatic read DQS gating calibration 1b - Start automatic read DQS gating calibration
4-7 DG_HC_DEL1	DG_HC_DEL1 Read DQS gating half cycles delay for Byte1 . 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 generated 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). 0000b - 0 cycles delay. 0001b - Half cycle delay. 0010b - 1 cycle delay 1101b - 6.5 cycles delay 1110b - Reserved 1111b - Reserved
8 DG_EXT_UP	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 according to the last passing comparison.
9-15 DG_DL_ABS_OFFSET1	DG_DL_ABS_OFFSET1

Table continues on the next page...

Field	Function
	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 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.
16-18 —	- Reserved
19 HW_DG_ERR	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. 0b - No error was found during the DQS gating HW calibration process. 1b - An error was found during the DQS gating HW calibration process.
20-23 DG_HC_DEL0	DG_HC_DEL0 Read DQS gating half cycles delay for Byte0 . 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 generated 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). 0000b - 0 cycles delay. 0001b - Half cycle delay. 0010b - 1 cycle delay 1101b - 6.5 cycles delay 1110b - Reserved 1111b - Reserved
24 —	- Reserved
25-31 DG_DL_ABS_OFFSET0	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 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.

23.9.1.39 MMDC PHY Read DQS Gating delay-line Status Register (MPDGDLST0)

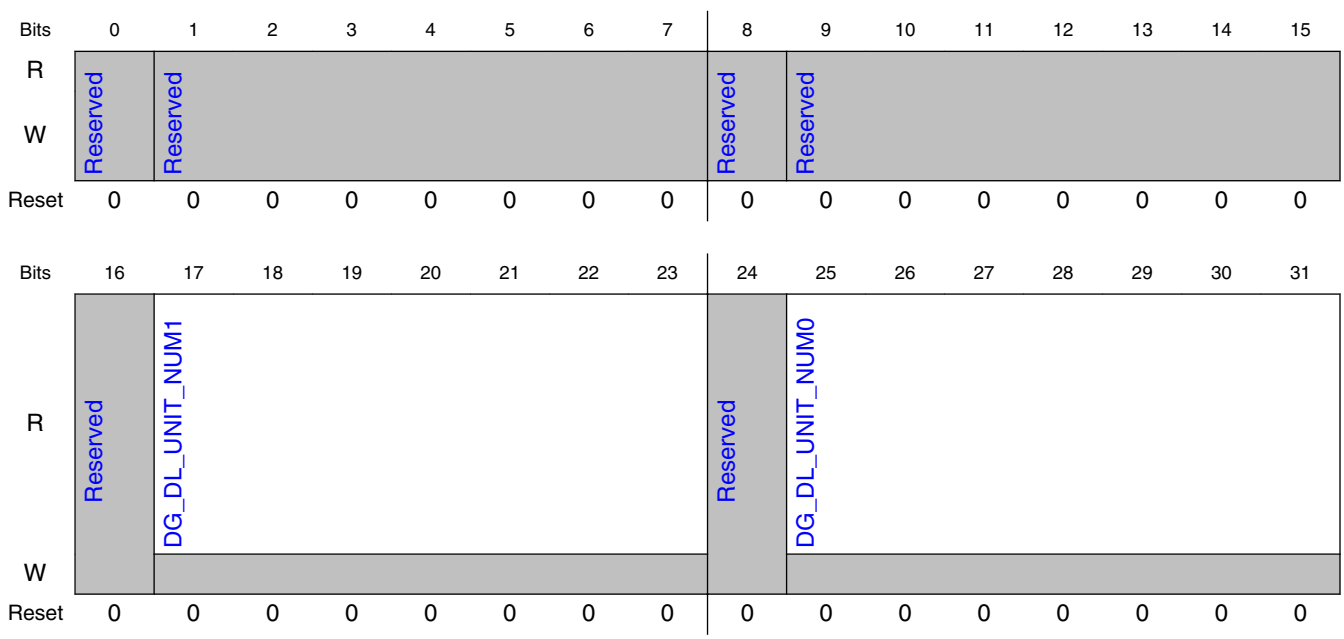
23.9.1.39.1 Offset

Register	Offset
MPDGDLST0	844h

23.9.1.39.2 Function

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

23.9.1.39.3 Diagram



23.9.1.39.4 Fields

Field	Function
0 —	- Reserved
1-7 —	- Reserved
8	-

Table continues on the next page...

Field	Function
—	Reserved
9-15 —	- Reserved
16 —	- Reserved
17-23 DG_DL_UNIT_NUM1	DG_DL_UNIT_NUM1 This field reflects the number of delay units that are actually used by read DQS gating delay-line 1.
24 —	- Reserved
25-31 DG_DL_UNIT_NUM0	DG_DL_UNIT_NUM0 This field reflects the number of delay units that are actually used by read DQS gating delay-line 0.

23.9.1.40 MMDC PHY Read delay-lines Configuration Register (MPRDDLCTL)

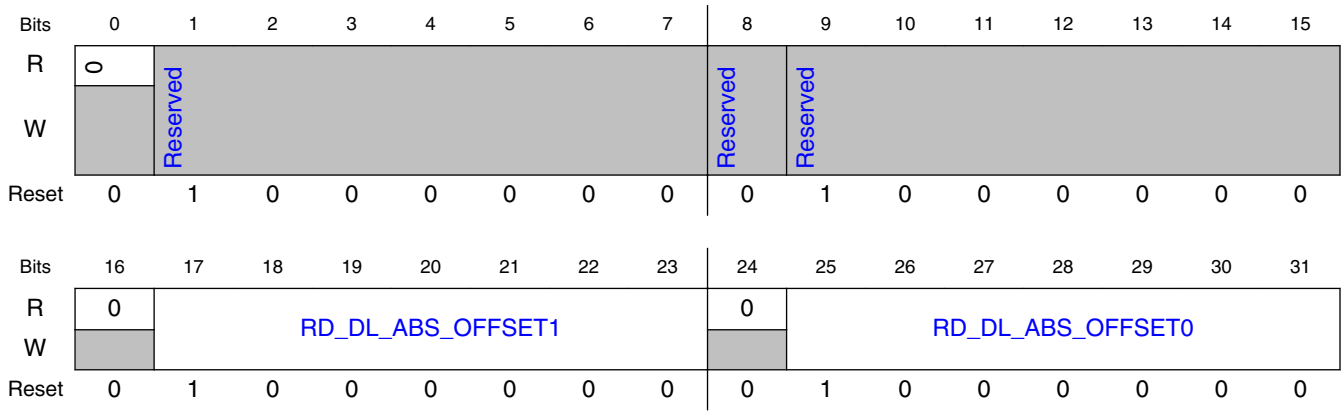
23.9.1.40.1 Offset

Register	Offset
MPRDDLCTL	848h

23.9.1.40.2 Function

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.

23.9.1.40.3 Diagram



23.9.1.40.4 Fields

Field	Function
0 —	- Reserved
1-7 —	- Reserved
8 —	- Reserved
9-15 —	- Reserved
16 —	- Reserved
17-23 RD_DL_ABS_OFFSET1	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 AXI clock (fast clock). So for the default value of 64, a quarter cycle delay is found. 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.
24 —	- Reserved
25-31 RD_DL_ABS_OFFSET0	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 AXI clock (fast clock). So for the default value of 64, a quarter cycle delay is found.

Field	Function
	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.

23.9.1.41 MMDC PHY Read delay-lines Status Register (MPRDDLST)

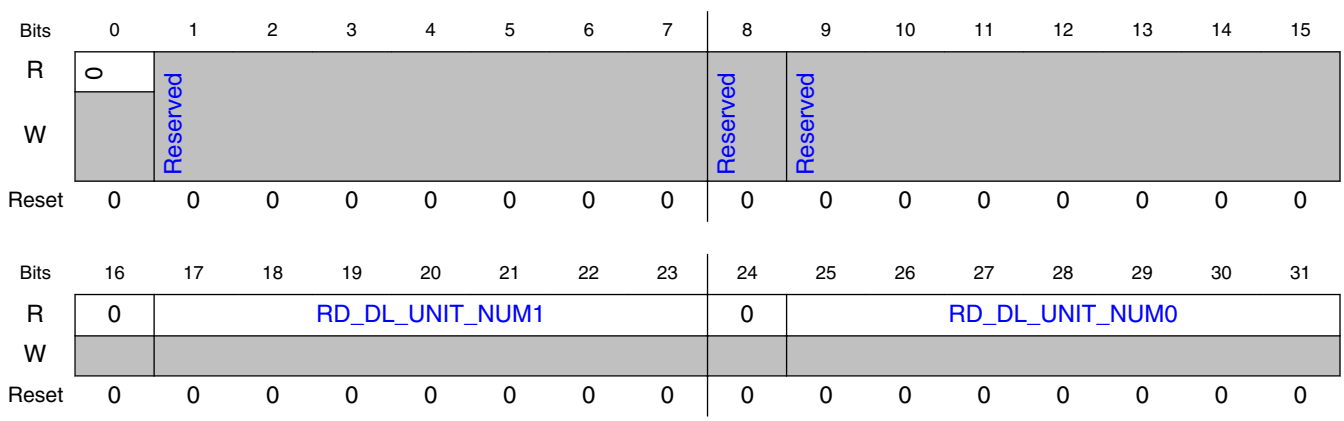
23.9.1.41.1 Offset

Register	Offset
MPRDDLST	84Ch

23.9.1.41.2 Function

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

23.9.1.41.3 Diagram



23.9.1.41.4 Fields

Field	Function
0 —	- Reserved
1-7	-

Table continues on the next page...

Field	Function
—	Reserved
8 —	- Reserved
9-15 —	- Reserved
16 —	- Reserved
17-23 RD_DL_UNIT_NUM1	RD_DL_UNIT_NUM1 This field reflects the number of delay units that are actually used by read delay-line 1.
24 —	- Reserved
25-31 RD_DL_UNIT_NUM0	RD_DL_UNIT_NUM0 This field reflects the number of delay units that are actually used by read delay-line 0.

23.9.1.42 MMDC PHY Write delay-lines Configuration Register (MPWRDLCTL)

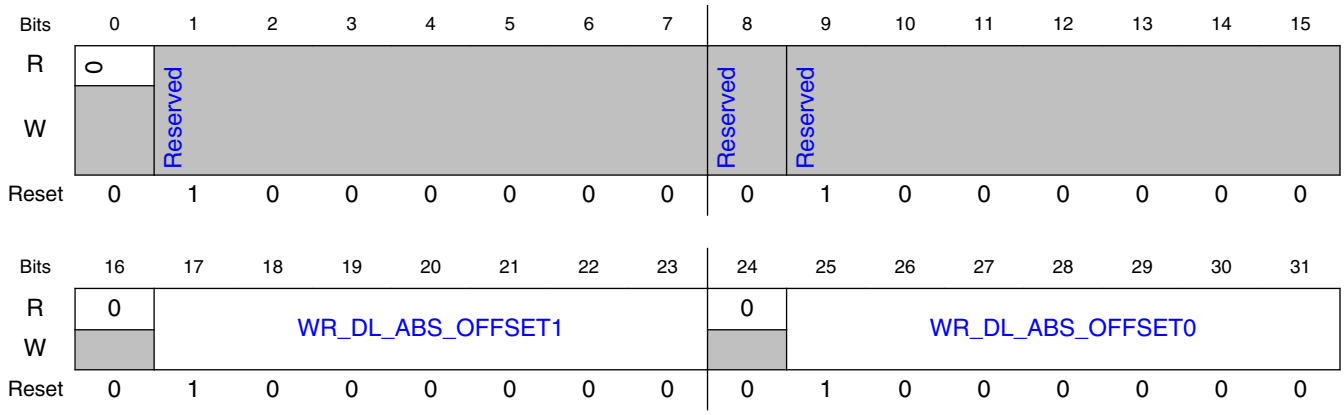
23.9.1.42.1 Offset

Register	Offset
MPWRDLCTL	850h

23.9.1.42.2 Function

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.

23.9.1.42.3 Diagram



23.9.1.42.4 Fields

Field	Function
0 —	- Reserved
1-7 —	- Reserved
8 —	- Reserved
9-15 —	- Reserved
16 —	- Reserved
17-23 WR_DL_ABS_OFFSET1	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 AXI clock (fast clock). So for the default value of 64, a quarter cycle delay is found. 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.
24 —	- Reserved
25-31 WR_DL_ABS_OFFSET0	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 frequency independent. The delay of the delay-line would be (WR_DL_ABS_OFFSET0 / 256) * MMDC AXI clock (fast clock). So for the default value of 64, a quarter cycle delay is found.

Field	Function
	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.

23.9.1.43 MMDC PHY Write delay-lines Status Register (MPWRDLST)

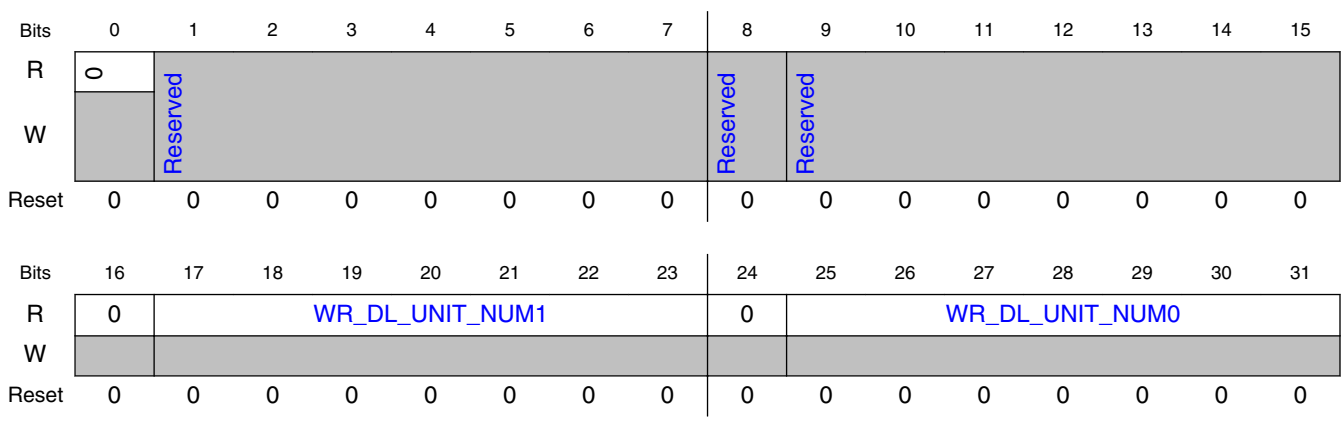
23.9.1.43.1 Offset

Register	Offset
MPWRDLST	854h

23.9.1.43.2 Function

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

23.9.1.43.3 Diagram



23.9.1.43.4 Fields

Field	Function
0 —	- Reserved
1-7	-

Table continues on the next page...

Field	Function
—	Reserved
8 —	- Reserved
9-15 —	- Reserved
16 —	- Reserved
17-23 WR_DL_UNIT_NUM1	WR_DL_UNIT_NUM1 This field reflects the number of delay units that are actually used by write delay-line 1.
24 —	- Reserved
25-31 WR_DL_UNIT_NUM0	WR_DL_UNIT_NUM0 This field reflects the number of delay units that are actually used by write delay-line 0.

23.9.1.44 MMDC PHY CK Control Register (MPSDCTRL)

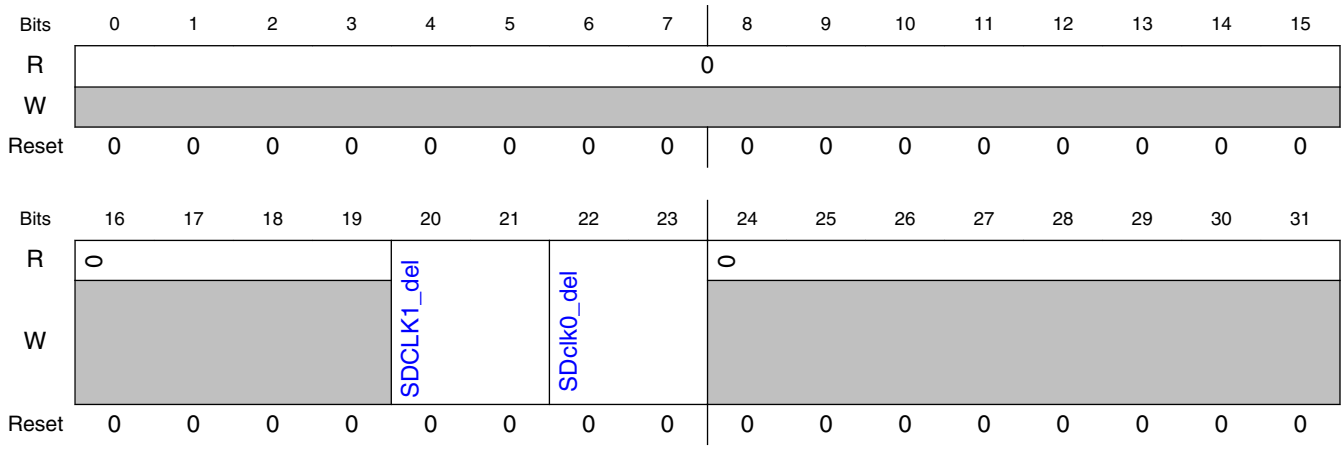
23.9.1.44.1 Offset

Register	Offset
MPSDCTRL	858h

23.9.1.44.2 Function

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

23.9.1.44.3 Diagram



23.9.1.44.4 Fields

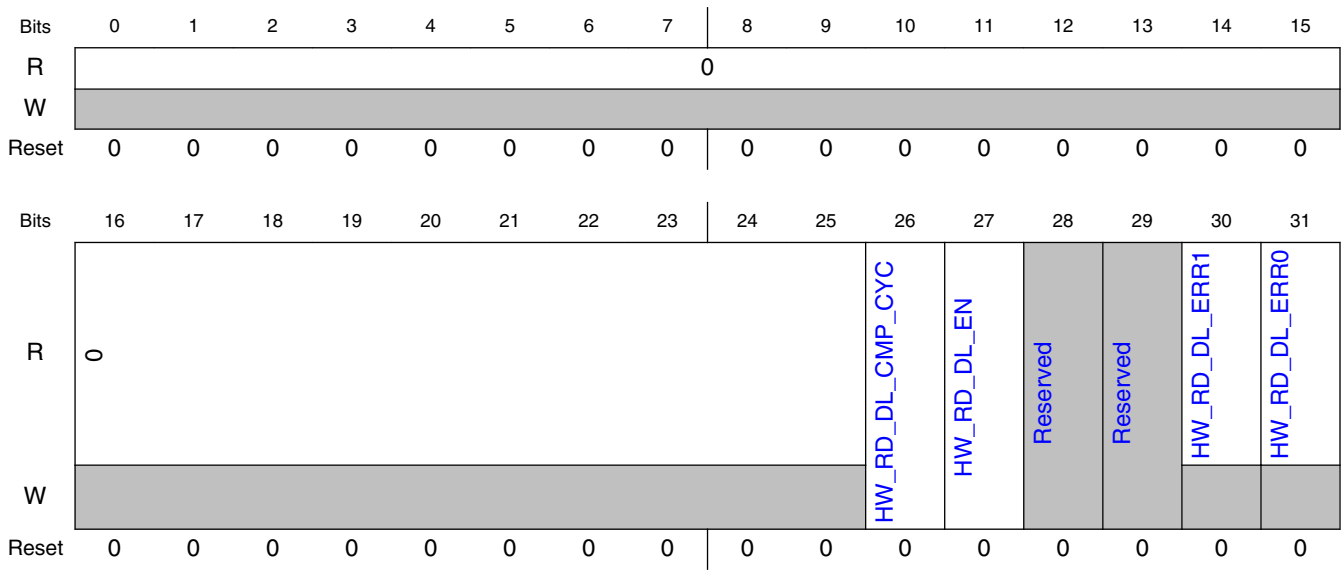
Field	Function
0-19 —	- Reserved
20-21 SDCLK1_del	SDCLK1_del DDR clock1 delay fine tuning. This field holds the number of delay units that are added to DDR clock1 (CK1). 00b - No change in DDR clock delay 01b - Add DDR clock delay of 1 delay unit. 10b - Add DDR clock delay of 2 delay units. 11b - Add DDR clock delay of 3 delay units.
22-23 SDclk0_del	SDclk0_del DDR clock0 delay fine tuning. This field holds the number of delay units that are added to DDR clock (CK0). 00b - No change in DDR clock0 delay 01b - Add DDR clock0 delay of 1 delay unit. 10b - Add DDR clock0 delay of 2 delay units. 11b - Add DDR clock0 delay of 3 delay units.
24-31 —	- Reserved

23.9.1.45 MMDC PHY Read Delay HW Calibration Control Register (MPRDDLHWCTL)

23.9.1.45.1 Offset

Register	Offset
MPRDDLHWCTL	860h

23.9.1.45.2 Diagram



23.9.1.45.3 Fields

Field	Function
0-25 —	- Reserved
26 HW_RD_DL_CMP_CYC	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.
27 HW_RD_DL_EN	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.
28 —	- Reserved
29 —	- Reserved
30	HW_RD_DL_ERR1

Table continues on the next page...

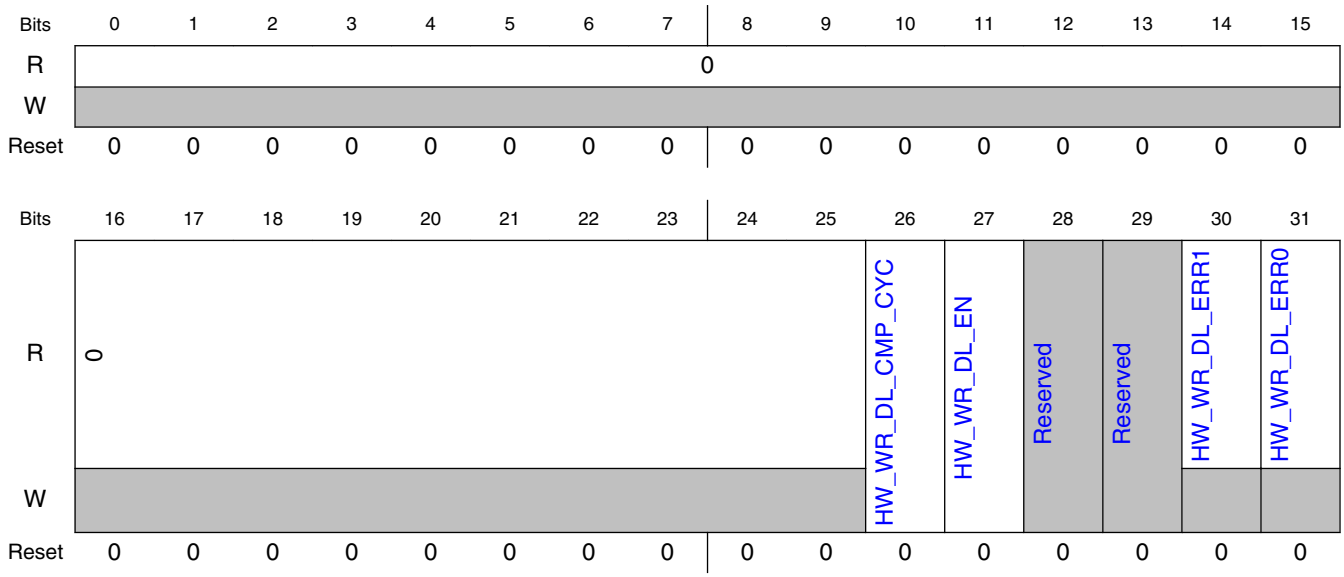
Field	Function
HW_RD_DL_ERR1	Automatic (HW) read calibration error of Byte1. If this bit is asserted then it indicates that an error was found during the HW calibration process of read delay-line 1. In case this bit is zero at the end of the calibration process then the boundary results can be found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

23.9.1.46 MMDC PHY Write Delay HW Calibration Control Register (MPWRDLHWCTL)

23.9.1.46.1 Offset

Register	Offset
MPWRDLHWCTL	864h

23.9.1.46.2 Diagram



23.9.1.46.3 Fields

Field	Function
0-25 —	- Reserved
26 HW_WR_DL_CMP_CYC	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.
27 HW_WR_DL_EN	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.
28 —	- Reserved
29 —	- Reserved
30 HW_WR_DL_ERR1	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. 0b - No error was found during the automatic (HW) write calibration process of write delay-line 1. 1b - An error was found during the automatic (HW) write calibration process of write delay-line 1.
31	HW_WR_DL_ERR0

Field	Function
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 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. 0b - No error was found during the automatic (HW) write calibration process of write delay-line 0. 1b - An error was found during the automatic (HW) write calibration process of write delay-line 0.

23.9.1.47 MMDC PHY Read Delay HW Calibration Status Register 0 (MPRDDLHWST0)

23.9.1.47.1 Offset

Register	Offset
MPRDDLHWST0	868h

23.9.1.47.2 Diagram

Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
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

Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
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

23.9.1.47.3 Fields

Field	Function
0 —	- Reserved
1-7 HW_RD_DL_UP1	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
8 —	- Reserved

Table continues on the next page...

Field	Function
9-15 HW_RD_DL_LOW1	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
16 —	- Reserved
17-23 HW_RD_DL_UP0	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.
24 —	- Reserved
25-31 HW_RD_DL_LOW0	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.

23.9.1.48 MMDC PHY Write Delay HW Calibration Status Register 0 (MPWRDLHWST0)

23.9.1.48.1 Offset

Register	Offset
MPWRDLHWST0	870h

23.9.1.48.2 Diagram

Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
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

Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
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

23.9.1.48.3 Fields

Field	Function
0 —	- Reserved
1-7 HW_WR_DL_UP1	HW_WR_DL_UP1 Automatic (HW) write automatic (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.
8 —	- Reserved
9-15 HW_WR_DL_LOW1	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.
16 —	- Reserved
17-23 HW_WR_DL_UP0	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.
24 —	- Reserved
25-31 HW_WR_DL_LOW0	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.

23.9.1.49 MMDC PHY Write Leveling HW Error Register (MPWLHWERR)

23.9.1.49.1 Offset

Register	Offset
MPWLHWERR	878h

23.9.1.49.2 Diagram

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

Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	HW_WL1_DQ								HW_WL0_DQ							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

23.9.1.49.3 Fields

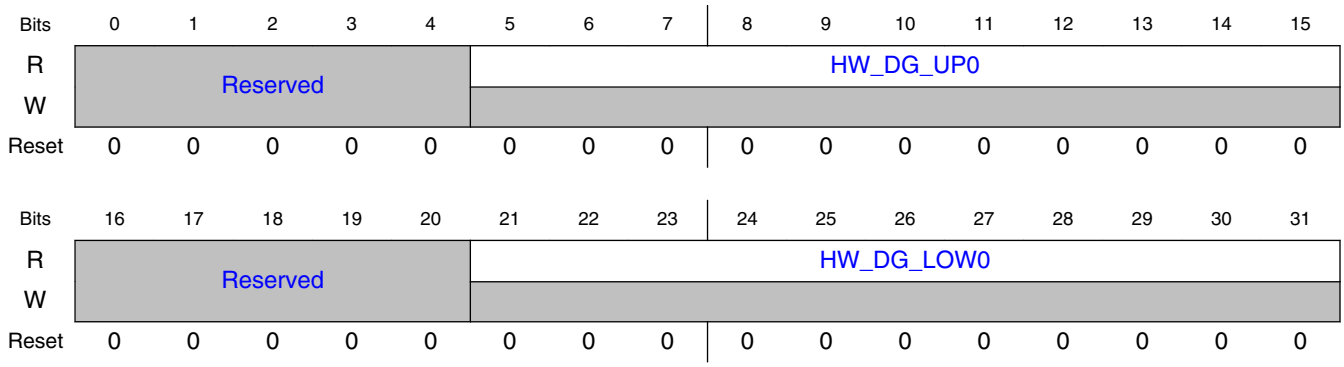
Field	Function
0-7 —	- Reserved
8-15 —	- Reserved
16-23 HW_WL1_DQ	HW_WL1_DQ HW write-leveling calibration result of Byte1. This field holds the results for all the 8 write-leveling steps of Byte1. i.e. bit 0 holds the result of the write-leveling calibration of 0 delay, bit 1holds the result of the write-leveling calibration of 1/8delay till bit 7 that holds the result of the write-leveling calibration of 7/8 delay
24-31 HW_WL0_DQ	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

23.9.1.50 MMDC PHY Read DQS Gating HW Status Register 0 (MPDG HWST0)

23.9.1.50.1 Offset

Register	Offset
MPDGHWST0	87Ch

23.9.1.50.2 Diagram



23.9.1.50.3 Fields

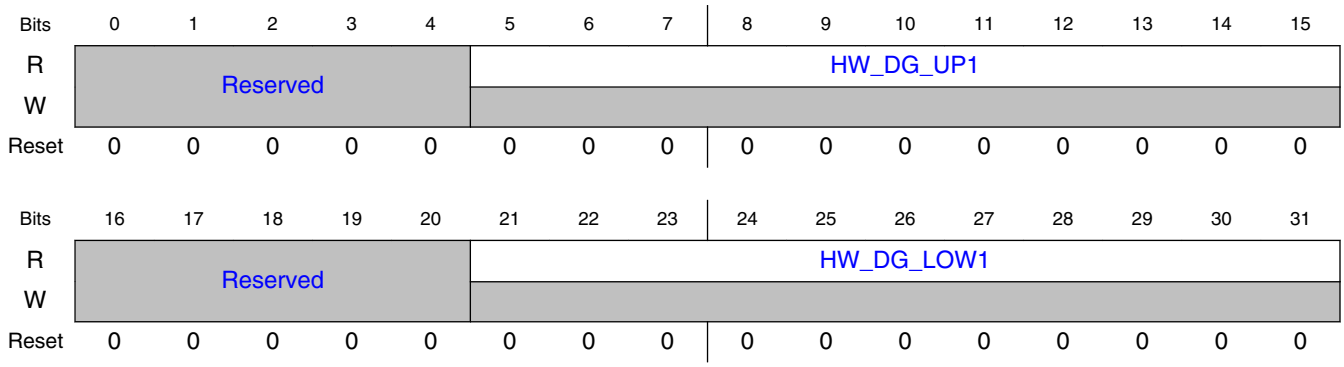
Field	Function
0-4 —	- Reserved
5-15 HW_DG_UP0	HW_DG_UP0 HW DQS gating calibration result of the upper boundary of Byte0. This field holds the HW DQS gating calibration result of the upper boundary of Byte0.
16-20 —	- Reserved
21-31 HW_DG_LOW0	HW_DG_LOW0 HW DQS gating calibration result of the lower boundary of Byte0. This field holds the HW DQS gating calibration result of the lower boundary of Byte0.

23.9.1.51 MMDC PHY Read DQS Gating HW Status Register 1 (MPDGHWST1)

23.9.1.51.1 Offset

Register	Offset
MPDGHWST1	880h

23.9.1.51.2 Diagram



23.9.1.51.3 Fields

Field	Function
0-4 —	- Reserved
5-15 HW_DG_UP1	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.
16-20 —	- Reserved
21-31 HW_DG_LOW1	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.

23.9.1.52 MMDC PHY Pre-defined Compare Register 1 (MPPDCMPR1)

23.9.1.52.1 Offset

Register	Offset
MPPDCMPR1	88Ch

23.9.1.52.2 Function

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 DDR3L (BL=8) the MMDC will duplicate PDV1,PDV2 and drive that data on Beat4-7 of the same byte

23.9.1.52.3 Diagram

Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	PDV2															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	PDV1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

23.9.1.52.4 Fields

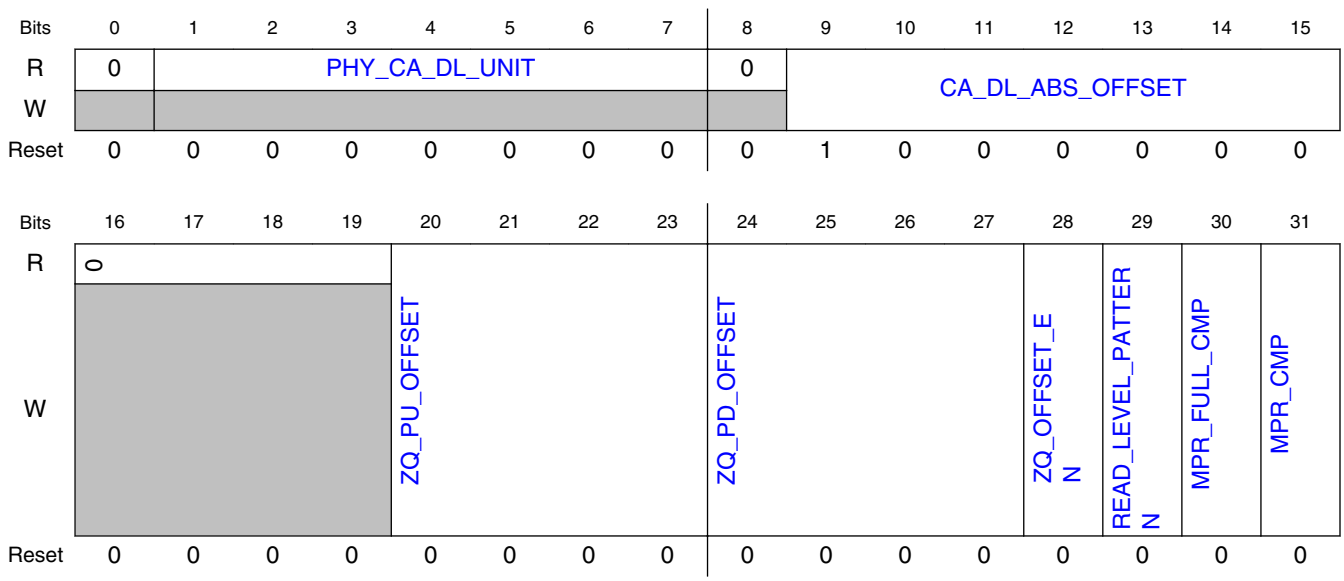
Field	Function
0-15 PDV2	PDV2 MMDC Pre defined compare value2. This field holds the 2 MSB of the data that will be driven to the DDR device during automatic read, read DQS gating and write calibrations in case MPR(DDR3L) mode are disabled (MPR_CMP is disabled). Upon read access during the calibration 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 Calibration
16-31 PDV1	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(DDR3L) 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 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.

23.9.1.53 MMDC PHY Pre-defined Compare and CA delay-line Configuration Register (MPPDCMPR2)

23.9.1.53.1 Offset

Register	Offset
MPPDCMPR2	890h

23.9.1.53.2 Diagram



23.9.1.53.3 Fields

Field	Function
0 —	- Reserved
1-7 PHY_CA_DL_UNIT	PHY_CA_DL_UNIT This field reflects the number of delay units that are actually used by CA(Command/Address) delay-line
8 —	- Reserved
9-15 CA_DL_ABS_OFFSET	CA_DL_ABS_OFFSET

Table continues on the next page...

Field	Function
	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 AXI clock (fast clock). So for the default value of 64, a quarter cycle delay is found.
16-19 ——	- Reserved
20-23 ZQ_PU_OFFSET	ZQ_PU_OFFSET Programmable offset from -7 to 7 added to the MMDC_MPZQHWCTRL[ZQ_HW_PU_RES] field when ZQ_OFFSET_EN is enabled. Bit[11] determines direction: ‘b0 = Offset is added to ZQ_HW_PU_RES field ‘b1 = Offset is subtracted from ZQ_HW_PU_RES field Bits[10:8] = Amount of change to apply. 0000b - Offset is added to ZQ_HW_PU_RES field 0001b - Offset is added to ZQ_HW_PU_RES field 0010b - Offset is added to ZQ_HW_PU_RES field 0011b - Offset is added to ZQ_HW_PU_RES field 0100b - Offset is added to ZQ_HW_PU_RES field 0101b - Offset is added to ZQ_HW_PU_RES field 0110b - Offset is added to ZQ_HW_PU_RES field 0111b - Offset is added to ZQ_HW_PU_RES field 1000b - Offset is subtracted from ZQ_HW_PU_RES field 1001b - Offset is subtracted from ZQ_HW_PU_RES field 1010b - Offset is subtracted from ZQ_HW_PU_RES field 1011b - Offset is subtracted from ZQ_HW_PU_RES field 1100b - Offset is subtracted from ZQ_HW_PU_RES field 1101b - Offset is subtracted from ZQ_HW_PU_RES field 1110b - Offset is subtracted from ZQ_HW_PU_RES field 1111b - Offset is subtracted from ZQ_HW_PU_RES field
24-27 ZQ_PD_OFFSET	ZQ_PD_OFFSET Programmable offset from -7 to 7 added to the MMDC_MPZQHWCTRL[ZQ_HW_PD_RES] field when ZQ_OFFSET_EN is enabled. Bit[7] determines direction: ‘b0 = Offset is added to ZQ_HW_PD_RES field ‘b1 = Offset is subtracted from ZQ_HW_PD_RES field Bits[6:4] = Amount of change to apply. 0000b - Offset is added to ZQ_HW_PD_RES field 0001b - Offset is added to ZQ_HW_PD_RES field 0010b - Offset is added to ZQ_HW_PD_RES field 0011b - Offset is added to ZQ_HW_PD_RES field 0100b - Offset is added to ZQ_HW_PD_RES field 0101b - Offset is added to ZQ_HW_PD_RES field 0110b - Offset is added to ZQ_HW_PD_RES field 0111b - Offset is added to ZQ_HW_PD_RES field 1000b - Offset is subtracted from ZQ_HW_PD_RES field 1001b - Offset is subtracted from ZQ_HW_PD_RES field 1010b - Offset is subtracted from ZQ_HW_PD_RES field 1011b - Offset is subtracted from ZQ_HW_PD_RES field 1100b - Offset is subtracted from ZQ_HW_PD_RES field

Table continues on the next page...

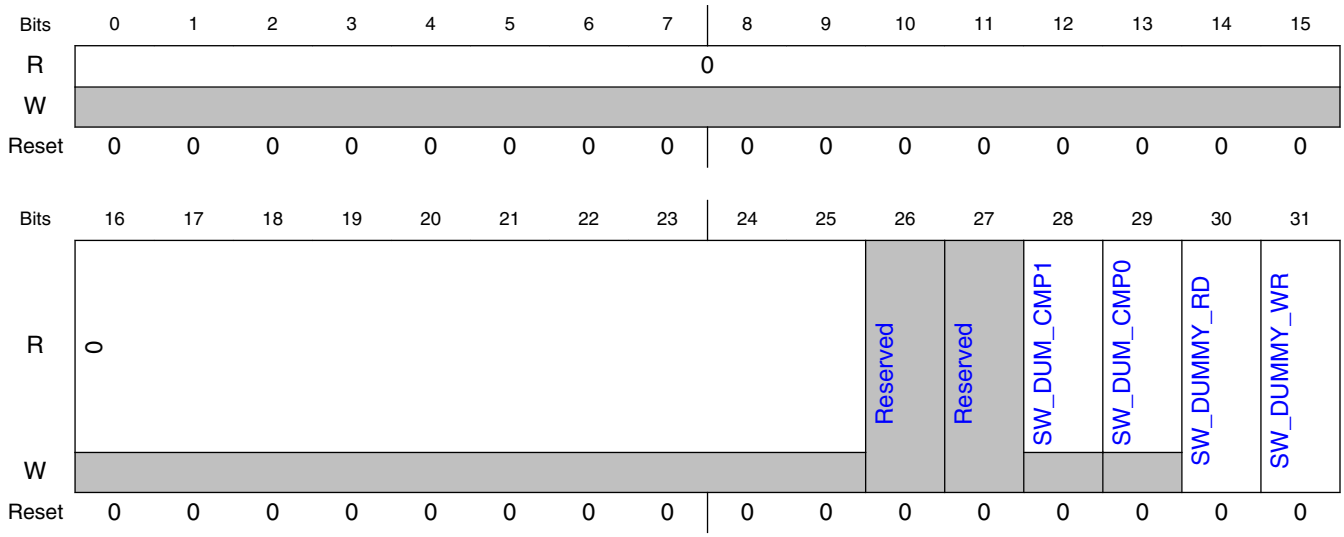
Field	Function
	1101b - Offset is subtracted from ZQ_HW_PD_RES field 1110b - Offset is subtracted from ZQ_HW_PD_RES field 1111b - Offset is subtracted from ZQ_HW_PD_RES field
28 ZQ_OFFSET_EN	ZQ_OFFSET_EN 0b - Hardware ZQ offset disabled 1b - Hardware ZQ offset enabled
29 READ_LEVEL_PATTERN	READ_LEVEL_PATTERN MPR(DDR3L) read compare pattern. In case MPR(DDR3L) modes are used during the calibration process (MPR_CMP is asserted) then this field indicates the read pattern for the comparison. 0b - Compare with read pattern 1010 1b - Reserved
30 MPR_FULL_CMP	MPR_FULL_CMP MPR(DDR3L) full compare enable. In case MPR(DDR3L) 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.
31 MPR_CMP	MPR_CMP MPR(DDR3L) 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 device (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.

23.9.1.54 MMDC PHY SW Dummy Access Register (MPSWDAR0)

23.9.1.54.1 Offset

Register	Offset
MPSWDAR0	894h

23.9.1.54.2 Diagram



23.9.1.54.3 Fields

Field	Function
0-25 —	- Reserved
26 —	- Reserved
27 —	- Reserved
28 SW_DUM_CMP1	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. 0b - Dummy read fail 1b - Dummy read pass
29 SW_DUM_CMP0	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. 0b - Dummy read fail 1b - Dummy read pass
30 SW_DUMMY_RD	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.
31	SW_DUMMY_WR

Field	Function
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.

23.9.1.55 MMDC PHY SW Dummy Read Data Register 0 (MPSWDRDR0)

23.9.1.55.1 Offset

Register	Offset
MPSWDRDR0	898h

23.9.1.55.2 Diagram

Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DUM_RD0															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DUM_RD0															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

23.9.1.55.3 Fields

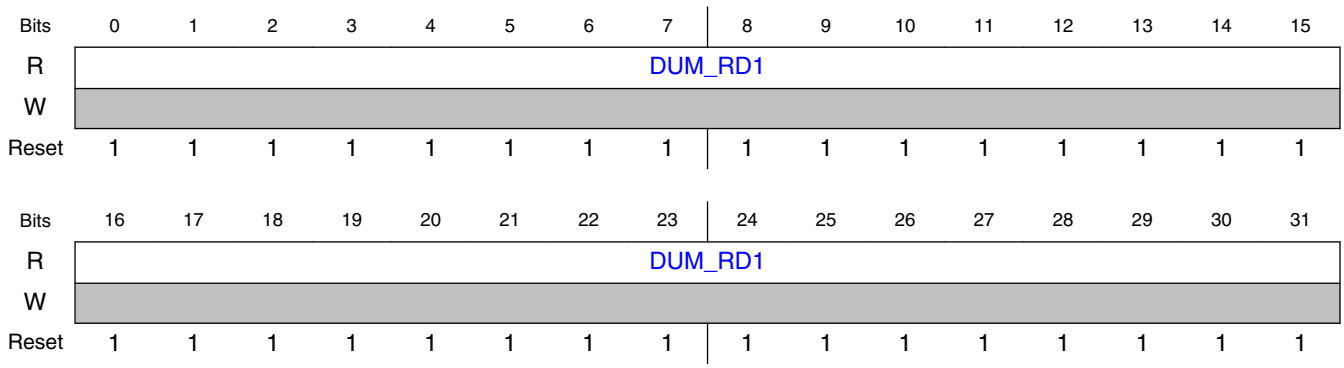
Field	Function
0-31 DUM_RD0	DUM_RD0 Dummy read data0. This field holds the first data that is read from the DDR during SW dummy read access (i.e. when SW_DUMMY_RD = 1). This field is valid only when SW_DUMMY_RD is de-assrted

23.9.1.56 MMDC PHY SW Dummy Read Data Register 1 (MPSWDRDR 1)

23.9.1.56.1 Offset

Register	Offset
MPSWDRDR1	89Ch

23.9.1.56.2 Diagram



23.9.1.56.3 Fields

Field	Function
0-31 DUM_RD1	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

23.9.1.57 MMDC PHY SW Dummy Read Data Register 2 (MPSWDRDR 2)

23.9.1.57.1 Offset

Register	Offset
MPSWDRDR2	8A0h

23.9.1.57.2 Diagram

Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DUM_RD2															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DUM_RD2															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

23.9.1.57.3 Fields

Field	Function
0-31 DUM_RD2	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.

23.9.1.58 MMDC PHY SW Dummy Read Data Register 3 (MPSWDRDR 3)

23.9.1.58.1 Offset

Register	Offset
MPSWDRDR3	8A4h

23.9.1.58.2 Diagram

Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DUM_RD3															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DUM_RD3															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

23.9.1.58.3 Fields

Field	Function
0-31 DUM_RD3	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.

23.9.1.59 MMDC PHY SW Dummy Read Data Register 4 (MPSWDRDR 4)

23.9.1.59.1 Offset

Register	Offset
MPSWDRDR4	8A8h

23.9.1.59.2 Diagram

Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	DUM_RD4															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	DUM_RD4															
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

23.9.1.59.3 Fields

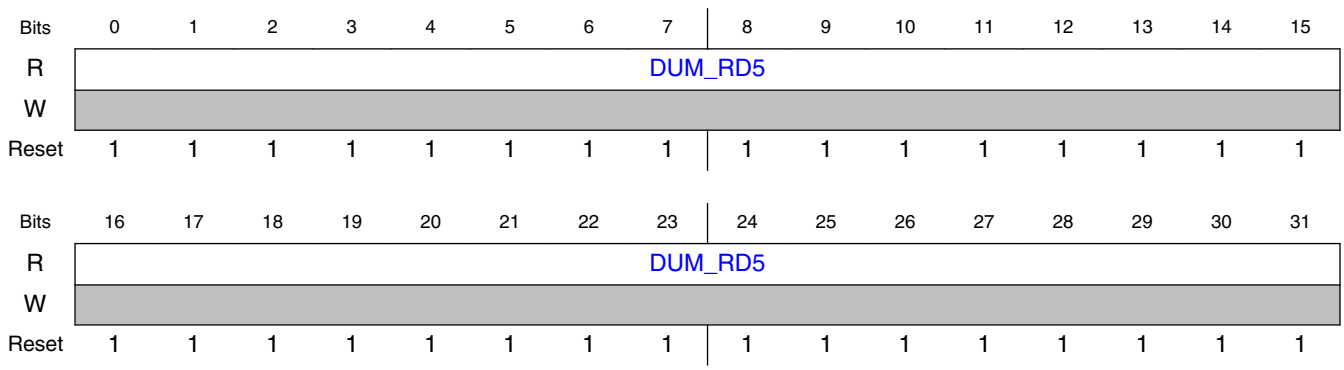
Field	Function
0-31 DUM_RD4	DUM_RD4 Dummy read data4. This field holds the fifth data (only in case of burst length 8 (BL =1)) that is read from the DDR during SW dummy read access (i.e. when SW_DUMMY_RD = 1). This field is valid only when SW_DUMMY_RD is de-assrted.

23.9.1.60 MMDC PHY SW Dummy Read Data Register 5 (MPSWDRDR5)

23.9.1.60.1 Offset

Register	Offset
MPSWDRDR5	8ACh

23.9.1.60.2 Diagram



23.9.1.60.3 Fields

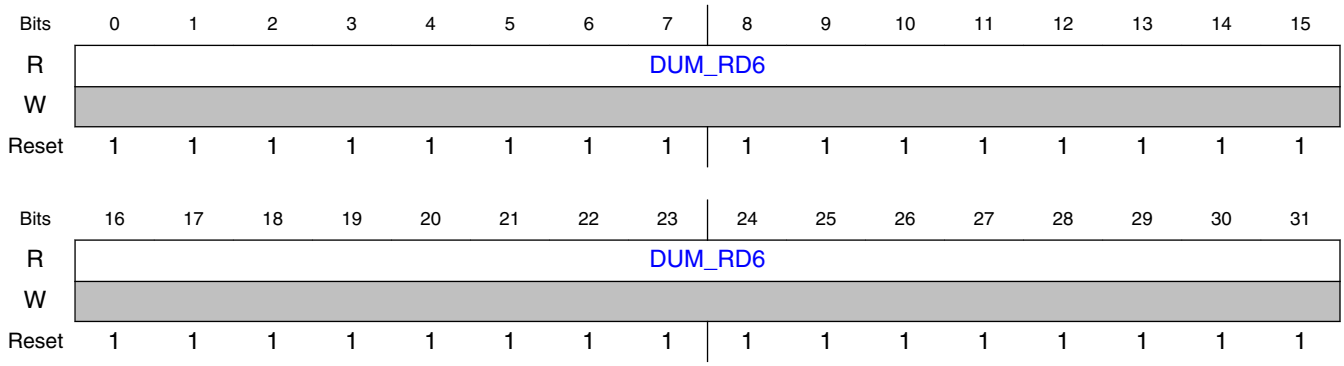
Field	Function
0-31 DUM_RD5	DUM_RD5 Dummy read data5. This field holds the sixth data (only in case of burst length 8 (BL =1)) that is read from the DDR during SW dummy read access (i.e. when SW_DUMMY_RD = 1). This field is valid only when SW_DUMMY_RD is de-assrted.

23.9.1.61 MMDC PHY SW Dummy Read Data Register 6 (MPSWDRDR6)

23.9.1.61.1 Offset

Register	Offset
MPSWDRDR6	8B0h

23.9.1.61.2 Diagram



23.9.1.61.3 Fields

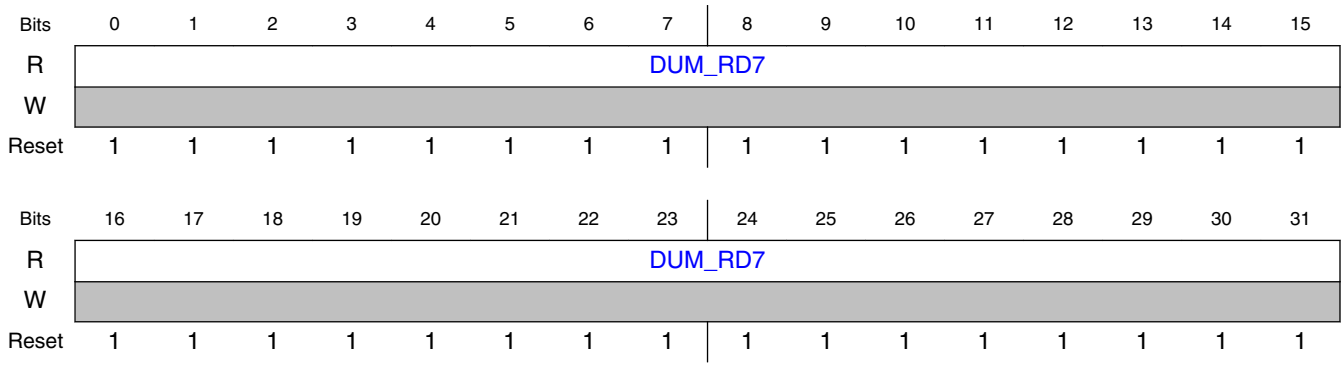
Field	Function
0-31 DUM_RD6	DUM_RD6 Dummy read data6. This field holds the seventh data (only in case of burst length 8 (BL =1)) that is read from the DDR during SW dummy read access (i.e. when SW_DUMMY_RD = 1). This field is valid only when SW_DUMMY_RD is de-assrted.

23.9.1.62 MMDC PHY SW Dummy Read Data Register 7 (MPSWDRDR 7)

23.9.1.62.1 Offset

Register	Offset
MPSWDRDR7	8B4h

23.9.1.62.2 Diagram



23.9.1.62.3 Fields

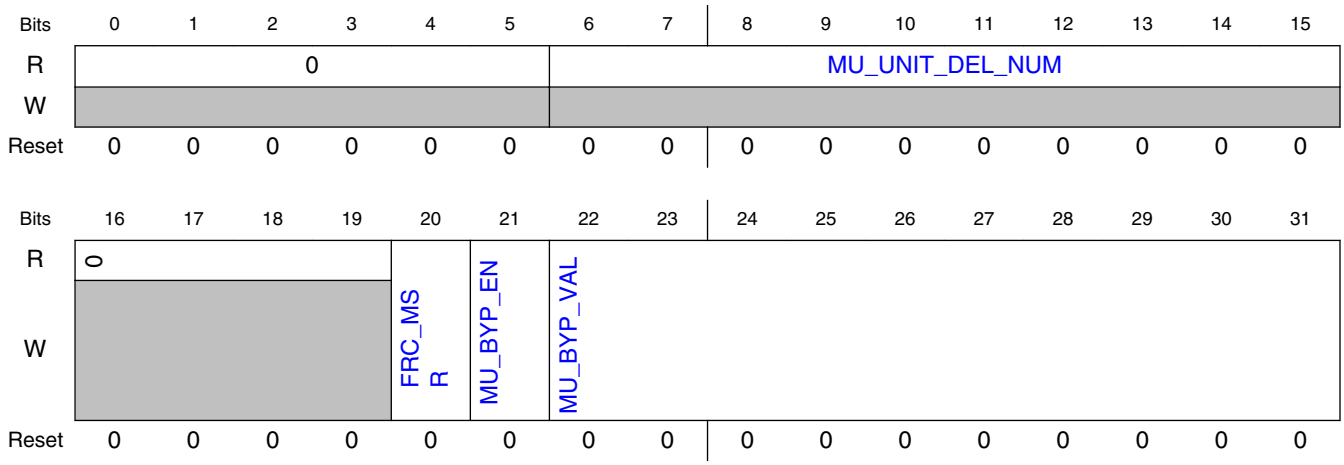
Field	Function
0-31 DUM_RD7	DUM_RD7 Dummy read data7. This field holds the eight data (only in case of burst length 8 (BL =1)) that is read from the DDR during SW dummy read access (i.e. when SW_DUMMY_RD = 1). This field is valid only when SW_DUMMY_RD is de-assrted.

23.9.1.63 MMDC PHY Measure Unit Register (MPMUR0)

23.9.1.63.1 Offset

Register	Offset
MPMUR0	8B8h

23.9.1.63.2 Diagram



23.9.1.63.3 Fields

Field	Function
0-5 —	- Reserved
6-15 MU_UNIT_DEL_NUM	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.
16-19 —	- Reserved
20 FRC_MSR	FRC_MSR Force measurement 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. 0b - No measurement is performed 1b - Perform measurement process
21 MU_BYP_EN	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 0b - The delay-lines use delay units as indicated at MU_UNIT_DEL_NUM. 1b - The delay-lines use delay units as indicated at MU_BYPASS_VAL.
22-31	MU_BYP_VAL

Field	Function
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.

23.9.1.64 MMDC Duty Cycle Control Register (MPDCCR)

23.9.1.64.1 Offset

Register	Offset
MPDCCR	8C0h

23.9.1.64.2 Function

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

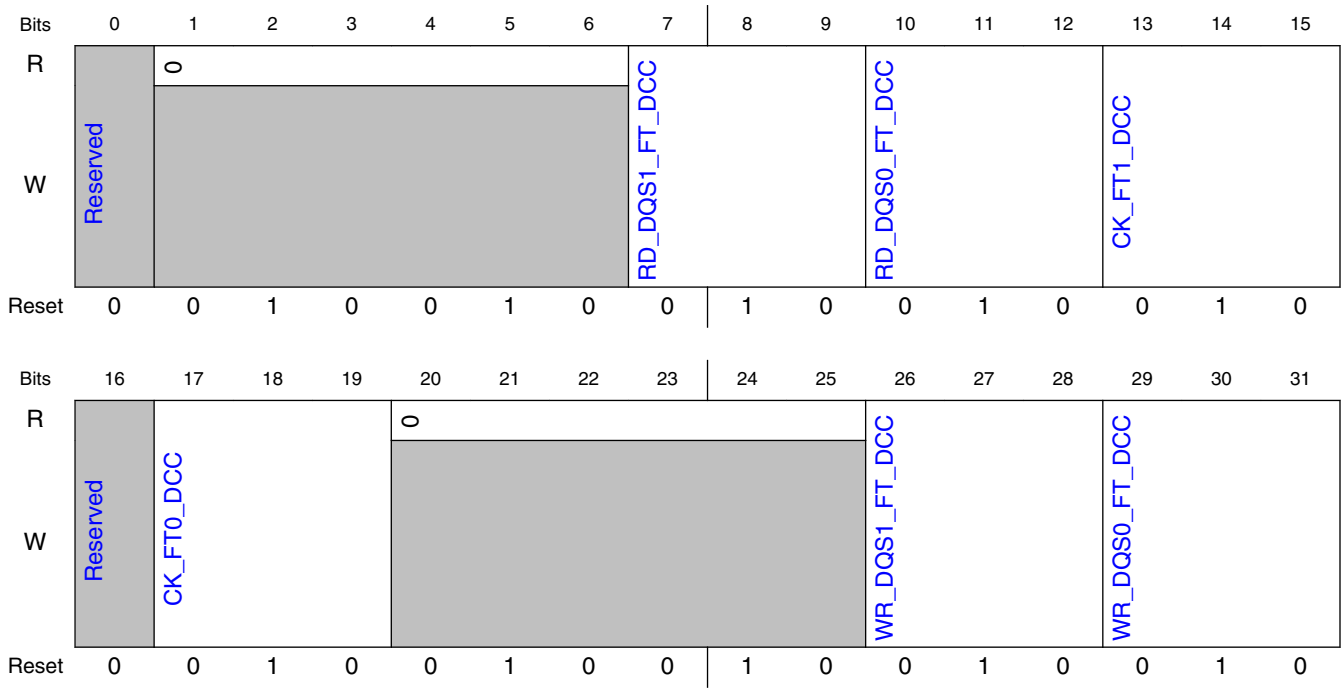
NOTE

If the duty cycle is modified after DDR initialization, the DDR will have to be placed in self-refresh mode.

NOTE

The duty cycle may be changed during initial DDR initialization without having to be placed in self-refresh mode.

23.9.1.64.3 Diagram



23.9.1.64.4 Fields

Field	Function
0 —	- reserved
1-6 —	Reserved.
7-9 RD_DQS1_FT_DCC	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 001b - 51.5% low 48.5% high 010b - 50% duty cycle (default) 100b - 48.5% low 51.5% high
10-12 RD_DQS0_FT_DCC	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 001b - 51.5% low 48.5% high 010b - 50% duty cycle (default) 100b - 48.5% low 51.5% high
13-15 CK_FT1_DCC	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

Table continues on the next page...

Field	Function
	Note all the other options are not allowed 001b - 48.5% low 51.5% high 010b - 50% duty cycle (default) 100b - 51.5% low 48.5% high
16 —	- Reserved
17-19 CK_FT0_DCC	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 001b - 48.5% low 51.5% high 010b - 50% duty cycle (default) 100b - 51.5% low 48.5% high
20-25 —	Reserved.
26-28 WR_DQS1_FT_DCC	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 001b - 51.5% low 48.5% high 010b - 50% duty cycle (default) 100b - 48.5% low 51.5% high
29-31 WR_DQS0_FT_DCC	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 001b - 51.5% low 48.5% high 010b - 50% duty cycle (default) 100b - 48.5% low 51.5% high

Chapter 24
Pre-Boot Loader (PBL)

24.1 Overview

The pre-boot loader (PBL) performs configuration register reads and writes to initialize the QuadSPI interface, loads the RCW and pre-boot initialization commands from external memory device through QuadSPI (QSPI_CS_A0) , and writes data to configuration registers or memory before the local cores are permitted to boot.

The following figure shows a block diagram of PBL, showing the functional organization of the module, which includes interface command modules (ICMs) for QuadSPI, and a common control module.

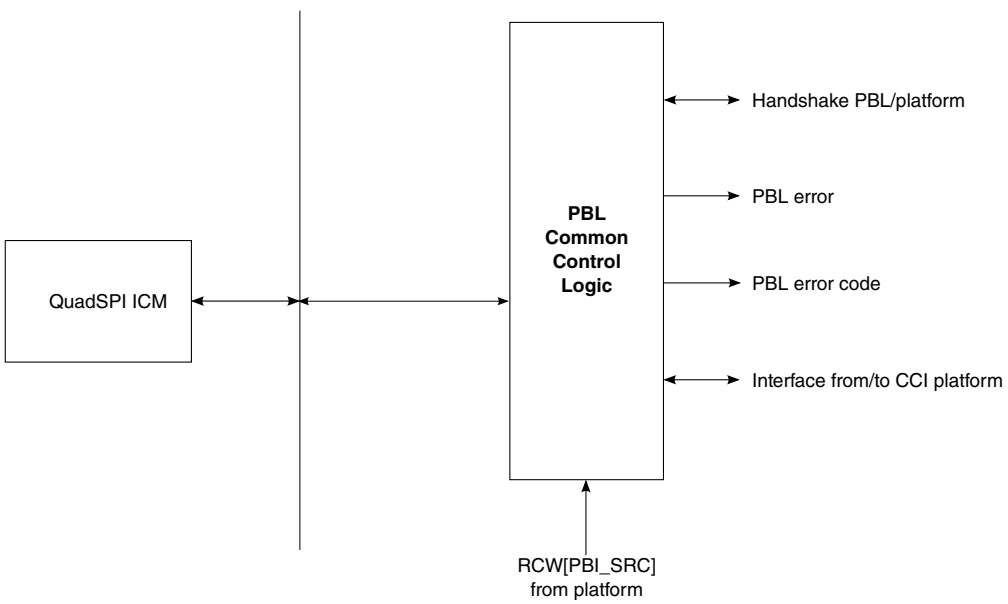


Figure 24-1. Pre-boot loader (PBL) block diagram

NOTE

There should not be any unaligned accesses from PBL.

24.2 Features summary

PBL supports the following features:

- Initializes and loads RCW/pre-boot initialization commands from QuadSPI interface
- Reports error upon receiving error response from each interface
- Reports error if RCW byte count is not 64 bytes
- Reports error if RCW address offset is not the specific 64 bytes aligned offset to RCW status registers (RCWSR*n*) to which RCW contents are written
- Reports error in secure boot mode if pre-boot initialization address offset is inside of protected CCSR spaces
- Reports byte count and address misalignment error during pre-boot initialization
- Reports error if PBL command is in invalid format
- Reports error if there is no response from platform when timeout counter expires

24.3 Modes of operation

PBL operates in the following mode:

- Load both RCW and pre-boot initialization commands from QuadSPI flash memory.

24.4 Functional description

This section describes the following topics:

- Device configuration using reset configuration word (RCW)
- Device initialization by PBL
- Required format of data structure used by PBL
- RCW loading by PBL
- Pre-boot initialization command loading by PBL
- Reserved address space used as internal PBL commands
- Error codes

24.4.1 Device configuration using reset configuration word (RCW)

The reset configuration word (RCW) data contains reset configuration information that is loaded by the PBL from a memory device during reset.

The size of the RCW is 64 bytes (512 bits). All of the data read from the RCW source is written to the RCW status registers (RCWSR*n*) in the device configuration and pin control block. See DCFG_RCWSR for more information. The RCWSRs have specific 64 bytes aligned address offsets that the PBL should ensure is written. Any address offset that is not that specific 64 bytes aligned address offset is reported by the PBL as an error to the platform, which asserts the RESET_REQ_B output pin upon receiving the error from the PBL. Refer the Reset configuration word (RCW) source section in Reset, Clocking, and Initialization chapter to see how each encoding for cfg_rcw_src selects a given source of RCW data.

24.4.2 Device initialization by PBL

The cfg_rcw_src configuration input determines the interface to use for retrieving the RCW data. Similarly, RCW[PBI_SRC] determines the interface to use for pre-boot initialization. In these cases, the PBL initializes the QuadSPI interface before accessing the RCW or PBI commands.

Each interface command module (ICM) inside the PBL issues CCSR space accesses to setup each interface properly. If there is an error response during initialization, the PBL reports the error to the platform.

NOTE

Only one interface can be used as the source of RCW and pre-boot initialization data. The PBL data structure must not be split across two interfaces.

24.4.2.1 CCSR registers blocked from PBL during secure boot

The following table lists the CCSR address ranges that are blocked from PBL during secure boot.

The table below specifies the address region blocked for write/read operation when OSPR[ITS]=1 or RCW[SB_EN]=1, during PBI process. This blocking behavior is independent of OSPR[PGE].

Table 24-1. CCSR registers blocked from PBL during secure boot

Address range	Size	Description
0x1E8_0000-0x1E8_FFFF	64 KB	Security fuse processor (SFP)
0x1E9_0000-0x1E9_FFFF	64 KB	SecMon
0x150_0000-0x150_FFFF	64 KB	TZASC
0x151_0000-0x151_FFFF	64 KB	CSU
0x152_0000-0x152_FFFF	64 KB	Platform control
0x170_0000-0x17F_FFFF	1024 KB	SEC
0x2B0_0000-0x2B0_FFFF	64 KB	System counter
0x2AD_0000-0x2AD_FFFF	64 KB	WDOG1
0x2AE_0000-0x2AE_FFFF	64 KB	WDOG2

The table below specifies the address region blocked for read/write operation when OSPR[ITS]=1 or RCW[SB_EN]=1 only when OSPR[PGE]=1, during PBI process.

Table 24-2. Registers blocked from PBL during secure boot

CCSR Offset	Size	Description
0x1EE_0000	2 KB	DCFG (SCRATCHWR offset at 0x1EE_0200)
0x1EE_1000	4 KB	Clocking (all registers)
0x1EE_2000	4 KB	RCPM (all registers)

24.4.3 Required format of data structure used by PBL

The RCW and pre-boot initialization commands share the same data structure format, regardless of the type of external memory device used as the source.

Table 24-3. Required format of data structure used by PBL

	0	1	2	3	4	5	6	7
Preamble (required)	1	0	1	0	1	0	1	0
	0	1	0	1	0	1	0	1
	1	0	1	0	1	0	1	0
	0	1	0	1	0	1	0	1
RCW data	ACS=0	BYTE_CNT = 000000 (64 bytes)						CONT=1
	SYS_ADDR[23-16][1]							
	SYS_ADDR[15-8] [1]							
	SYS_ADDR[7-0] [1]							
	BYTE0							
	BYTE1							

Table continues on the next page...

Table 24-3. Required format of data structure used by PBL (continued)

	0	1	2	3	4	5	6	7
	BYTE2							
	.....							
	BYTE63							
First pre-boot initialization command (optional)	ACS	BYTE_CNT						CONT=1
	SYS_ADDR[23-16]							
	SYS_ADDR[15-8]							
	SYS_ADDR[7-0]							
	BYTE0							
	BYTE1							
	BYTE2							
	.....							
	BYTE N-1 (up to 63)							
Second pre-boot initialization command (optional)	ACS	BYTE_CNT						CONT=1
	SYS_ADDR[23-16]							
	SYS_ADDR[15-8]							
	SYS_ADDR[7-0]							
	BYTE0							
	BYTE1							
	BYTE2							
	.....							
	BYTE N-1 (up to 63)							
.....								
.....								
Last pre-boot initialization command (optional)	ACS	BYTE_CNT						CONT=1
	SYS_ADDR[23-16]							
	SYS_ADDR[15-8]							
	SYS_ADDR[7-0]							
	BYTE0							
	BYTE1							
	BYTE2							
	.....							
	BYTE N-1 (up to 63)							
End command (required, special CRC check command with CONT=0)	ACS=0	BYTE_CNT = 000100 (4 bytes)						CONT=0
	SYS_ADDR[23:16] = 0x61[2]							
	SYS_ADDR[15:8] = 0x00[2]							
	SYS_ADDR[7:0] = 0x40[2]							
	CRC0							
	CRC1							
	CRC2							

Table continues on the next page...

Table 24-3. Required format of data structure used by PBL (continued)

	0	1	2	3	4	5	6	7
	CRC3							

1. SYS_ADDR for the RCW should point to the DCFG_CCSR_RCWSR*n* register.
2. SYS_ADDR = 0x61_0040 is a PBI CRC check command (PBL block base address 0x161_0000, but only 24-bits address are used here). See Pre-boot initialization command loading by PBL for more information.

The following table provides field descriptions for the PBL data structure.

Table 24-4. PBL data structure field descriptions

Field name	Description
ACS	Alternate Configuration Space. Logic 1 for alternate configuration space. Logic 0 for CCSR Space. Note that this field must be logic 0 for the RCW data.
BYTE_CNT	Byte Count. Represents the number of bytes associated with this address/data pair. Note that an encoding of all zeros represents 64 bytes. Only power of two multiples are legal (1, 2, 4, 8, 16, 32, and 64 bytes). The associated SYS_ADDR must be aligned to the byte count of the command. 000000 64 bytes 000001 1 byte 000010 2 bytes 000100 4 bytes 001000 8 bytes 010000 16 bytes 100000 32 bytes All other encoding are reserved
CONT	Continue. This bit should be logic 1 for all commands except for the End command.
SYS_ADDR	System Address. The lowest order system address bits. Note that this permits addressability down to a byte for single byte transactions. SYS_ADDR must be aligned to the byte count of the command. The upper bits are either selected from alternate configuration BAR (depending on value of ACS) and then concatenated with SYS_ADDR to form the full address associated with the command.
BYTEn	Byte number *n* where *n* may range from 0 to 63. Byte 0 corresponds to address 0 relative to the SYS_ADDR (starting address), byte 1 for address 1,..., and byte 63 for address 63. Note the first command must be the RCW address/data pair and must be 64 bytes of data . Note that BYTE*n* is only present/valid in the data structure if *n* is less than the BYTE_CNT.
CRCn	Cyclic Redundancy Check data. CRC value is calculated using CRC-32 algorithm with a start value of 0xFFFF_FFFF and an XOR with 0x0000_0000. The CRC covers all bytes stored in the ROM prior to the CRC. The polynomial used is $1 + x^1 + x^2 + x^4 + x^5 + x^7 + x^8 + x^{10} + x^{11} + x^{12} + x^{16} + x^{22} + x^{23} + x^{26} + x^{32}$

24.4.4 RCW loading by PBL

If initialization of the source memory device completes without error, the PBL fetches RCW data from the source memory device and writes it to the RCW status registers (DCFG_CCSR_RCWSR*n*). See DCFG_RCWSR for more details.

The PBL reports an error and sends a corresponding error code to the platform when one of the following conditions occurs:

- No preamble is detected.
- Alternate configuration space (ACS) is selected (ACS = 1).
- The byte count (BYTE_CNT) is not 64 bytes.
- The system address (SYS_ADDR) is not the specific address offset to the RCWSR to which the RCW contents are written.
- The system does not respond before internal time-out counter (32 bit) expires.

24.4.5 Blocks not accessible by PBL

The following blocks are not accessible through PBL:

- USB3 PHY
- CCI-400

24.4.6 Pre-boot initialization command loading by PBL

If PBI is selected by the RCW[PBI_SRC] field, the PBL PBI commands are processed and routed to CCSR, DDR, and other memory space.

The PBL reports an error and sends a corresponding error code to the platform when one of the following conditions occurs:

- CRC check error
- The byte count is not one of the supported values (1, 2, 4, 8, 16, 32, or 64 bytes). See Table 24-4 for the supported encodings. Note that if byte count is 1 or 2 bytes, the PBL expects 3 or 2 bytes of padding (zeros) after the real data.
- The system address is misaligned to the byte count.
- The system address is in an offset range of protected CCSR space. See CCSR registers blocked from PBL during secure boot, for more information.
- The system does not respond before internal time-out counter expires.
- Invalid internal PBL commands.
- Invalid End command, for example, not valid CRC command with CONT=0.

24.4.7 Reserved address space used as internal PBL commands

When a PBL command accesses the 64 KB (0x161_0000-0x161_FFFF) CCSR address space that is assigned to the PBL, the PBL treats this command entry as a PBL internal command.

Only 24 bits of address are used in PBL command, so the address for PBL command is 0x61_0000 + offset. There are 4 bytes allocated for each command for command parameters. The byte count must be 4 bytes and unused data must be filled with 0s.

The following table defines each PBL command.

Table 24-5. Description of PBL commands

Command name	Offset	Parameter(s)	Command description
Flush	000	N/A	Follows the previous write with a read from the same address. The purpose of this command is to ensure the previous write has taken effect. **NOTE:** Use of the FLUSH command is restricted to CCSR space. Software should use the WAIT command after commands to non-CCSR space to allow them time to complete before issuing subsequent commands to non-CCSR space.
CRC check	040	The 4 bytes of data indicate the cyclic redundancy check (CRC) value. CRC value is calculated using CRC-32 algorithm with a start value of 0xFFFF_FFFF and an XOR with 0x0000_0000.	Checks CRC for data blocks from the point where the last CRC check was performed until the data block before this CRC check command.
Jump	080	The 4 bytes of data indicate the target address for the jump destination. For QuadSPI/, this address is 4 bytes address relative to current address. The address after the jump is a 4 bytes aligned address.	Upon receiving this command, the PBL reads next data from address starting from jump pointer indicated by the command parameter.
Wait	0C0	The 4 bytes of data indicate the number of cycles to wait.The clock source for the wait cycle is a PBL clock cycle (platform/2).	Upon receiving this command, the PBL waits a number of cycles indicated by the command parameter before reading the next data.

24.4.8 Error codes

The PBL reports errors to the platform under various conditions. PBL error codes are also sent along with the error indication.

The platform stores the error code in the DCFG_CCSR_RSTRQPBLSR[ERR_CODE] bit. See DCFG_RSTRQPBLSR for more information.

The following table shows the encoding for each pre-boot error condition.

Table 24-6. Pre-boot error encoding

Error code[0:6]	Error condition
0x00-0x0F	Reserved
0x10	Reserved
0x40-0x6F	Reserved
0x70	No preamble is detected
0x71	ACS is logic 1 during RCW
0x72	Byte count is not 64 for RCW or 4 for PBL commands or 1/2/4/8/16/32/64 for pre-boot initialization commands
0x73	Misaligned address
0x74	Address is in protected space
0x75	CRC error
0x76	Time out counter expires
0x77	Unrecognized PBL commands, including unrecognized offset and invalid parameter.
0x78	Data transfer error from memory devices
0x79	Invalid end command error
0x7A	Invalid RCW or pre-boot initialization command source encoding
0x7B-0x7F	Reserved

24.5 Initialization/Application information

The PBL starts searching for the first good block. It begins to fetch the PBL image at the starting address of that block. It automatically searches for the next good block should the PBL image extend past one block.

The PBL block search routine continues to search as necessary until reaching the end of the memory device.

24.5.1 Starting addresses

The following table lists the starting addresses of data fetched from each interface.

Table 24-7. Starting addresses

Interface	Starting address
QuadSPI	0x0000_0000

24.5.2 Software restrictions

Following are the restrictions for programming the PBI commands:

- Software must issue a Flush command after updating the alternate configuration space register (ALTCBAR) using PBI commands. This ensures that the logic has seen the update prior to any dependent writes being issued by the PBL.
- Use of the Flush command is restricted to CCSR space. Software must use the Wait command after commands to non-CCSR space to allow them time to complete before issuing subsequent commands to non-CCSR space.

24.5.3 Software recommendations

Following are the recommendations for programming the PBL data structure:

- A CRC check command should always be placed immediately following the RCW command to ensure that a corrupted RCW is not consumed.

Chapter 25
PCI Express Interface Controller

25.1 The PCI Express module as implemented on the chip

This section provides details about how the PCI Express module is implemented on the chip.

NOTE

The PCI Express controller as instantiated on this chip does not support hardware coherency. All incoming PCI Express transactions are made non IO-coherent.

25.1.1 PCI Express MSI implementation

The following steps describe the flow sequence of PCI Express MSI implementation:

- Write to SCFG_PEXnMSIIR[IBS] sets the index and triggers the corresponding PEX MSI interrupt. For example, write to PEX1MSIIR with a value 0x7800_0000 triggers the PCI Express controller 1 MSI interrupt line (148) with MSI interrupt index (15^{th} bit) set in PEX1MSIR.
- The SCFG_PEXnMSIR[SHn] indicates one or more corresponding shared MSI interrupts are pending with the bit numbers selected by the SCFG_PEXnMSIIR[IBS] bit field.
- The status register and the MSI interrupt line is cleared on read access to SCFG_PEXnMSIR register.

See SCFG_PEX1MSIIR and SCFG_PEX1MSIR for more information.

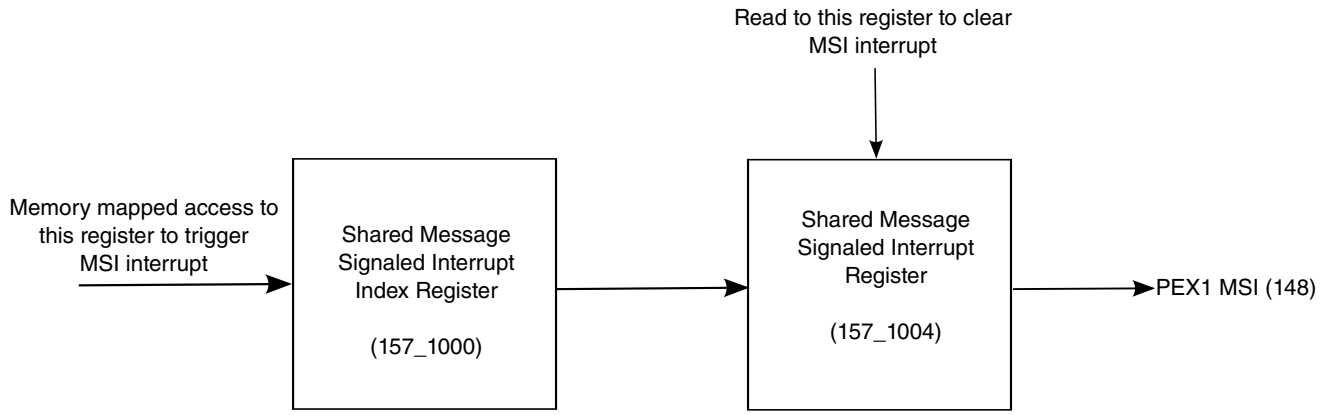


Figure 25-1. PCI Express MSI implementation

25.1.2 PCI Express soft reset support

The PEX_LUT_PEXLDBG provides the software configurability to reset PCI Express controllers.

- Write to PEX_LUT_PEXLDBG[SR] bit resets the PCI Express 1 controller. This bit needs to be cleared for the reset to be de-asserted.

25.1.3 PCI Express PM turnoff message support

The PEX PFa PCIE message command register (PEX_PF0_MCR) provides the software configurability for PM turnoff message. Software needs to set and clear PEX_PF0_MCR[PTOMR] to generate PM turnoff message for power management support for PCI Express controllers.

25.2 Introduction

The PCI Express interface is compatible with the *PCI Express™ Base Specification, Revision 3.0* (available from http://www.pcisig.org). It is beyond the scope of this manual to document the intricacies of the PCI Express protocol. This chapter describes the PCI Express controller of this device and provides a basic description of the PCI Express protocol. The specific emphasis is directed at how the device implements the PCI Express specification. Designers of systems incorporating PCI Express devices should refer to the specification for a thorough description of PCI Express.

NOTE

Much of the available PCI Express literature refers to a 16-bit quantity as a WORD and a 32-bit quantity as a DWORD. Note that this is inconsistent with the terminology in the rest of this manual where the terms 'word' and 'double word' refer to a 32-bit and 64-bit quantity, respectively. Where necessary to avoid confusion, the precise number of bits or bytes is specified.

25.2.1 Overview

The PCI Express controller connects the internal platform to a serial interface.

As both an initiator and a target device, the PCI Express interface is capable of high-bandwidth data transfer and is designed to support next generation I/O devices. Upon coming out of reset, the PCI Express interface performs link width negotiation and exchanges flow control credits with its link partner. Once link autonegotiation is successful, the controller is in operation.

Internally, the design contains queues to keep track of inbound and outbound transactions. There is control logic that handles buffer management, bus protocol, transaction spawning and tag generation. In addition, there are memory blocks used to store inbound and outbound data.

The PCI Express controller can be configured to operate as either a PCI Express Root Complex (RC) or Endpoint (EP) device. An RC device connects the host CPU/memory subsystem to I/O devices while an EP device typically denotes a peripheral or I/O device. In RC mode, a PCI Express type 1 configuration header is used; in EP mode, a PCI Express type 0 configuration header is used.

This figure shows a high-level block diagram of the PCI Express controller.

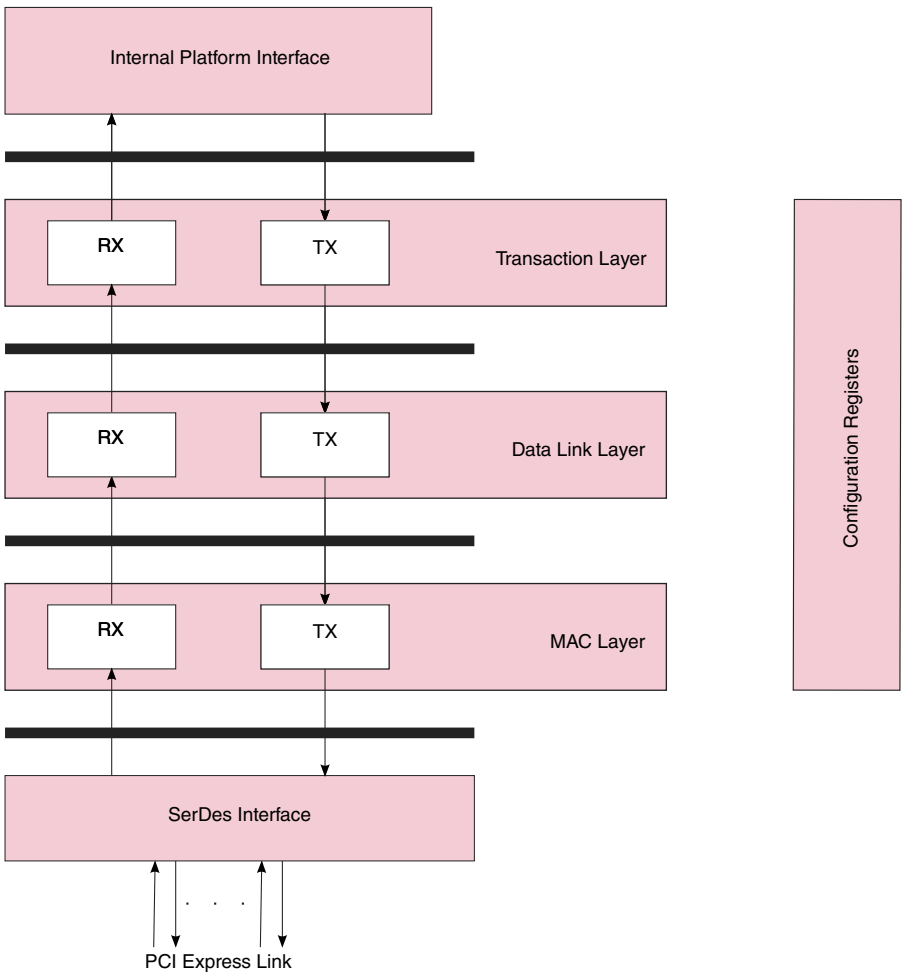


Figure 25-2. PCI Express Controller Block Diagram

As an initiator, the PCI Express controller supports memory read and write operations. In addition, configuration and I/O transactions are supported if the PCI Express controller is in RC mode. As a target interface, the PCI Express controller accepts read and write operations to local memory space. As an EP device, the PCI Express controller accepts configuration transactions to the internal PCI Express configuration registers. Message generation and acceptance are supported in both RC and EP modes. Locked transactions and inbound I/O transactions are not supported.

25.2.1.1 Outbound Transactions

Outbound internal platform transactions to PCI Express are first mapped to a translation window to determine what PCI Express transactions are to be issued. A transaction from the internal platform can become a PCI Express Memory, I/O,Message, or Configuration transaction depending on the window attributes.

A transaction may be broken up into smaller sized transactions depending on the original request size, transaction type, and either the PCI Express Device Control register [MAX_PAYLOAD_SIZE] field for write requests or the PCI Express Device Control register [MAX_READ_SIZE] field for read requests. The controller performs PCI Express ordering rule checking to determine which transaction is to be sent on the PCI Express link.

In general, transactions are serviced in the order that they are received from the internal platform . Only when there is a stalled condition does the controller apply PCI Express ordering rules to outstanding transactions. For posted write transactions, once all data has been received from the internal platform , the data is forwarded to the PCI Express link and the transaction is considered as done. For non-posted write transactions, the controller waits for the completion packets to return before considering the transaction finished. For non-posted read transactions, the controller waits for all completion packets to return and then forwards all data back to the internal platform before terminating the transaction.

Note that after reset or when recovering from a link down condition, external transactions should not be attempted until the link has successfully trained. Software can poll the LTSSM state status to check the status of link training before issuing external requests.

In EP mode, after reset or when recovering from a link down condition, the SoC must not generate any outbound memory or I/O transactions until the remote host has configured the Bus Master Enable bit in the PCI Command register.

25.2.1.2 Inbound Transactions

Inbound PCI Express transactions to internal platform are first mapped to a translation window to determine what internal platform transactions are to be issued.

A transaction may be broken up into smaller sized transactions when sending to the internal platform depending on the original request size, byte enables and starting/ending addresses. The controller performs PCI Express ordering rule checking to determine what transaction is to be sent next to the internal platform.

In general, transactions are serviced in the order that they are received from the PCI Express link. Only when there is a stalled condition does the controller apply PCI Express ordering to outstanding transactions. For posted write transactions, once all data has been received from the PCI Express link, the data is forwarded to the internal platform and the transaction is considered as done. For non-posted read transactions, the controller forwards internal platform data back to the PCI Express link.

Note that the controller splits transactions at the crossing of every 256-byte-aligned boundary when sending data back to the PCI Express link.

25.2.2 Features

The following is a list of features supported by the PCI Express controller:

- Compatible with the *PCI Express™ Base Specification, Revision 3.0*
- Supports Root Complex (RC) and Endpoint (EP) configurations
- 32- and 64-bit PCI Express address support
- 40-bit internal platform address support
- x1 link support.
- Supports accesses to all PCI Express memory and I/O address spaces (requestor only)
- Supports posting of processor-to-PCI Express and PCI Express-to-memory writes
- Supports strong and relaxed transaction ordering rules
- Enforces outbound PCI Express ordering rules and inbound internal platform priority
- PCI Express configuration registers (type 0 in EP mode, type 1 in RC mode)
- Baseline and advanced error reporting support
- One virtual channel (VC0)
- 256-byte maximum payload size (MAX_PAYLOAD_SIZE)
- Supports 64-bit MSI interrupts. Note that MSI support is provided in the SCFG module.
- Credit-based flow control management handled by PCI Express core.
- Supports PCI Express messages and interrupts
- Accepts up to 256-byte transactions from the internal platform
- Supports Expansion ROM.

25.2.3 Modes of Operation

Several parameters that affect the PCI Express controller modes of operation are determined at power-on reset (POR) by reset configuration word (RCW) fields configured depending on SoC product.

Table 25-1. POR Parameters for PCI Express Controller

RCW Parameter	Description
Host/Agent	Selects between Root Complex (RC) and Endpoint (EP) modes
SerDes Protocol Select SRDS_PRTCL_Sn	Determines the link width

Table continues on the next page...

Table 25-1. POR Parameters for PCI Express Controller (continued)

RCW Parameter	Description
SerDes frequency divider for PCI Express	Determines the link speed

25.2.3.1 Root Complex/Endpoint Modes

The PCI Express controller can function as either a Root Complex (RC) or an Endpoint (EP) on the PCI Express link. The host/agent configuration field, RCW[HOST_AGT_PEX], determines the RC/EP mode.

25.2.3.2 Link Speed

The initial link speed is determined by the SerDes frequency divider for PCI Express field (RCW[SRDS_DIV_PEX_Sn]).

See RCW Field Definitions for more information. The specific configurations are detailed in Reference Clocks for SerDes Protocols.

25.3 External Signal Descriptions

The PCI Express specification defines the connection between two devices as a link, which can be composed of a single or multiple lanes. Each lane consists of a differential pair for transmitting (TX[n]_P and TX[n]_N) and a differential pair for receiving (RX[n]_P and RX[n]_N) with an embedded data clock.

Table 25-2. PCI Express Interface Signals—Detailed Signal Descriptions

Signal	I/O	Description	
SD_RX[n]_P	I	Receive data, positive. The receive data signals carry PCI Express packet information.	
		State Meaning	Asserted/Negated—Represents data being received from the PCI Express interface.
		Timing	Assertion/Negation—As described in the *PCI Express Base Specification, Revision 3.0*.
SD_RX[n]_N	I	Receive data, negative. The receive data signals carry PCI Express packet information.	
		State Meaning	Asserted/Negated—Represents the inverse of data being received from the PCI Express interface.
		Timing	Assertion/Negation—As described in the *PCI Express Base Specification, Revision 3.0*.
SD_TX[n]_P	O	Transmit data, positive. The transmit data signals carry PCI Express packet information.	

Table continues on the next page...

Table 25-2. PCI Express Interface Signals—Detailed Signal Descriptions (continued)

Signal	I/O	Description	
		State Meaning	Asserted/Negated—Represents data being transmitted to the PCI Express interface.
		Timing	Assertion/Negation—As described in the *PCI Express Base Specification, Revision 3.0*.
SD_TX[n]_N	O	Transmit data, negative. The transmit data signals carry PCI Express packet information.	
		State Meaning	Asserted/Negated—Represents the inverse of data being transmitted to the PCI Express interface.
		Timing	Assertion/Negation—As described in the *PCI Express Base Specification, Revision 3.0*.

25.4 Memory map/register overview

25.4.1 PCI Express configuration registers

The PCI Express module supports the same configuration registers in both the internal memory map and in PCI Express configuration space. They differ only in whether they are accessed from an internal initiator or from an external initiator on the PCI Express interface. With the exception of the registers in the PEX module internal configuration space, the configuration registers are specified by the PCI Express specification for every PCI Express device. The registers in the PEX module internal configuration space are used for chip-specific functionality and are not defined by the PCI Express specification.

Table 25-3. PCI Express memory map

Register space	Offset	NEXT pointer	Notes
Type 0 configuration header	0x000	0x40	EP use Type 0
Type 1 configuration header	0x000	0x40	RC use Type 1
Power management capability structure	0x040	0x50	
MSI message capability structure	0x050	0x70	
PCI Express capability structure	0x070	0x000 (NULL)	
Advanced error reporting capability structure	0x100	0x148	
Secondary PCI Express capability structure	0x148	0x000 (NULL)	
PEX module internal configuration space	0x700	—	
BAR Mask Registers	0x1000	—	

25.4.2 PEX register descriptions

25.4.2.1 PCI_Express_Configuration_Registers Memory map

PEX1 base address: 340_0000h

Offset	Register	Width (In bits)	Access	Reset value
0h	PCI Express Vendor ID Register (Vendor_ID_Register)	16	RO	1957h
2h	PCI Express Device ID Register (Device_ID_Register)	16	RO	8100h
4h	PCI Express Command Register (Command_Register)	16	RW	0000h
6h	PCI Express Status Register (Status_Register)	16	W1C	0010h
8h	PCI Express Revision ID Register (Revision_ID_Register)	8	RO	01h
9h	PCI Express Class Code Register (Class_Code_Register)	24	RO	0B_2000h
Ch	PCI Express Cache Line Size Register (Cache_Line_Size_Register)	8	RW	00h
Dh	PCI Express Latency Timer Register (Latency_Timer_Register)	8	RO	00h
Eh	PCI Express Header Type Register (Header_Type_Register)	8	RO	00h
10h	PCI Express Base Address Register 0 (BAR0)	32	RW	0000_0000h
14h	PCI Express Base Address Register 1 (BAR1)	32	RW	0000_0000h
18h	PCI Express Base Address Register 2 (BAR2)	32	RW	0000_000Ch
18h	PCI Express Primary Bus Number Register (Primary_Bus_Number_Register)	8	RW	00h
19h	PCI Express Secondary Bus Number Register (Secondary_Bus_Number_Register)	8	RW	00h
1Ah	PCI Express Subordinate Bus Number Register (Subordinate_Bus_Number_Register)	8	RW	00h
1Ch	PCI Express Base Address Register 3 (BAR3)	32	RW	0000_0000h
1Ch	PCI Express I/O Base Register (IO_Base_Register)	8	RW	01h
1Dh	PCI Express I/O Limit Register (IO_Limit_Register)	8	RW	01h
1Eh	PCI Express Secondary Status Register (Secondary_Status_Register)	16	W1C	0000h
20h	PCI Express Base Address Register 4 (BAR4)	32	RW	0000_000Ch
20h	PCI Express Memory Base Register (Memory_Base_Register)	16	RW	0000h
22h	PCI Express Memory Limit Register (Memory_Limit_Register)	16	RW	0000h
24h	PCI Express Base Address Register 5 (BAR5)	32	RW	0000_0000h
24h	PCI Express Prefetchable Memory Base Register (Prefetchable_Memory_Base_Register)	16	RW	0001h
26h	PCI Express Prefetchable Memory Limit Register (Prefetchable_Memory_Limit_Register)	16	RW	0001h

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
28h	PCI Express Prefetchable Base Upper 32 Bits Register (Prefetchable_Base_Upper_32_Bits_Register)	32	RW	0000_0000h
2Ch	PCI Express Prefetchable Limit Upper 32 Bits Register (Prefetchable_Limit_Upper_32_Bits_Register)	32	RW	0000_0000h
2Ch	PCI Express Subsystem Vendor ID Register (Subsystem_Vendor_ID_Register)	16	RO	0000h
2Eh	PCI Express Subsystem ID Register (Subsystem_ID_Register)	16	RO	0000h
30h	PCI Express Expansion ROM Base Address Register (EP-Mode) (Expansion_ROM_BAR_Type0)	32	RW	0000_0000h
30h	PCI Express I/O Base Upper 16 Bits Register (IO_Base_Upper_16_Bits_Register)	16	RO	0000h
32h	PCI Express I/O Limit Upper 16 Bits Register (IO_Limit_Upper_16_Bits_Register)	16	RO	0000h
34h	Capabilities Pointer Register (Capabilities_Pointer_Register)	8	RO	40h
38h	PCI Express Expansion ROM Base Address Register (RC-Mode) (Expansion_ROM_BAR_Type1)	32	RW	0000_0000h
3Ch	PCI Express Interrupt Line Register (Interrupt_Line_Register)	8	RW	FFh
3Dh	PCI Express Interrupt Pin Register (Interrupt_Pin_Register)	8	RO	01h
3Eh	PCI Express Bridge Control Register (Bridge_Control_Register)	16	RW	0000h
3Eh	PCI Express Minimum Grant Register (Minimum_Grant_Register)	8	RO	00h
3Fh	PCI Express Maximum Latency Register (Maximum_Latency_Register)	8	RO	00h
40h	PCI Express Power Management Capability ID Register (Power_Management_Capability_ID_Register)	8	RO	01h
42h	PCI Express Power Management Capabilities Register (Power_Management_Capabilities_Register)	16	RO	7E23h
44h	PCI Express Power Management Status and Control Register (Power_Management_Status_and_Control_Register)	16	RW	0000h
47h	PCI Express Power Management Data Register (Power_Management_Data_Register)	8	RO	00h
50h	PCI Express MSI Message Capability ID Register (MSI_Message_Capability_ID_Register)	8	RO	05h
52h	PCI Express MSI Message Control Register (MSI_Message_Control_Register)	16	RW	0088h
54h	PCI Express MSI Message Address Register (MSI_Message_Address_Register)	32	RW	0000_0000h
58h	PCI Express MSI Message Upper Address Register (MSI_Message_Upper_Address_Register)	32	RW	0000_0000h
5Ch	PCI Express MSI Message Data Register (MSI_Message_Data_Register)	16	RW	0000h
70h	PCI Express Capability ID Register (Capability_ID_Register)	8	RO	10h
72h	PCI Express Capabilities Register (Capabilities_Register)	16	RO	0002h
74h	PCI Express Device Capabilities Register (Device_Capabilities_Register)	32	RO	0000_8001h
78h	PCI Express Device Control Register (Device_Control_Register)	16	RW	2810h

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
7Ah	PCI Express Device Status Register (Device_Status_Register)	16	W1C	0000h
7Ch	PCI Express Link Capabilities Register (Link_Capabilities_Register)	32	RO	0043_F483h
80h	PCI Express Link Control Register (Link_Control_Register)	16	RW	0008h
82h	PCI Express Link Status Register (Link_Status_Register)	16	RO	1000h
84h	PCI Express Slot Capabilities Register (Slot_Capabilities_Register)	32	RO	0000_0000h
88h	PCI Express Slot Control Register (Slot_Control_Register)	16	RW	03C0h
8Ah	PCI Express Slot Status Register (Slot_Status_Register)	16	W1C	0008h
8Ch	PCI Express Root Control Register (Root_Control_Register)	16	RW	0000h
8Eh	PCI Express Root Capabilities Register (Root_Capabilities_Register)	16	RW	0000h
90h	PCI Express Root Status Register (Root_Status_Register)	32	RW	0000_0000h
94h	PCI Express Device Capabilities 2 Register (Device_Capabilities_2_Register)	32	RO	0000_001Fh
98h	PCI Express Device Control 2 Register (Device_Control_2_Register)	16	RW	0000h
9Ch	PCI Express Link Capabilities 2 Register (Link_Capabilities_2_Register)	32	RO	0000_000Eh
A0h	PCI Express Link Control 2 Register (Link_Control_2_Register)	16	RW	0003h
A2h	PCI Express Link Status 2 Register (Link_Status_2_Register)	16	RO	0000h
100h	PCI Express Advanced Error Reporting Capability ID Register (Advanced_Error_Reporting_Capability_ID_Register)	16	RO	0001h
104h	PCI Express Uncorrectable Error Status Register (Uncorrectable_Error_Status_Register)	32	W1C	0000_0000h
108h	PCI Express Uncorrectable Error Mask Register (Uncorrectable_Error_Mask_Register)	32	RW	0000_0000h
10Ch	PCI Express Uncorrectable Error Severity Register (Uncorrectable_Error_Severity_Register)	32	RW	0046_2030h
110h	PCI Express Correctable Error Status Register (Correctable_Error_Status_Register)	32	W1C	0000_0000h
114h	PCI Express Correctable Error Mask Register (Correctable_Error_Mask_Register)	32	RW	0000_2000h
118h	PCI Express Advanced Error Capabilities and Control Register (Advanced_Error_Capabilities_and_Control_Register)	32	RW	0000_00A0h
11Ch	PCI Express Header Log Register 1 (Header_Log_Register_DWORD1)	32	RO	0000_0000h
120h	PCI Express Header Log Register 2 (Header_Log_Register_DWORD2)	32	RO	0000_0000h
124h	PCI Express Header Log Register 3 (Header_Log_Register_DWORD3)	32	RO	0000_0000h
128h	PCI Express Header Log Register 4 (Header_Log_Register_DWORD4)	32	RO	0000_0000h
12Ch	PCI Express Root Error Command Register (Root_Error_Command_Register)	32	RW	0000_0000h
130h	PCI Express Root Error Status Register (Root_Error_Status_Register)	32	W1C	0000_0000h

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
134h	PCI Express Correctable Error Source ID Register (Correctable_Error_Source_ID_Register)	16	RO	0000h
136h	PCI Express Error Source ID Register (Error_Source_ID_Register)	16	RO	0000h
148h	Secondary PCI Express Extended Capability Header (SPCIE_CAP_HEADER_REG)	32	RO	0001_0019h
14Ch	Link Control 3 Register (LINK_CONTROL3_REG)	32	RW	0000_0000h
150h	Lane Error Status Register (LANE_ERR_STATUS_REG)	32	W1C	0000_0000h
154h	Lane Equalization Control Register (LANE0_EQUALIZATION_CONTROL)	16	RO	7F7Fh
71Ch	Symbol Timer Register and Filter Mask 1 Register (SYMBOL_TIMER_FILTER_1_OFF)	32	RW	0000_0280h
890h	Gen3 Control Register (GEN3_RELATED_OFF)	32	RW	0000_0001h
8BCh	DBI Read-only Write Enable Register (MISC_CONTROL_1_OFF)	32	RW	0000_0000h
8E0h	Coherency Control Register 1 (COHERENCY_CONTROL_1_OFF)	32	RW	0000_0000h
8E4h	Coherency Control Register 2 (COHERENCY_CONTROL_2_OFF)	32	RW	0000_0000h
8E8h	Coherency Control Register 3 (COHERENCY_CONTROL_3_OFF)	32	RU	0000_0000h
900h	iATU Index Register (IATU_VIEWPORT_OFF)	32	RW	0000_0000h
904h	iATU Region Control 1 Register (IATU_REGION_CTRL_1_OFF_INBOUND_0)	32	RW	0000_0000h
904h	iATU Region Control 1 Register (IATU_REGION_CTRL_1_OFF_OUTBOUND_0)	32	RW	0000_0000h
908h	iATU Region Control 2 Register (IATU_REGION_CTRL_2_OFF_INBOUND_0)	32	RW	0000_0000h
908h	iATU Region Control 2 Register (IATU_REGION_CTRL_2_OFF_OUTBOUND_0)	32	RW	0000_0000h
90Ch	iATU Lower Base Address Register (IATU_LWR_BASE_ADDR_OFF_INBOUND_0)	32	RW	0000_0000h
90Ch	iATU Lower Base Address Register (IATU_LWR_BASE_ADDR_OFF_OUTBOUND_0)	32	RW	0000_0000h
910h	iATU Upper Base Address Register (IATU_UPPER_BASE_ADDR_OFF_INBOUND_0)	32	RW	0000_0000h
910h	iATU Upper Base Address Register (IATU_UPPER_BASE_ADDR_OFF_OUTBOUND_0)	32	RW	0000_0000h
914h	iATU Limit Address Register (IATU_LIMIT_ADDR_OFF_INBOUND_0)	32	RW	0000_0FFFh
914h	iATU Limit Address Register (IATU_LIMIT_ADDR_OFF_OUTBOUND_0)	32	RW	0000_0FFFh
918h	iATU Region#N Lower Offset Address Register (IATU_LWR_TARGET_ADDR_OFF_INBOUND_0)	32	RW	0000_0000h
918h	iATU Outbound Region#N Lower Offset Address Register (IATU_LWR_TARGET_ADDR_OFF_OUTBOUND_0)	32	RW	0000_0000h
91Ch	iATU Upper Target Address Register (IATU_UPPER_TARGET_ADDR_OFF_INBOUND_0)	32	RW	0000_0000h
91Ch	iATU Upper Target Address Register (IATU_UPPER_TARGET_ADDR_OFF_OUTBOUND_0)	32	RW	0000_0000h

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
1010h	Base Address Register 0 Mask (BAR0_MASK)	32	WO	00FF_FFFFh
1014h	Base Address Register 1 Mask (BAR1_MASK)	32	WO	03FF_FFFFh
1018h	Base Address Register 2 Mask (BAR2_MASK)	32	WO	FFFF_FFFFh
101Ch	Base Address Register 3 Mask (BAR3_MASK)	32	WO	0000_00FFh
1020h	Base Address Register 4 Mask (BAR4_MASK)	32	WO	FFFF_FFFFh
1024h	Base Address Register 5 Mask (BAR5_MASK)	32	WO	0000_00FFh
1030h	Expansion ROM Base Address Register Mask (EP mode) (EXP_ROM_BAR_MASK_EP)	32	WO	00FF_FFFFh
1038h	Expansion ROM Base Address Register Mask (RC mode) (EXP_ROM_BAR_MASK_RC)	32	WO	00FF_FFFFh

25.4.2.2 PCI Express Vendor ID Register (Vendor_ID_Register)

25.4.2.2.1 Offset

Register	Offset
Vendor_ID_Register	0h

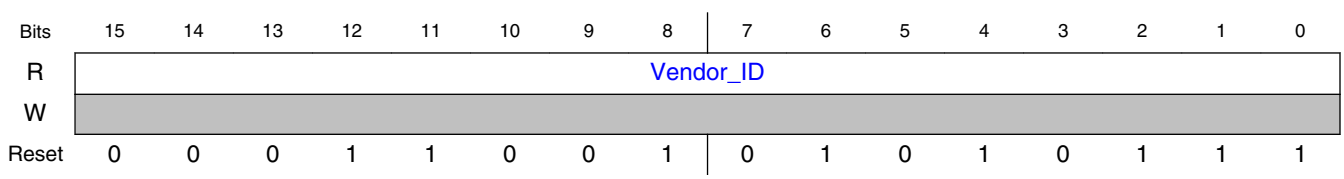
25.4.2.2.2 Function

The vendor ID register is used to identify the manufacturer of the device.

NOTE

This register is writeable using internal accesses, but is read-only from inbound configuration accesses by an external host.

25.4.2.2.3 Diagram



25.4.2.2.4 Fields

Field	Function
15-0 Vendor_ID	Vendor ID 0x1957 (NXP)

25.4.2.3 PCI Express Device ID Register (Device_ID_Register)

25.4.2.3.1 Offset

Register	Offset
Device_ID_Register	2h

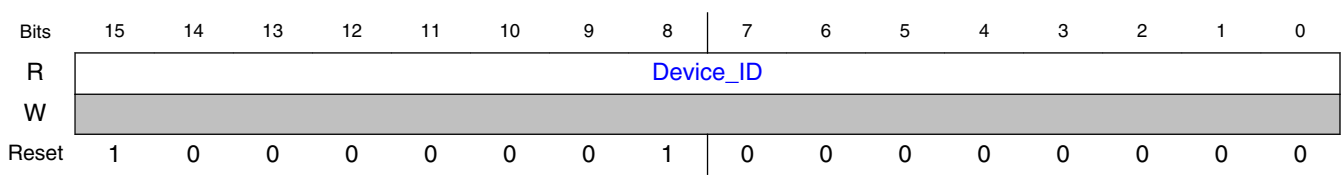
25.4.2.3.2 Function

The device ID register is used to identify the device.

NOTE

This register is writeable using internal accesses, but is read-only from inbound configuration accesses by an external host.

25.4.2.3.3 Diagram



25.4.2.3.4 Fields

Field	Function
15-0 Device_ID	Device ID 1000000100000000b - LS1012AE with security 1000000100000001b - LS1012A without security

25.4.2.4 PCI Express Command Register (Command_Register)

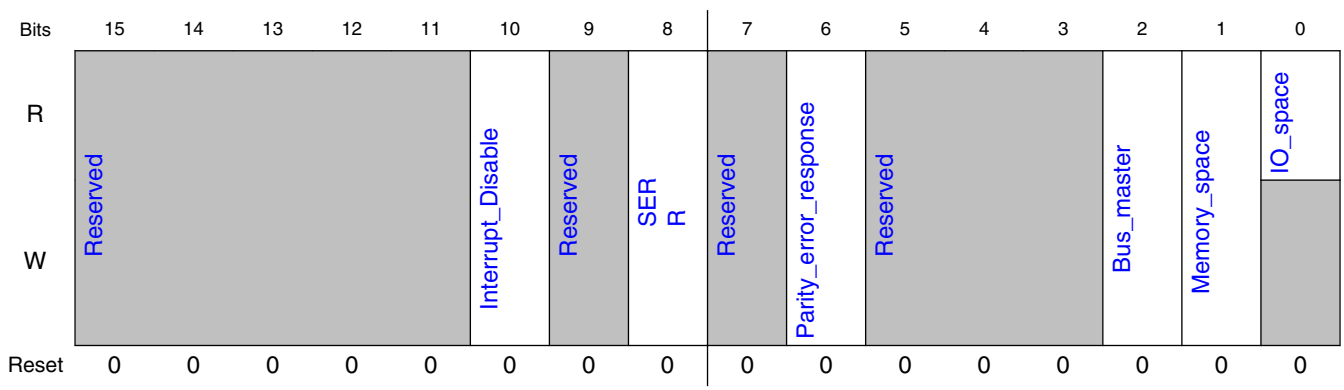
25.4.2.4.1 Offset

Register	Offset
Command_Register	4h

25.4.2.4.2 Function

The command register provides control over the ability to generate and respond to PCI Express cycles.

25.4.2.4.3 Diagram



25.4.2.4.4 Fields

Field	Function
15-11 —	Reserved
10 Interrupt_Disable	Interrupt disable Controls the ability to generate INTx interrupt messages. Any INTx emulation interrupts already asserted by this device must be deasserted when this bit is set. 0b - Enables INTx interrupt messages 1b - Disables INTx interrupt messages
9 —	Reserved
8 SERR	SERR# enable Controls the reporting of fatal and non-fatal errors detected by the device to the Root Complex.

Table continues on the next page...

Field	Function
	NOTE: The error control and status bits in the command and status registers control PCI-compatible error reporting. PCI Express advanced error reporting is controlled by the PCI Express device control register described in PCI Express Device Control Register (Device_Control_Register) and the advanced error reporting registers (offsets 100h through 137h). 0b - Disables reporting 1b - Enables reporting
7 ——	Reserved
6 Parity_error_response	Parity error response Controls whether this PCI Express controller responds to parity errors. **NOTE:** The error control and status bits in the command and status registers control PCI-compatible error reporting. PCI Express advanced error reporting is controlled by the PCI Express device control register described in PCI Express Device Control Register (Device_Control_Register) and the advanced error reporting registers (offsets 100h through 137h). 0b - Parity errors are ignored and normal operation continues. 1b - Parity errors cause the appropriate bit in the PCI Express status register to be set. However, note that errors are reported based on the values set in the PCI Express error enable and detection registers.
5-3 ——	Reserved
2 Bus_master	Bus master enable Indicates whether this PCI Express device is configured as a master. EP mode: Clearing this bit prevent the device from issuing any memory or I/O transactions. Because MSI interrupts are effectively memory writes, clearing this bit also disables the ability of the device to issue MSI interrupts. RC mode: Clearing this bit disables the ability of the device to forward memory transactions upstream. This causes any inbound memory transaction to be treated as an unsupported request. 0b - Disables the ability to generate PCI Express accesses 1b - Enables this PCI Express controller to behave as a PCI Express bus master
1 Memory_space	Memory space enable Controls whether this PCI Express device (as a target) responds to memory accesses. EP mode: Clearing this bit prevents the device from accepting any memory transaction. RC mode: This bit is ignored. It does not affect outbound memory transaction 0b - This PCI Express device does not respond to PCI Express memory space accesses. 1b - This PCI Express device responds to PCI Express memory space accesses.
0 IO_space	I/O space enable EP mode: Clearing this bit prevents the device from accepting any IO transaction. Note that this bit is a don't care in EP mode since the device does not support IO transaction. RC mode: This bit is ignored. It does not affect outbound IO transaction. 0b - This PCI Express device (as a target) does not respond to PCI Express I/O space accesses. 1b - This PCI Express device (as a target) does respond to PCI Express I/O space accesses.

25.4.2.5 PCI Express Status Register (Status_Register)

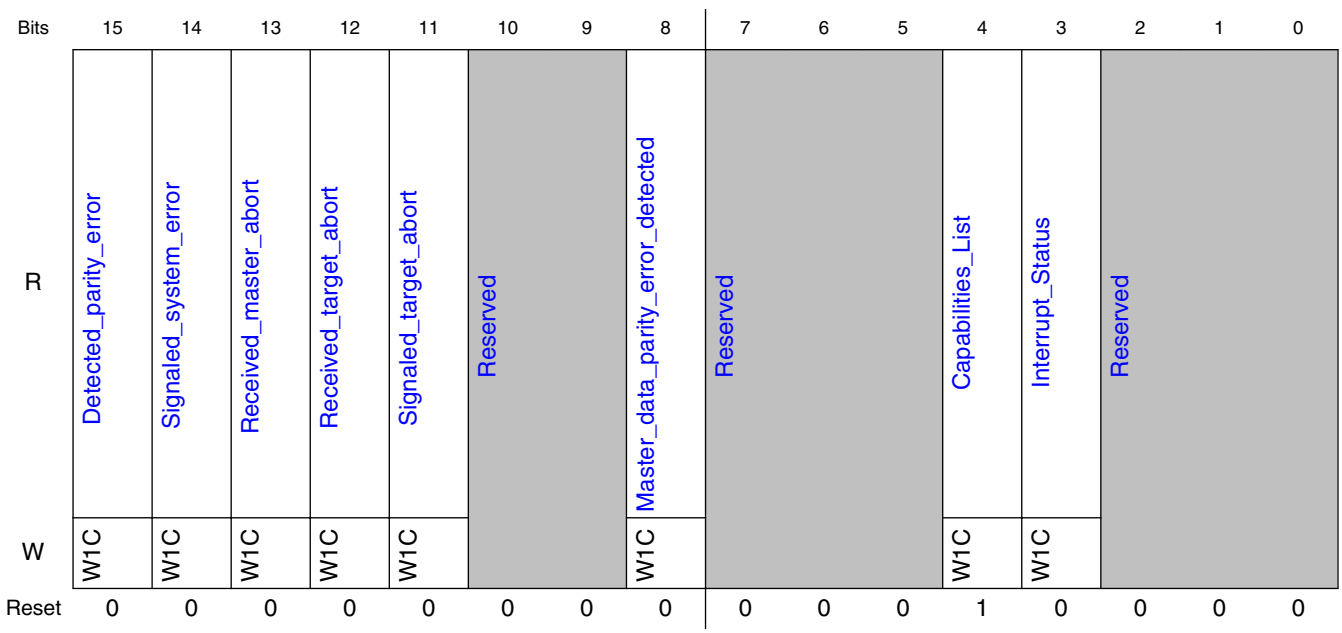
25.4.2.5.1 Offset

Register	Offset
Status_Register	6h

25.4.2.5.2 Function

The status register is used to record status information for PCI Express related events.

25.4.2.5.3 Diagram



25.4.2.5.4 Fields

Field	Function
15 Detected_parity_error	Detected parity error Set whenever a device receives a poisoned TLP regardless of the state of bit 6 in the command register. [1]
14 Signaled_system_error	Signaled system error Set whenever a device sends a ERR_FATAL or ERR_NONFATAL message and the SERR enable bit in the command register is set.[1]
13	Received master abort

Table continues on the next page...

Field	Function
Received_master_abort	Set whenever a requestor receives a completion with unsupported request completion status.[1]
12 Received_target_abort	Received target abort Set whenever a device receives a completion with completer abort completion status.[1]
11 Signaled_target_abort	Signaled target abort Set whenever a device completes a request using completer abort completion status.[1]
10-9 —	Reserved
8 Master_data_parity_error_detected	Master data parity error Set by the requestor (primary side for Type1 headers) when either the requestor receives a completion marked poisoned or the requestor poisons a write request. Note that the parity error enable bit (bit 6) in the command register must be set for this bit to be set.[1]
7-5 —	Reserved
4 Capabilities_List	Capabilities list All PCI Express devices are required to implement the PCI Express capability structure.
3 Interrupt_Status	Interrupt status Set when an INTx interrupt message is pending internally to the device. Note that this bit is associated with INTx messages and not Message Signaled Interrupts.
2-0 —	Reserved

1. The error control and status bits in the command and status registers control PCI-compatible error reporting. PCI Express advanced error reporting is controlled by the PCI Express device control register described in PCI Express Device Control Register (Device_Control_Register) and the advanced error reporting capability structure starting at offset 100h.

25.4.2.6 PCI Express Revision ID Register (Revision_ID_Register)

25.4.2.6.1 Offset

Register	Offset
Revision_ID_Register	8h

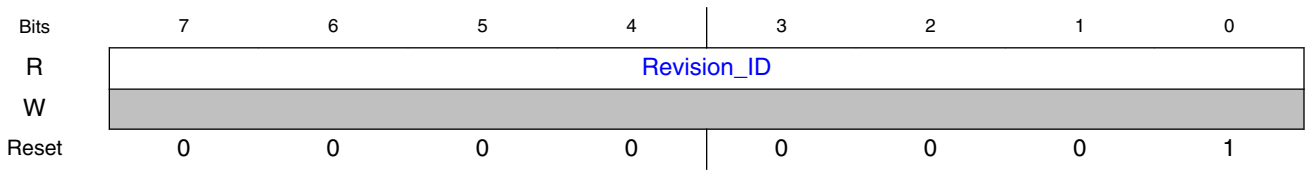
25.4.2.6.2 Function

The revision ID register is used to identify the revision of the device.

NOTE

This register is writeable using internal accesses, but is read-only from inbound configuration accesses by an external host.

25.4.2.6.3 Diagram



25.4.2.6.4 Fields

Field	Function
7-0 Revision_ID	Revision ID Revision specific.

25.4.2.7 PCI Express Class Code Register (Class_Code_Register)

25.4.2.7.1 Offset

Register	Offset
Class_Code_Register	9h

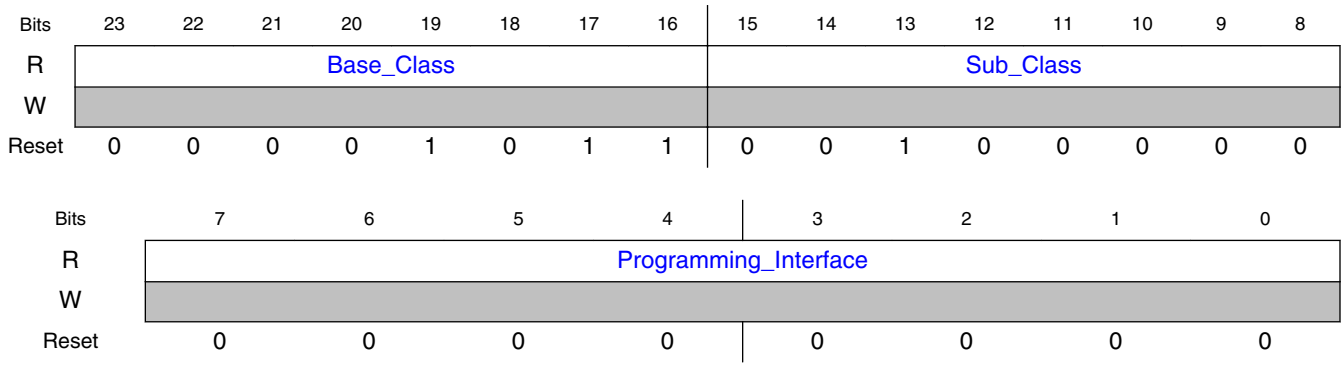
25.4.2.7.2 Function

The class code register is comprised of three single-byte fields—base class (offset 0x0B), sub-class (offset 0x0A), and programming interface (offset 0x09)—that indicate the basic functionality of the function.

NOTE

This register is writeable using internal accesses, but is read-only from inbound configuration accesses by an external host.

25.4.2.7.3 Diagram



25.4.2.7.4 Fields

Field	Function
23-16 Base_Class	Base Class 0x0B-Processor
15-8 Sub_Class	Sub-Class
7-0 Programming_Interface	Programming_Interface

25.4.2.8 PCI Express Cache Line Size Register (Cache_Line_Size_ Register)

25.4.2.8.1 Offset

Register	Offset
Cache_Line_Size_Register	Ch

25.4.2.8.2 Function

The cache line size register is provided for legacy compatibility purposes (PCI 2.3); it is not used for PCI Express device functionality.

25.4.2.8.3 Diagram

Bits	7	6	5	4	3	2	1	0
R W	Cache_Line_Size							
Reset	0	0	0	0	0	0	0	0

25.4.2.8.4 Fields

Field	Function
7-0 Cache_Line_Size	Cache Line Size Represents the cache line size of the processor in terms of 32-bit words (8 32-bit words = 32 bytes). Note that for PCI Express operation this register is ignored.

25.4.2.9 PCI Express Latency Timer Register (Latency_Timer_Register)

25.4.2.9.1 Offset

Register	Offset
Latency_Timer_Register	Dh

25.4.2.9.2 Function

The latency timer register is provided for legacy compatibility purposes (PCI 2.3); it is not used for PCI Express device functionality.

25.4.2.9.3 Diagram

Bits	7	6	5	4	3	2	1	0
R	Latency_Timer							
W								
Reset	0	0	0	0	0	0	0	0

25.4.2.9.4 Fields

Field	Function
7-0 Latency_Timer	Latency_Timer Note that for PCI Express operation this register is ignored.

25.4.2.10 PCI Express Header Type Register (Header_Type_Register)

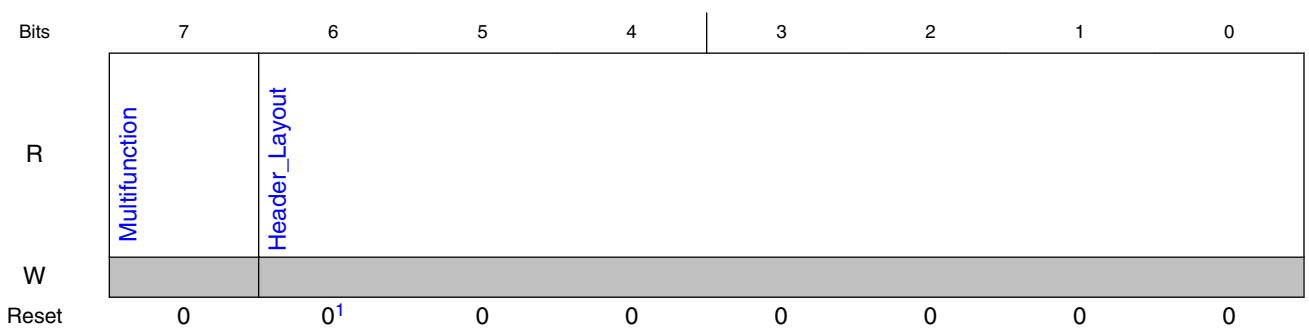
25.4.2.10.1 Offset

Register	Offset
Header_Type_Register	Eh

25.4.2.10.2 Function

The PCI Express header type register is used to identify the layout of the PCI compatible header. The reset value depends on the RC/EP mode configured at power-on reset.

25.4.2.10.3 Diagram



1. EP=00h; RC=01h

25.4.2.10.4 Fields

Field	Function
7 Multifunction	Multifunction Identifies whether a device supports multiple functions 0b - Single function device 1b - Multiple function device

Table continues on the next page...

Field	Function
6-0 Header_Layout	Header Layout All other encodings reserved. 0000000b - Endpoint - Type 0 layout. 0000001b - Root Complex - Type 1 layout.

25.4.2.11 PCI Express Base Address Register 0 (BAR0)

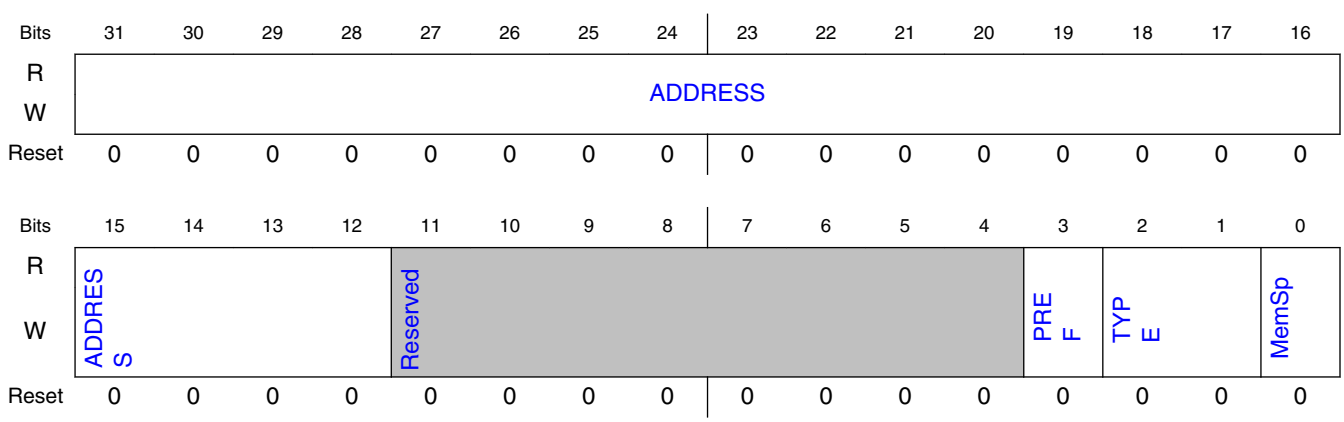
25.4.2.11.1 Offset

Register	Offset
BAR0	10h

25.4.2.11.2 Function

The PCI Express base address registers (BARs) point to the beginning of distinct address ranges which the device should claim.

25.4.2.11.3 Diagram



25.4.2.11.4 Fields

Field	Function
31-12 ADDRESS	Base address Indicates the base address of the inbound memory window 0. The default size is 16 MB.

Table continues on the next page...

Field	Function
11-4 —	Reserved
3 PREF	Prefetchable
2-1 TYPE	Type 00b - Locate anywhere in 32-bit address space.
0 MemSp	Memory space indicator Base Address registers that map to Memory Space must return a 0 in bit 0. Base Address registers that map to I/O Space must return a 1 in bit 0.

25.4.2.12 PCI Express Base Address Register 1 (BAR1)

25.4.2.12.1 Offset

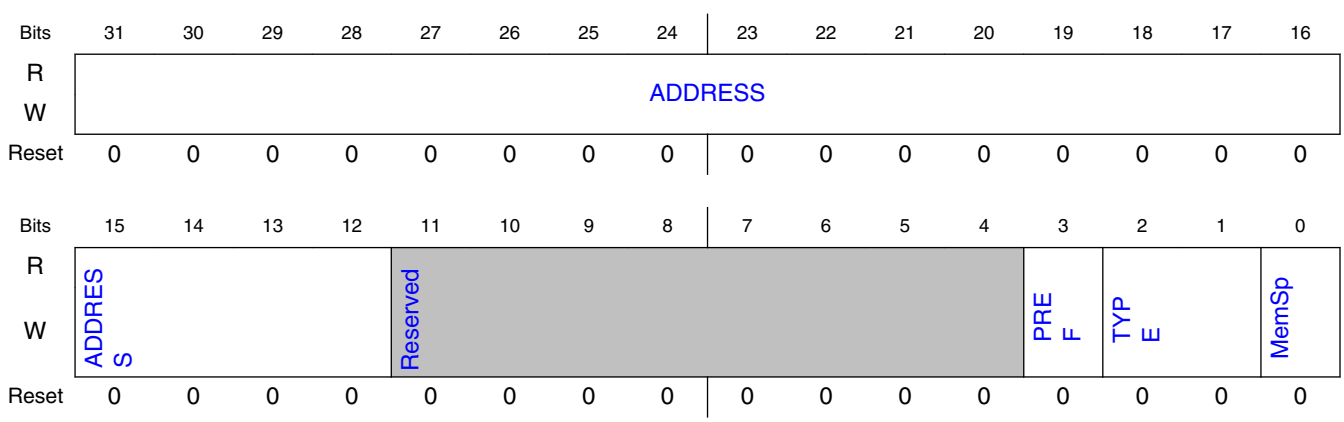
Register	Offset
BAR1	14h

25.4.2.12.2 Function

This register is present only in the Type 0 Header (EP mode).

Base address register 1 at offset 0x14 is used to define the inbound memory window in the 32-bit memory space.

25.4.2.12.3 Diagram



25.4.2.12.4 Fields

Field	Function
31-12 ADDRESS	Base address Indicates the base address where the inbound memory window 1 begins. The default size of this window is 64 MB.
11-4 —	Reserved. The device allows a 4 Kbyte window minimum.
3 PREF	Prefetchable
2-1 TYPE	Type 00b - Locate anywhere in 32-bit address space.
0 MemSp	Memory space indicator Base Address registers that map to Memory Space must return a 0 in bit 0. Base Address registers that map to I/O Space must return a 1 in bit 0.

25.4.2.13 PCI Express Base Address Register 2 (BAR2)

25.4.2.13.1 Offset

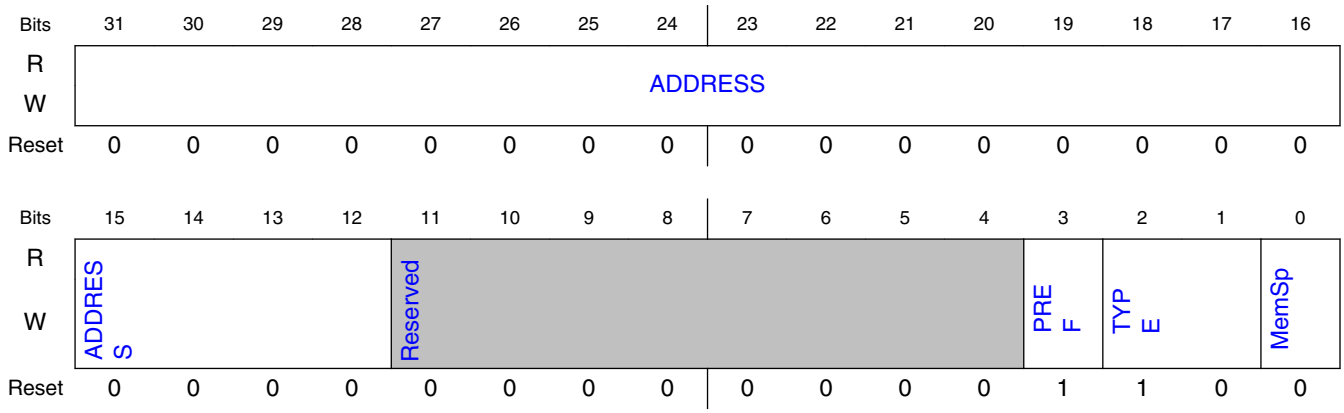
Register	Offset
BAR2	18h

25.4.2.13.2 Function

This register is present only in the Type 0 Header (EP mode).

Together, base address register 2 (BAR2) and base address register 3 (BAR3) define a 64-bit inbound memory window. BAR2 defines the lower portion of the memory window; BAR3 defines the upper portion of the memory window.

25.4.2.13.3 Diagram



25.4.2.13.4 Fields

Field	Function
31-12 ADDRESS	Base address (lower portion) Indicates the lower portion of the base address where the inbound memory window begins.
11-4 —	Reserved. The device allows a 4 Kbyte window minimum.
3 PREF	Prefetchable
2-1 TYPE	Type 00b - Locate anywhere in 32-bit address space. 10b - Locate anywhere in 64-bit address space.
0 MemSp	Memory space indicator Base Address registers that map to Memory Space must return a 0 in bit 0. Base Address registers that map to I/O Space must return a 1 in bit 0.

25.4.2.14 PCI Express Primary Bus Number Register (Primary_Bus_Number_Register)

25.4.2.14.1 Offset

Register	Offset
Primary_Bus_Number_Register	18h

25.4.2.14.2 Function

This register is present only in the Type 1 Header (RC mode).

25.4.2.14.3 Diagram

Bits	7	6	5	4	3	2	1	0
R W	Primary_Bus_Number							
Reset	0	0	0	0	0	0	0	0

25.4.2.14.4 Fields

Field	Function
7-0 Primary_Bus_Number	Primary Bus Number Bus that is connected to the upstream interface. Note that this register is programmed during system enumeration; in RC mode this register should remain 0x00.

25.4.2.15 PCI Express Secondary Bus Number Register (Secondary_Bus_Number_Register)

25.4.2.15.1 Offset

Register	Offset
Secondary_Bus_Number_Register	19h

25.4.2.15.2 Function

This register is present only in the Type 1 Header (RC mode).

25.4.2.15.3 Diagram

Bits	7	6	5	4	3	2	1	0
R W	Secondary_Bus_Number							
Reset	0	0	0	0	0	0	0	0

25.4.2.15.4 Fields

Field	Function
7-0 Secondary_Bus_Number	Secondary Bus Number Bus that is directly connected to the downstream interface. Note that this register is programmed during system enumeration; in RC mode, this register is typically programmed to 0x01.

25.4.2.16 PCI Express Subordinate Bus Number Register (Subordinate_Bus_Number_Register)

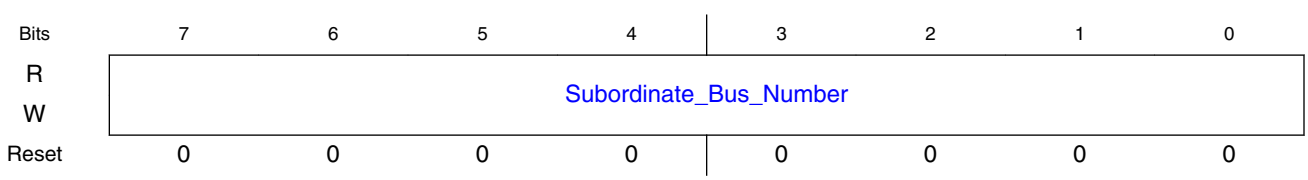
25.4.2.16.1 Offset

Register	Offset
Subordinate_Bus_Number_Register	1Ah

25.4.2.16.2 Function

This register is present only in the Type 1 Header (RC mode).

25.4.2.16.3 Diagram



25.4.2.16.4 Fields

Field	Function
7-0 Subordinate_Bus_Number	Subordinate Bus Number Highest bus number that is on the downstream interface.

25.4.2.17 PCI Express Base Address Register 3 (BAR3)

25.4.2.17.1 Offset

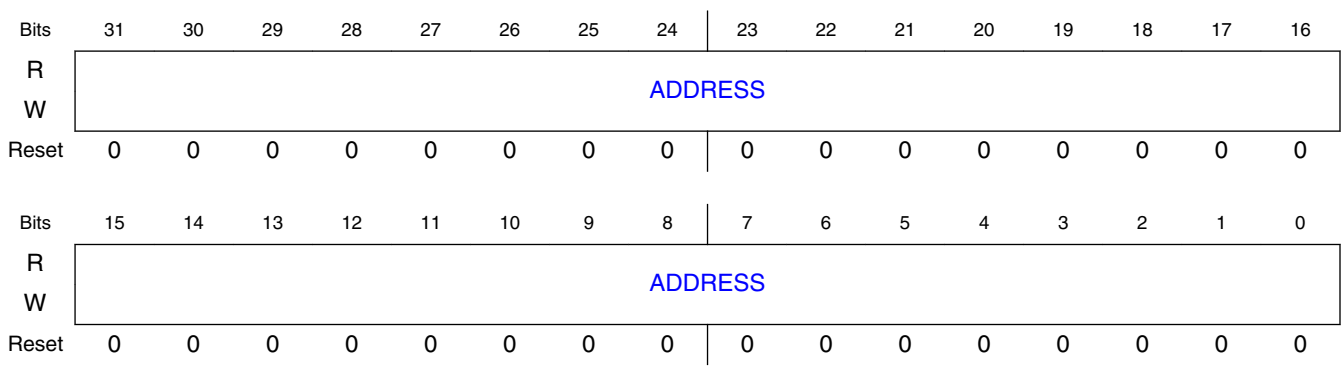
Register	Offset
BAR3	1Ch

25.4.2.17.2 Function

This register is present only in the Type 0 Header (EP mode).

Together, base address register 2 (BAR2) and base address register 3 (BAR3) define a 64-bit inbound memory window. BAR2 defines the lower portion of the memory window; BAR3 defines the upper portion of the memory window.

25.4.2.17.3 Diagram



25.4.2.17.4 Fields

Field	Function
31-0 ADDRESS	Base address (upper portion) Indicates the upper portion of the base address where the inbound memory window begins. If no access to local memory is to be permitted by external requesters, then all bits are programmed.

25.4.2.18 PCI Express I/O Base Register (IO_Base_Register)

25.4.2.18.1 Offset

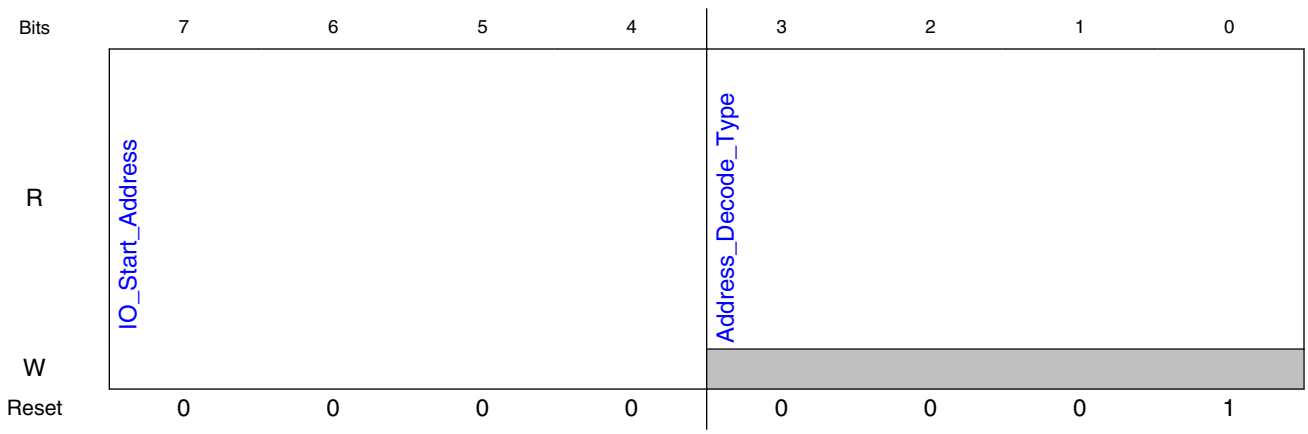
Register	Offset
IO_Base_Register	1Ch

25.4.2.18.2 Function

This register is present only in the Type 1 Header (RC mode).

Note that this device does not support inbound I/O transactions.

25.4.2.18.3 Diagram



25.4.2.18.4 Fields

Field	Function
7-4 IO_Start_Address	I/O Start Address Specifies bits 15:12 of the I/O space start address
3-0 Address_Decode_Type	Address Decode Type Specifies the number of I/O address bits. All other settings are reserved. 0000b - 16-bit I/O address decode 0001b - 32-bit I/O address decode

25.4.2.19 PCI Express I/O Limit Register (IO_Limit_Register)

25.4.2.19.1 Offset

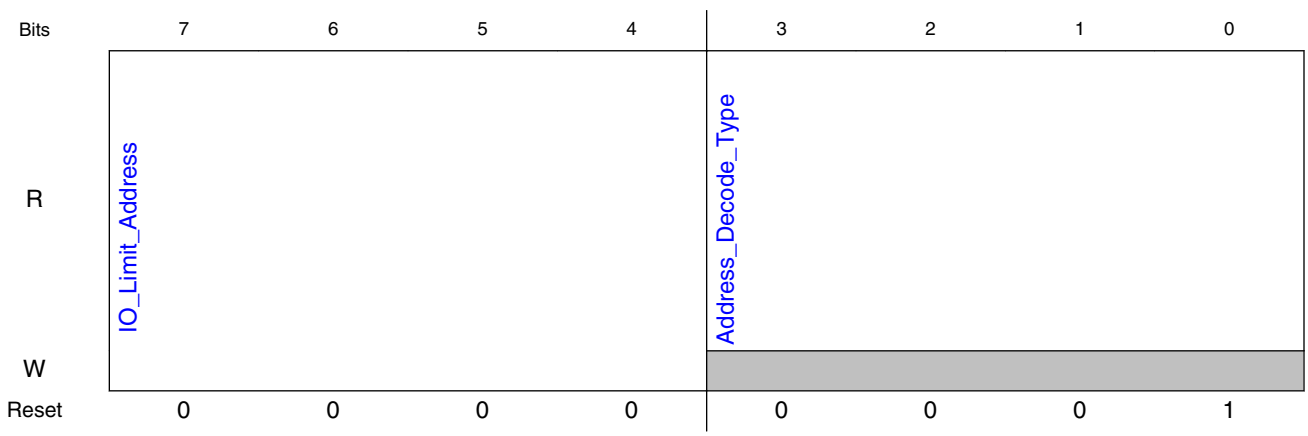
Register	Offset
IO_Limit_Register	1Dh

25.4.2.19.2 Function

This register is present only in the Type 1 Header (RC mode).

Note that this device does not support inbound I/O transactions.

25.4.2.19.3 Diagram



25.4.2.19.4 Fields

Field	Function
7-4 IO_Limit_Address	I/O Limit Address Specifies bits 15:12 of the I/O space ending address
3-0 Address_Decode_Type	Address Decode Type Specifies the number of I/O address bits. All other settings are reserved. 0000b - 16-bit I/O address decode 0001b - 32-bit I/O address decode

25.4.2.20 PCI Express Secondary Status Register (Secondary_Status_Register)

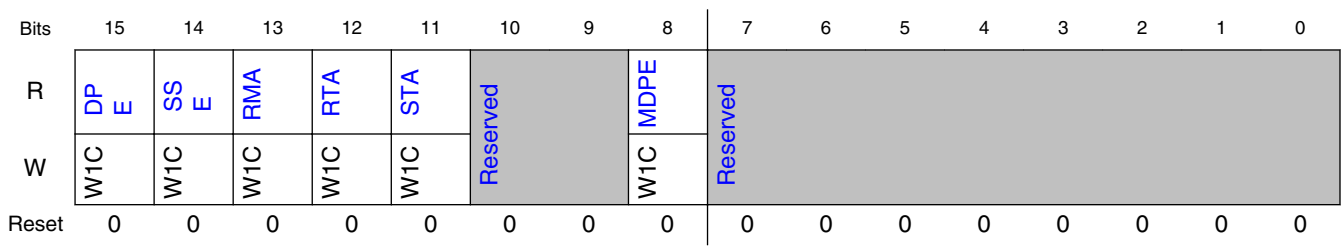
25.4.2.20.1 Offset

Register	Offset
Secondary_Status_Register	1Eh

25.4.2.20.2 Function

This register is present only in the Type 1 Header (RC mode).

25.4.2.20.3 Diagram



25.4.2.20.4 Fields

Field	Function
15 DPE	Detected parity error This bit is set whenever the secondary side receives a poisoned TLP regardless of the state of the parity error response bit.
14 SSE	Signaled system error This bit is set when a device sends a ERR_FATAL or ERR_NONFATAL message, provided the SERR enable bit in the command register is set to enable reporting.
13 RMA	Received master abort This bit is set when the secondary side receives an unsupported request (UR) completion.
12 RTA	Received target abort This bit is set when the secondary side receives a completer abort (CA) completion.
11 STA	Signaled target abort This bit is set when the secondary side issues a CA completion.
10-9	Reserved

Table continues on the next page...

Field	Function
—	
8 MDPE	Master data parity error This bit is set when the parity error response bit is set and the secondary side requester receives a poisoned completion or poisons a write request. If the parity error response bit is cleared, this bit is never set.
7-0 —	Reserved

25.4.2.21 PCI Express Base Address Register 4 (BAR4)

25.4.2.21.1 Offset

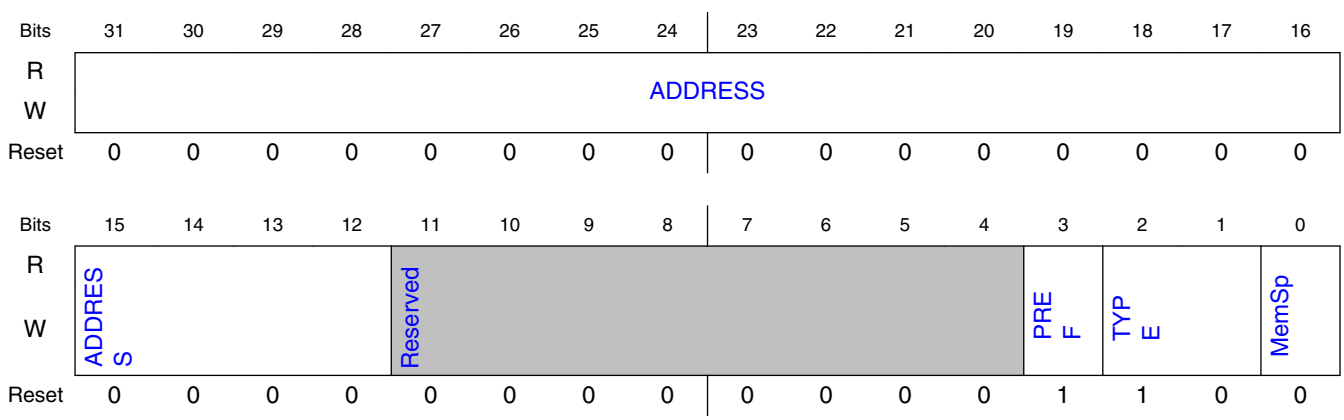
Register	Offset
BAR4	20h

25.4.2.21.2 Function

This register is present only in the Type 0 Header (EP mode).

Together, base address register 4 (BAR4) and base address register 5 (BAR5) define a 64-bit inbound memory window. BAR4 defines the lower portion of the memory window; BAR5 defines the upper portion of the memory window.

25.4.2.21.3 Diagram



25.4.2.21.4 Fields

Field	Function
31-12 ADDRESS	Base address (lower portion) Indicates the lower portion of the base address where the inbound memory window begins.
11-4 —	Reserved. The device allows a 4 Kbyte window minimum.
3 PREF	Prefetchable
2-1 TYPE	Type 00b - Locate anywhere in 32-bit address space. 10b - Locate anywhere in 64-bit address space.
0 MemSp	Memory space indicator Base Address registers that map to Memory Space must return a 0 in bit 0. Base Address registers that map to I/O Space must return a 1 in bit 0.

25.4.2.22 PCI Express Memory Base Register (Memory_Base_Register)

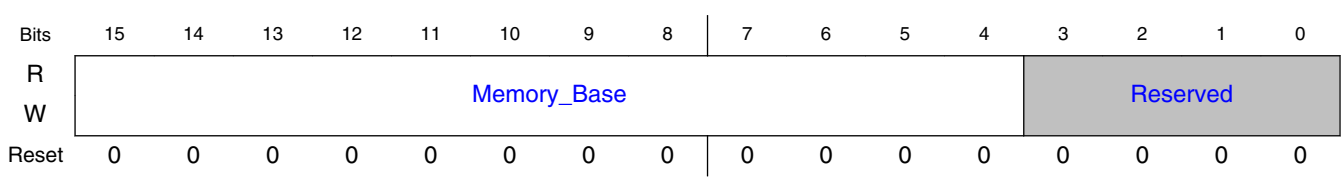
25.4.2.22.1 Offset

Register	Offset
Memory_Base_Register	20h

25.4.2.22.2 Function

This register is present only in the Type 1 Header (RC mode).

25.4.2.22.3 Diagram



25.4.2.22.4 Fields

Field	Function
15-4 Memory_Base	Memory base address Specifies bits 31:20 of the non-prefetchable memory space start address. Typically used for specifying memory-mapped I/O space. **NOTE:** Inbound posted transactions hitting into the mem base/limit range are ignored; inbound non-posted transactions hitting into the mem base/limit range results in an unsupported request response.
3-0 —	Reserved

25.4.2.23 PCI Express Memory Limit Register (Memory_Limit_Register)

25.4.2.23.1 Offset

Register	Offset
Memory_Limit_Register	22h

25.4.2.23.2 Function

This register is present only in the Type 1 Header (RC mode).

25.4.2.23.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Memory_Limit												Reserved			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

25.4.2.23.4 Fields

Field	Function
15-4 Memory_Limit	Memory limit address Specifies bits 31:20 of the non-prefetchable memory space ending address. Typically used for specifying memory-mapped I/O space.

Table continues on the next page...

Field	Function
	NOTE: Inbound posted transactions hitting into the mem base/limit range are ignored; inbound non-posted transactions hitting into the mem base/limit range results in unsupported request response.
3-0 —	Reserved

25.4.2.24 PCI Express Base Address Register 5 (BAR5)

25.4.2.24.1 Offset

Register	Offset
BAR5	24h

25.4.2.24.2 Function

This register is present only in the Type 0 Header (EP mode).

Together, base address register 4 (BAR4) and base address register 5 (BAR5) define a 64-bit inbound memory window. BAR4 defines the lower portion of the memory window; BAR5 defines the upper portion of the memory window.

25.4.2.24.3 Diagram

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

Bits	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

25.4.2.24.4 Fields

Field	Function
31-0	Base address (upper portion)

Field	Function
ADDRESS	Indicates the upper portion of the base address where the inbound memory window begins. If no access to local memory is to be permitted by external requestors, then all bits are programmed.

25.4.2.25 PCI Express Prefetchable Memory Base Register (Prefetchable_Memory_Base_Register)

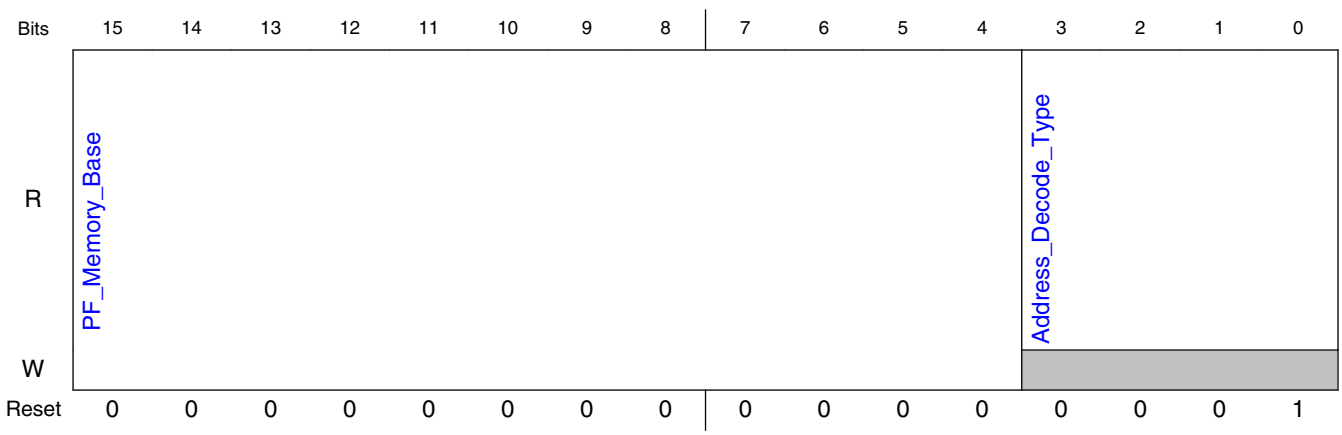
25.4.2.25.1 Offset

Register	Offset
Prefetchable_Memory_Base_Register	24h

25.4.2.25.2 Function

This register is present only in the Type 1 Header (RC mode).

25.4.2.25.3 Diagram



25.4.2.25.4 Fields

Field	Function
15-4 PF_Memory_Base	Prefetchable memory base address Specifies bits 31:20 of the prefetchable memory space start address.
3-0	Address Decode Type

Field	Function
Address_Decode_Type	Specifies the number of prefetchable memory address bits. All other settings reserved. 0000b - 32-bit memory address decode 0001b - 64-bit memory address decode

25.4.2.26 PCI Express Prefetchable Memory Limit Register (Prefetchable_Memory_Limit_Register)

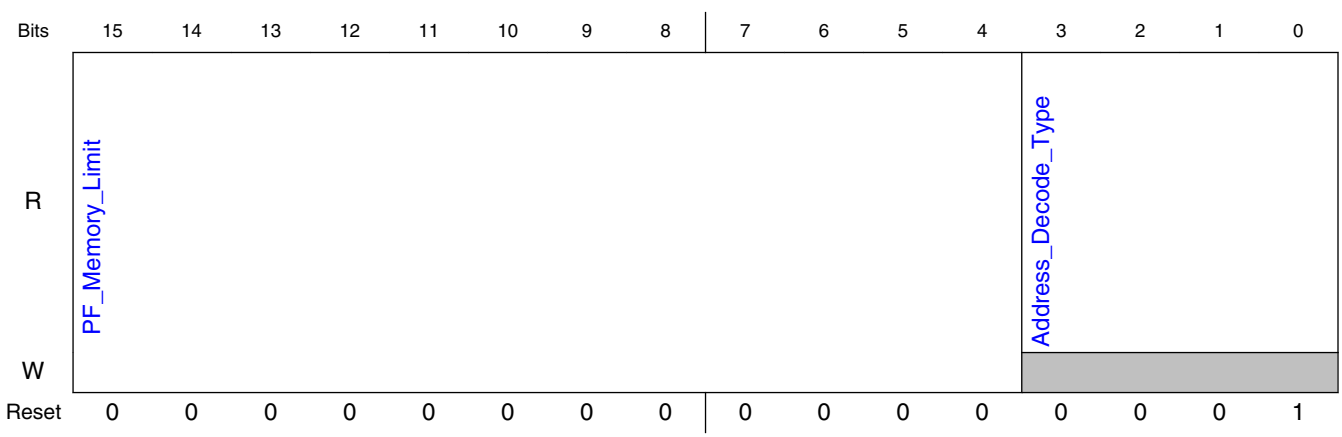
25.4.2.26.1 Offset

Register	Offset
Prefetchable_Memory_Limit_Register	26h

25.4.2.26.2 Function

This register is present only in the Type 1 Header (RC mode).

25.4.2.26.3 Diagram



25.4.2.26.4 Fields

Field	Function
15-4	Prefetchable memory limit address Specifies bits 31:20 of the prefetchable memory space ending address.

Table continues on the next page...

Field	Function
PF_Memory_Limit	
3-0 Address_Decode_Type	Address decode type Specifies the number of prefetchable memory address bits. All other settings reserved. 0000b - 32-bit memory address decode 0001b - 64-bit memory address decode

25.4.2.27 PCI Express Prefetchable Base Upper 32 Bits Register (Prefetchable_Base_Upper_32_Bits_Register)

25.4.2.27.1 Offset

Register	Offset
Prefetchable_Base_Upper_32_Bits_Register	28h

25.4.2.27.2 Function

This register is present only in the Type 1 Header (RC mode).

25.4.2.27.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PF_Base_Upper_32_Bits															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PF_Base_Upper_32_Bits															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

25.4.2.27.4 Fields

Field	Function
31-0	Prefetchable memory base address (upper portion)

Field	Function
PF_Base_Upper_32_Bits	Specifies bits 64:32 of the prefetchable memory space start address when the address decode type field in the prefetchable memory base register is 0x01.

25.4.2.28 PCI Express Prefetchable Limit Upper 32 Bits Register (Prefetchable_Limit_Upper_32_Bits_Register)

25.4.2.28.1 Offset

Register	Offset
Prefetchable_Limit_Upper_32_Bits_Register	2Ch

25.4.2.28.2 Function

This register is present only in the Type 1 Header (RC mode).

25.4.2.28.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PF_Limit_Upper_32_Bits															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PF_Limit_Upper_32_Bits															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

25.4.2.28.4 Fields

Field	Function
31-0 PF_Limit_Upper_32_Bits	Prefetchable memory limit address (upper portion) Specifies bits 64-32 of the prefetchable memory space ending address when the address decode type field in the prefetchable memory limit register is 0x01.

25.4.2.29 PCI Express Subsystem Vendor ID Register (Subsystem_Vendor_ID_Register)

25.4.2.29.1 Offset

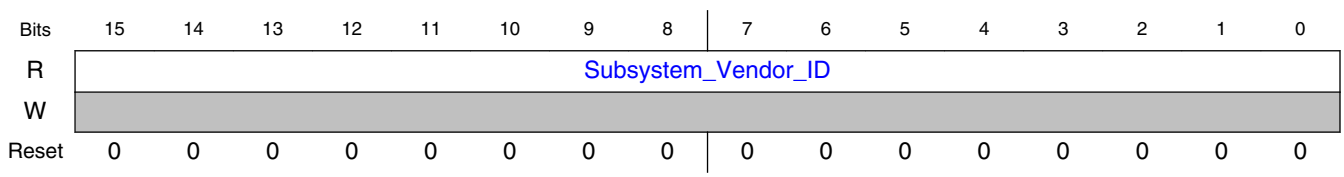
Register	Offset
Subsystem_Vendor_ID_Register	2Ch

25.4.2.29.2 Function

This register is present only in the Type 0 Header (EP mode).

The PCI Express subsystem vendor ID register is used to identify the subsystem.

25.4.2.29.3 Diagram



25.4.2.29.4 Fields

Field	Function
15-0 Subsystem_Vendor_ID	Subsystem Vendor ID This register is present only in the Type 0 Header (EP mode). The PCI Express subsystem vendor ID register is used to identify the subsystem vendor.

25.4.2.30 PCI Express Subsystem ID Register (Subsystem_ID_Register)

25.4.2.30.1 Offset

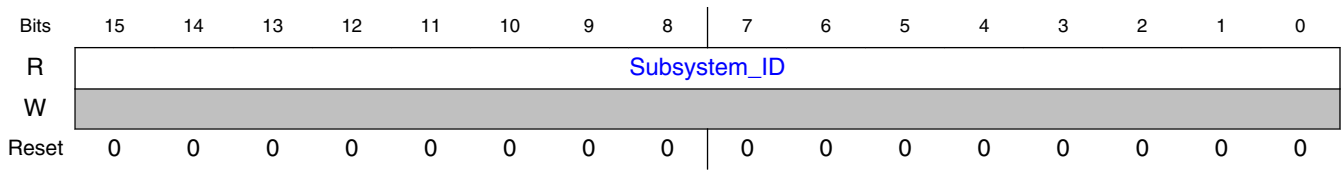
Register	Offset
Subsystem_ID_Register	2Eh

25.4.2.30.2 Function

This register is present only in the Type 0 Header (EP mode).

The PCI Express subsystem ID register is used to identify the subsystem.

25.4.2.30.3 Diagram



25.4.2.30.4 Fields

Field	Function
15-0 Subsystem_ID	Subsystem ID

25.4.2.31 PCI Express Expansion ROM Base Address Register (EP-Mode) (Expansion_ROM_BAR_Type0)

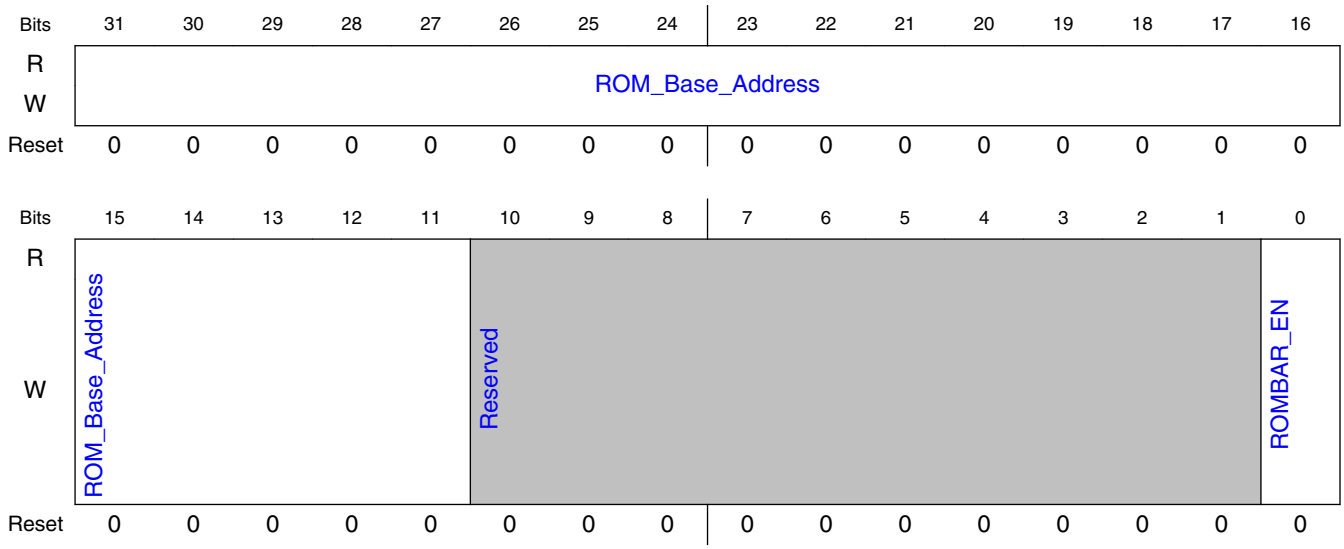
25.4.2.31.1 Offset

Register	Offset
Expansion_ROM_BAR_Type0	30h

25.4.2.31.2 Function

The Expansion ROM Base Address register is located at offset 0x30 in the Type 0 Header (EP mode); it is located at offset 0x38 in the Type 1 Header (RC mode).

25.4.2.31.3 Diagram



25.4.2.31.4 Fields

Field	Function
31-11 ROM_Base_Address	Expansion ROM base address Specifies bits 31:11 of the non-prefetchable expansion ROM space start address. Typically used for specifying memory-mapped I/O space. The default size is 16M.
10-1 —	Reserved
0 ROMBAR_EN	Expansion ROM enable This bit controls whether or not the device accepts accesses to its expansion ROM 0b - The expansion ROM address space is disabled. 1b - Address decoding is enabled.

25.4.2.32 PCI Express I/O Base Upper 16 Bits Register (IO_Base_Upper_16_Bits_Register)

25.4.2.32.1 Offset

Register	Offset
IO_Base_Upper_16_Bits_Register	30h

25.4.2.32.2 Function

This register is present only in the Type 1 Header (RC mode).

Note that this device does not support inbound I/O transactions.

25.4.2.32.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IO_Base_Upper_16_Bits															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

25.4.2.32.4 Fields

Field	Function
15-0 IO_Base_Upper_16_Bits	I/O base address (upper portion) Specifies bits 31-16 of the I/O space start address when the address decode type field in the I/O base register is 0x01.

25.4.2.33 PCI Express I/O Limit Upper 16 Bits Register (IO_Limit_Upper_16_Bits_Register)

25.4.2.33.1 Offset

Register	Offset
IO_Limit_Upper_16_Bits_Register	32h

25.4.2.33.2 Function

This register is present only in the Type 1 Header (RC mode).

Note that this device does not support inbound I/O transactions.

25.4.2.33.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IO_Limit_Upper_16_Bits															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

25.4.2.33.4 Fields

Field	Function
15-0 IO_Limit_Upper_16_Bits	I/O limit address (upper portion) Specifies bits 31-16 of the I/O space ending address when the address decode type field in the I/O limit register is 0x01.

25.4.2.34 Capabilities Pointer Register (Capabilities_Pointer_Register)

25.4.2.34.1 Offset

Register	Offset
Capabilities_Pointer_Register	34h

25.4.2.34.2 Function

The capabilities pointer identifies additional functionality supported by the device.

25.4.2.34.3 Diagram

Bits	7	6	5	4	3	2	1	0
R	Capabilities_Pointer							
W								
Reset	0	1	0	0	0	0	0	0

25.4.2.34.4 Fields

Field	Function
7-0 Capabilities_Pointer	Capabilities Pointer The capabilities pointer provides the offset for additional PCI-compatible registers above the common 64-byte header.

25.4.2.35 PCI Express Expansion ROM Base Address Register (RC-Mode) (Expansion_ROM_BAR_Type1)

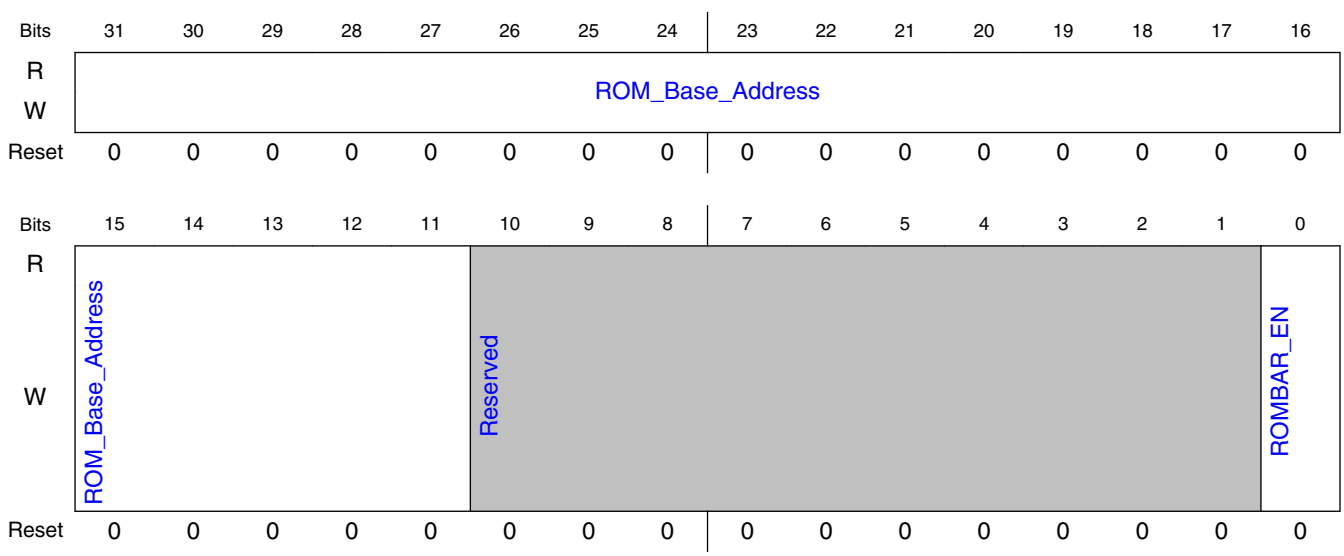
25.4.2.35.1 Offset

Register	Offset
Expansion_ROM_BAR_Type1	38h

25.4.2.35.2 Function

The Expansion ROM Base Address register is located at offset 0x38 in the Type 1 Header (RC mode); it is located at offset 0x30 in the Type 0 Header (EP mode).

25.4.2.35.3 Diagram



25.4.2.35.4 Fields

Field	Function
31-11 ROM_Base_Address	Expansion ROM base address Specifies bits 31:11 of the non-prefetchable expansion ROM space start address. Typically used for specifying memory-mapped I/O space. The default size is 16M.
10-1 —	Reserved
0 ROMBAR_EN	Expansion ROM enable This bit controls whether or not the device accepts accesses to its expansion ROM 0b - The expansion ROM address space is disabled. 1b - Address decoding is enabled.

25.4.2.36 PCI Express Interrupt Line Register (Interrupt_Line_Register)

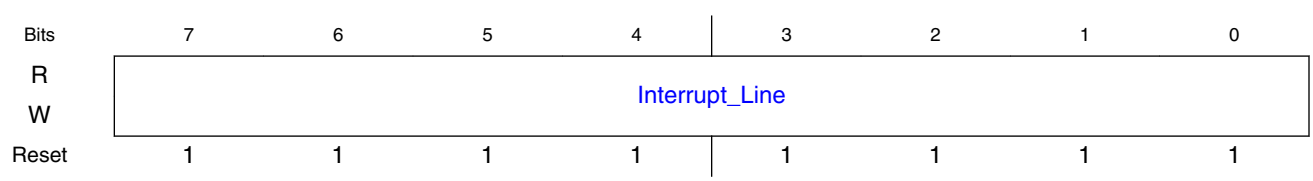
25.4.2.36.1 Offset

Register	Offset
Interrupt_Line_Register	3Ch

25.4.2.36.2 Function

The interrupt line register is used by device drivers and OS software to communicate interrupt line routing information. Values in this register are programmed by system software and are system specific.

25.4.2.36.3 Diagram



25.4.2.36.4 Fields

Field	Function
7-0 Interrupt_Line	Interrupt line Used to communicate interrupt line routing information.

25.4.2.37 PCI Express Interrupt Pin Register (Interrupt_Pin_Register)

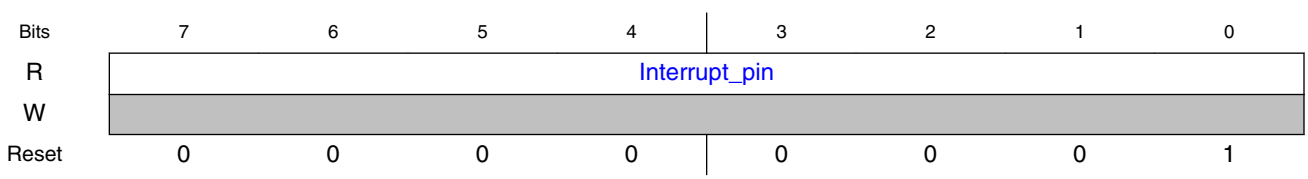
25.4.2.37.1 Offset

Register	Offset
Interrupt_Pin_Register	3Dh

25.4.2.37.2 Function

The interrupt pin register identifies the legacy interrupt (INTx) messages the device (or function) uses.

25.4.2.37.3 Diagram



25.4.2.37.4 Fields

Field	Function
7-0 Interrupt_pin	Interrupt pin Legacy INTx message used by this device. All other settings reserved. 00000000b - This device does not use legacy interrupt (INTx) messages. 00000001b - INTA

25.4.2.38 PCI Express Bridge Control Register (Bridge_Control_Register)

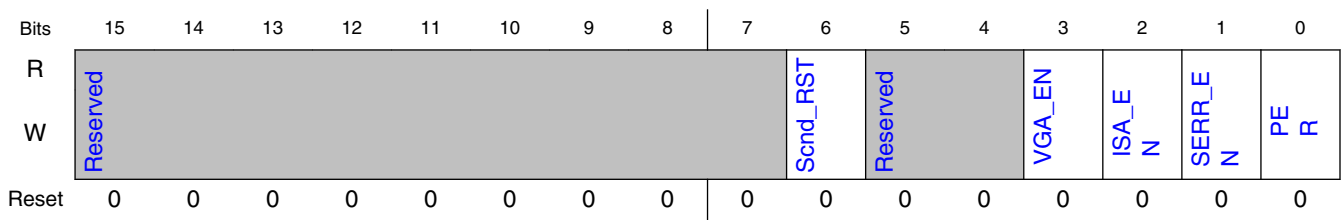
25.4.2.38.1 Offset

Register	Offset
Bridge_Control_Register	3Eh

25.4.2.38.2 Function

This register is present only in the Type 1 Header (RC mode).

25.4.2.38.3 Diagram



25.4.2.38.4 Fields

Field	Function
15-7 —	Reserved
6 Scnd_RST	Secondary bus reset
5-4 —	Reserved
3 VGA_EN	VGA enable
2 ISA_EN	ISA enable
1 SERR_EN	SERR enable This bit controls the propagation of ERR_COR, ERR_NONFATAL, and ERR_FATAL responses received on the secondary side.
0 PER	Parity error response This bit controls the logging of poisoned TLPs in the Master Data Parity Error bit in the Secondary Status register.

25.4.2.39 PCI Express Minimum Grant Register (Minimum_Grant_Register)

25.4.2.39.1 Offset

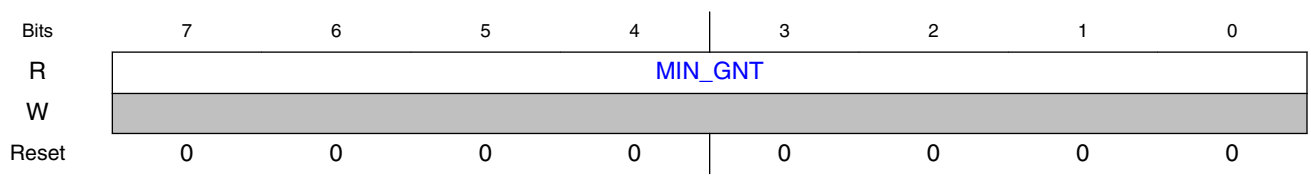
Register	Offset
Minimum_Grant_Register	3Eh

25.4.2.39.2 Function

This register is present only in the Type 0 Header (EP mode).

This register does not apply to PCI Express. It is present for legacy purposes.

25.4.2.39.3 Diagram



25.4.2.39.4 Fields

Field	Function
7-0 MIN_GNT	Minimum grant Does not apply for PCI Express.

25.4.2.40 PCI Express Maximum Latency Register (Maximum_Latency_Register)

25.4.2.40.1 Offset

Register	Offset
Maximum_Latency_Register	3Fh

25.4.2.40.2 Function

This register is present only in the Type 0 Header (EP mode).

This register does not apply to PCI Express. It is present for legacy purposes.

25.4.2.40.3 Diagram

Bits	7	6	5	4	3	2	1	0
R	MAX_LAT							
W								
Reset	0	0	0	0	0	0	0	0

25.4.2.40.4 Fields

Field	Function
7-0 MAX_LAT	Maximum latency Does not apply for PCI Express.

25.4.2.41 PCI Express Power Management Capability ID Register (Power_Management_Capability_ID_Register)

25.4.2.41.1 Offset

Register	Offset
Power_Management_ Capability_ID_Register	40h

25.4.2.41.2 Diagram

Bits	7	6	5	4	3	2	1	0
R	Power_Mgmt_Capability_ID							
W								
Reset	0	0	0	0	0	0	0	1

25.4.2.41.3 Fields

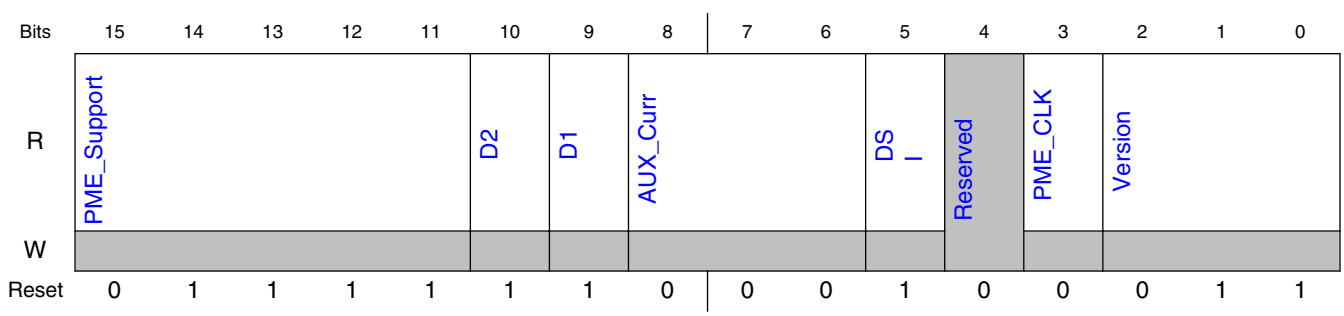
Field	Function
7-0 Power_Mgmt_Capability_ID	CAP_ID Power Management = 0x01

25.4.2.42 PCI Express Power Management Capabilities Register (Power_Management_Capabilities_Register)

25.4.2.42.1 Offset

Register	Offset
Power_Management_Capabilities_Register	42h

25.4.2.42.2 Diagram



25.4.2.42.3 Fields

Field	Function
15-11 PME_Support	PME support Indicates the power states that this device supports
10 D2	D2 support
9 D1	D1 support
8-6	AUX Current

Table continues on the next page...

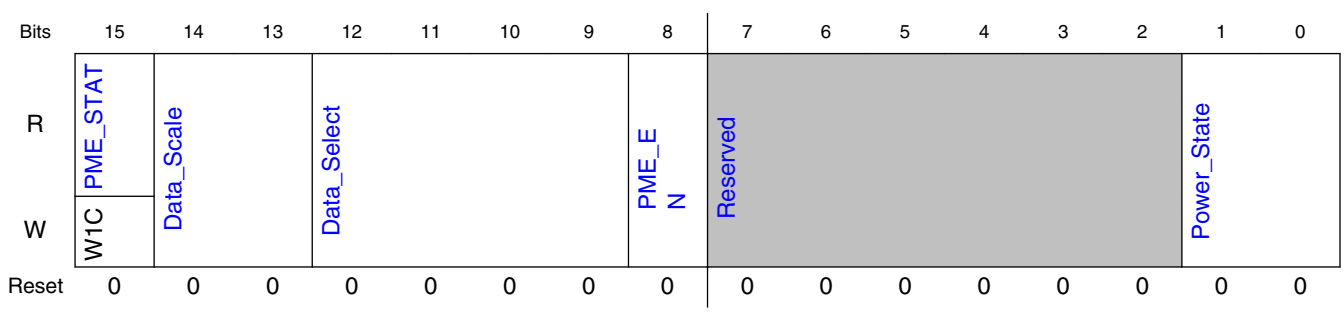
Field	Function
AUX_Curr	
5 DSI	Device Specific Initialization
4 —	Reserved
3 PME_CLK	PME clock Does not apply to PCI Express.
2-0 Version	Version

25.4.2.43 PCI Express Power Management Status and Control Register (Power_Management_Status_and_Control_Register)

25.4.2.43.1 Offset

Register	Offset
Power_Management_Status_and_Control_Register	44h

25.4.2.43.2 Diagram



25.4.2.43.3 Fields

Field	Function
15	PME Status

Table continues on the next page...

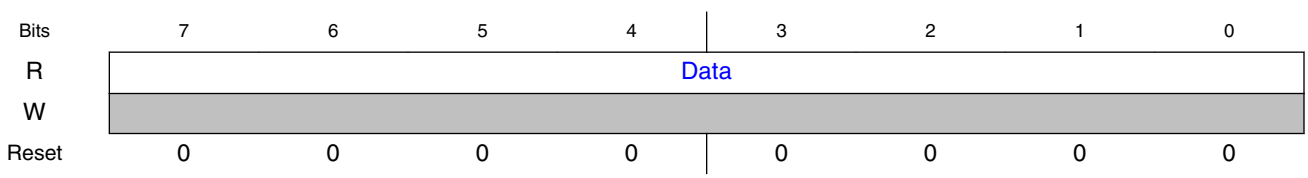
Field	Function
PME_STAT	
14-13 Data_Scale	Data Scale Obtained directly from the PCI Express base specification.
12-9 Data_Select	Data Select Obtained directly from the PCI Express base specification.
8 PME_EN	PME_En PME Enable. **NOTE:** Bitfield access is sticky.
7-2 —	Reserved
1-0 Power_State	Power State Indicates the current power state of the function. 00b - D0 01b - D1 10b - D2 11b - D3

25.4.2.44 PCI Express Power Management Data Register (Power_Management_Data_Register)

25.4.2.44.1 Offset

Register	Offset
Power_Management_Data_Register	47h

25.4.2.44.2 Diagram



25.4.2.44.3 Fields

Field	Function
7-0 Data	Data Obtained from the PCI Express base specification.

25.4.2.45 PCI Express MSI Message Capability ID Register (MSI_Message_Capability_ID_Register)

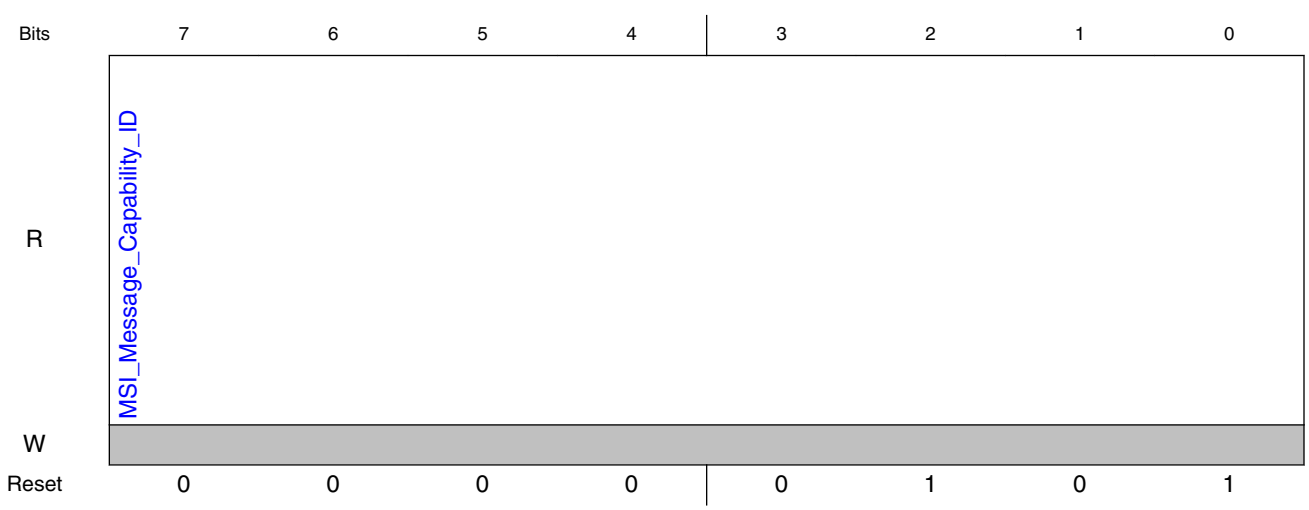
25.4.2.45.1 Offset

Register	Offset
MSI_Message_Capability_ID_Register	50h

25.4.2.45.2 Function

This register is supported only for EP mode.

25.4.2.45.3 Diagram



25.4.2.45.4 Fields

Field	Function
7-0 MSI_Message_Capability_ID	CAP_ID MSI Message = 0x05

25.4.2.46 PCI Express MSI Message Control Register (MSI_Message_Control_Register)

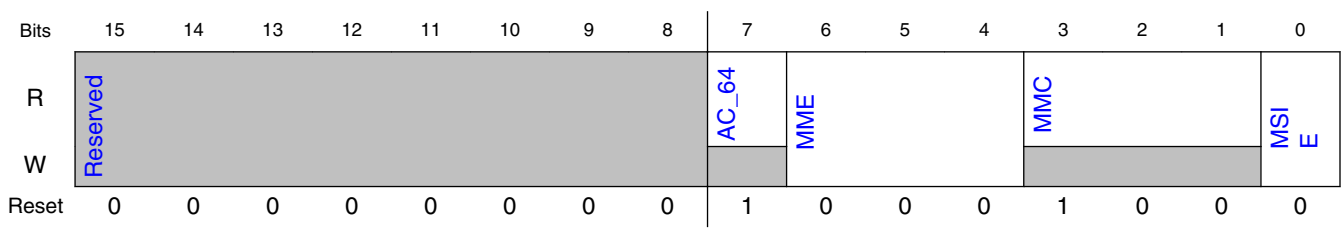
25.4.2.46.1 Offset

Register	Offset
MSI_Message_Control_Register	52h

25.4.2.46.2 Function

This register is supported only for EP mode.

25.4.2.46.3 Diagram



25.4.2.46.4 Fields

Field	Function
15-8 —	Reserved
7 AC_64	64-bit address capable
6-4	Multiple message enable

Table continues on the next page...

Field	Function
MME	
3-1 MMC	Multiple message capable
0 MSIE	MSI enable

25.4.2.47 PCI Express MSI Message Address Register (MSI_Message_Address_Register)

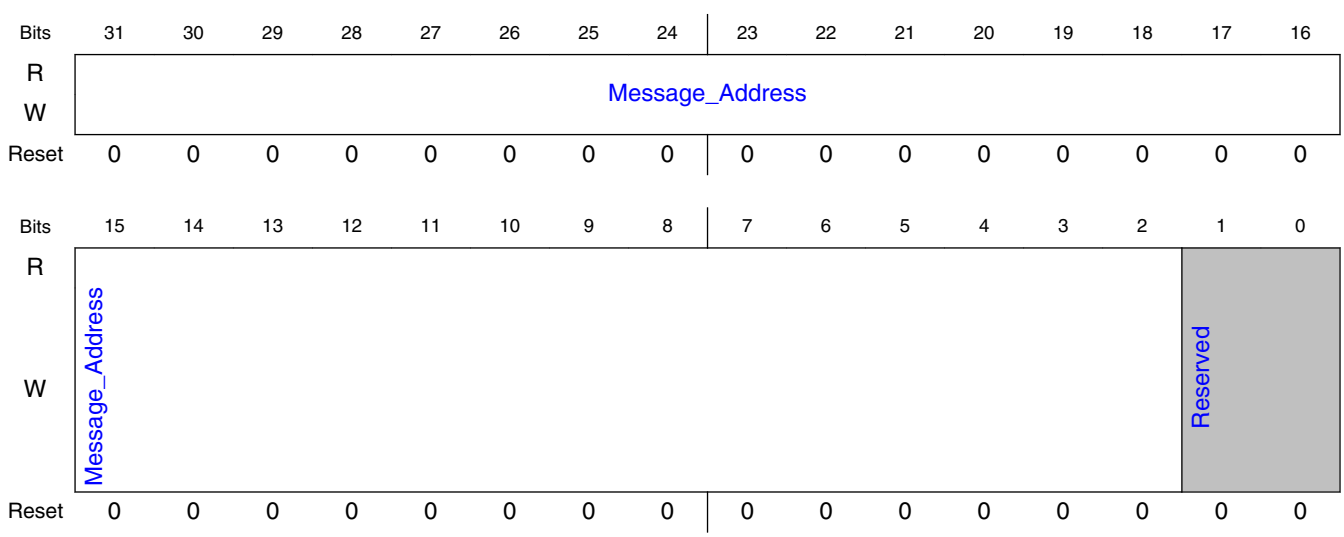
25.4.2.47.1 Offset

Register	Offset
MSI_Message_Address_Register	54h

25.4.2.47.2 Function

This register is supported only for EP mode.

25.4.2.47.3 Diagram



25.4.2.47.4 Fields

Field	Function
31-2 Message_Address	Message Address System-specified message address
1-0 —	Reserved. Always returns 00 on reads; write operations have no effect.

25.4.2.48 PCI Express MSI Message Upper Address Register (MSI_Message_Upper_Address_Register)

25.4.2.48.1 Offset

Register	Offset
MSI_Message_Upper_Address_Register	58h

25.4.2.48.2 Function

This register is supported only for EP mode.

25.4.2.48.3 Diagram

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

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

25.4.2.48.4 Fields

Field	Function
31-0	Message Address (upper portion)

Field	Function
Message_Upper_Address	System-specified message upper address

25.4.2.49 PCI Express MSI Message Data Register (MSI_Message_Data_Register)

25.4.2.49.1 Offset

Register	Offset
MSI_Message_Data_Register	5Ch

25.4.2.49.2 Function

This register is supported only for EP mode.

25.4.2.49.3 Diagram

25.4.2.49.4 Fields

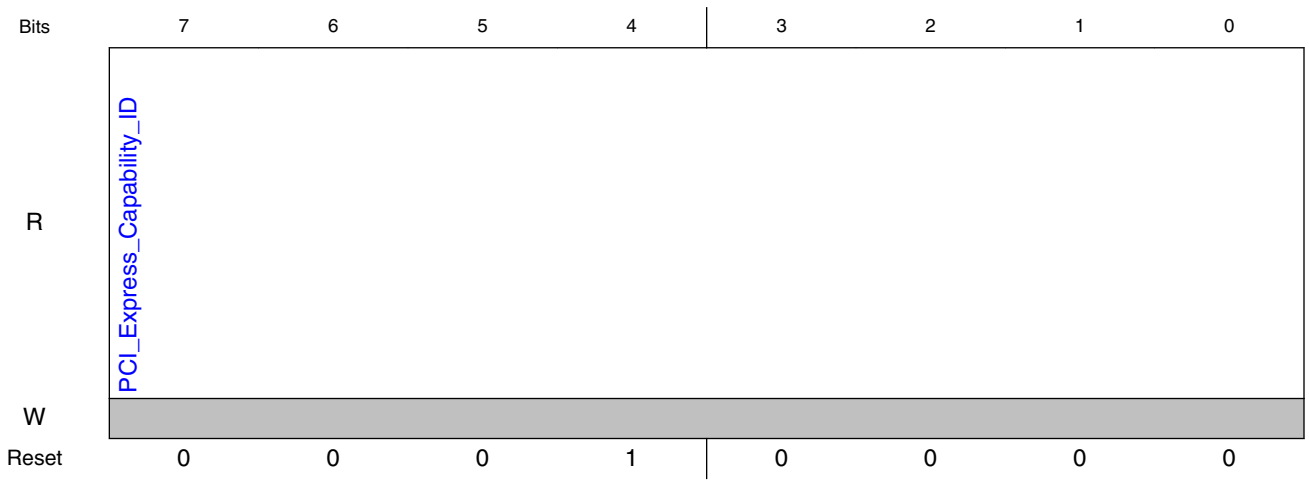
Field	Function
15-0 Message_Data	Data System-specified message data.

25.4.2.50 PCI Express Capability ID Register (Capability_ID_Register)

25.4.2.50.1 Offset

Register	Offset
Capability_ID_Register	70h

25.4.2.50.2 Diagram



25.4.2.50.3 Fields

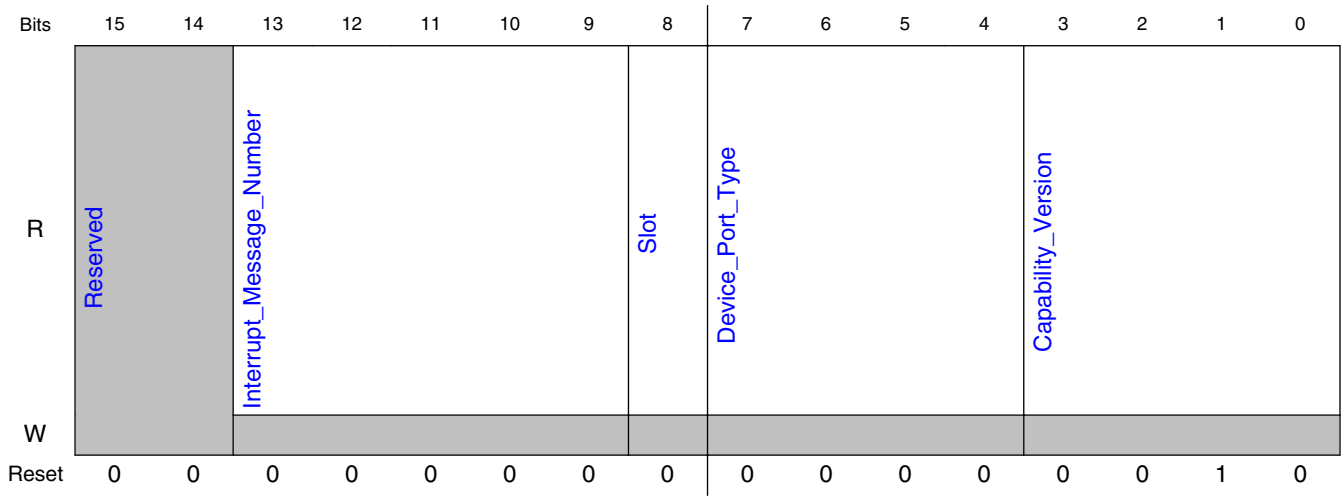
Field	Function
7-0 PCI_Express_Capability_ID	Capability ID PCI Express = 0x10

25.4.2.51 PCI Express Capabilities Register (Capabilities_Register)

25.4.2.51.1 Offset

Register	Offset
Capabilities_Register	72h

25.4.2.51.2 Diagram



25.4.2.51.3 Fields

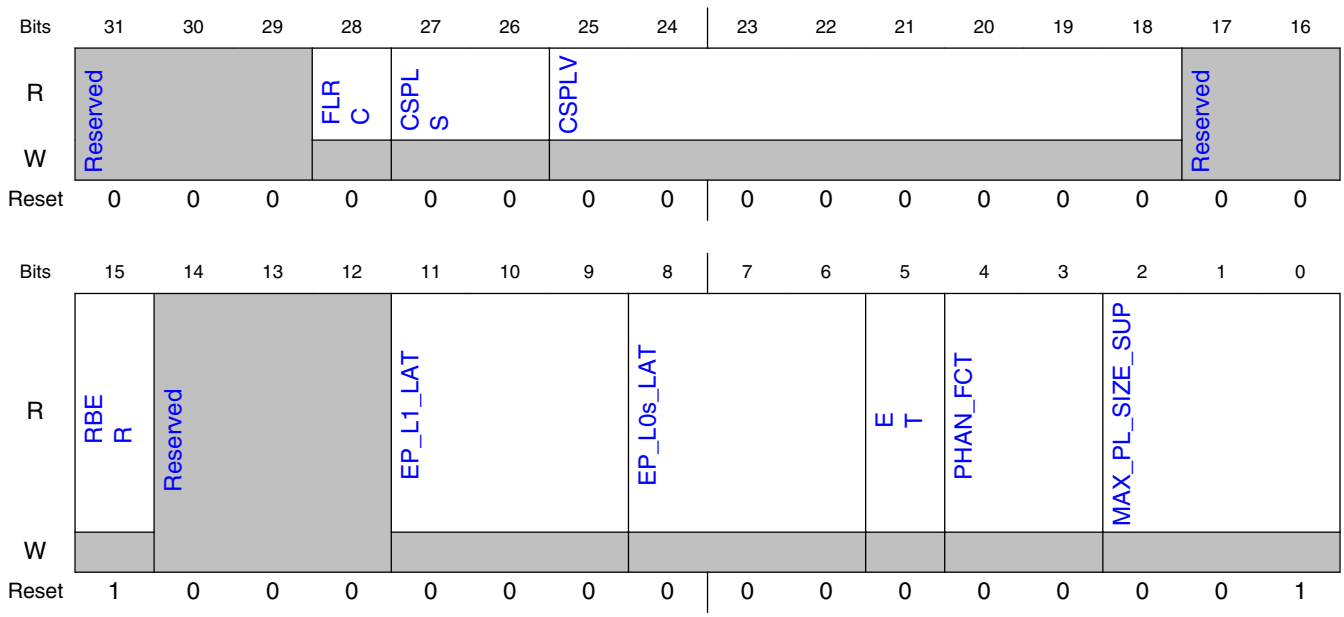
Field	Function
15-14 —	Reserved
13-9 Interrupt_Message_Number	Interrupt Message Number If this function is allocated more than one MSI interrupt number, then this register is required to contain the offset between the base Message Data and the MSI Message that is generated when any of the status bits in either the Slot Status register or the Root Port Status register, of this capability structure, are set.
8 Slot	Slot Implemented (RC mode only)
7-4 Device_Port_Type	Device/Port Type 0000b - (EP mode) 0100b - (RC mode)
3-0 Capability_Version	Capability Version Indicates the defined PCI Express capability structure version number.

25.4.2.52 PCI Express Device Capabilities Register (Device_Capabilities_Register)

25.4.2.52.1 Offset

Register	Offset
Device_Capabilities_Register	74h

25.4.2.52.2 Diagram



25.4.2.52.3 Fields

Field	Function
31-29 —	Reserved
28 FLRC	Function Level Reset Capability For RC mode, the reset value for this bit is 0; for EP mode, the reset value is 1.
27-26 CSPLS	Captured Slot Power Limit Scale
25-18 CSPLV	Captured Slot Power Limit Value For RC mode, the reset value for this field is 00h; for EP mode, the value is determined by received Set_Slot_Power_Limit messages.
17-16 —	Reserved
15 RBER	Role-Based Error Reporting

Table continues on the next page...

Field	Function
14-12 —	Reserved. System software must ignore the value read from these bits. System software is permitted to write any value to these bits.
11-9 EP_L1_LAT	Endpoint L1 Acceptable Latency
8-6 EP_L0s_LAT	Endpoint L0s Acceptable Latency
5 ET	Extended Tag Field Supported
4-3 PHAN_FCT	Phantom Functions Supported
2-0 MAX_PL_SIZE_SUP	Max_Payload_Size Supported Maximum payload size supported. 001 = 256-bytes

25.4.2.53 PCI Express Device Control Register (Device_Control_Register)

25.4.2.53.1 Offset

Register	Offset
Device_Control_Register	78h

25.4.2.53.2 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	IFLR	MAX_READ_SIZE			ENS	APE	PFE	ETE	MAX_PAYLOAD_SIZE			RO	URR	FER	NFER	CER
Reset	0	0	1	0	1	0	0	0	0	0	0	1	0	0	0	0

25.4.2.53.3 Fields

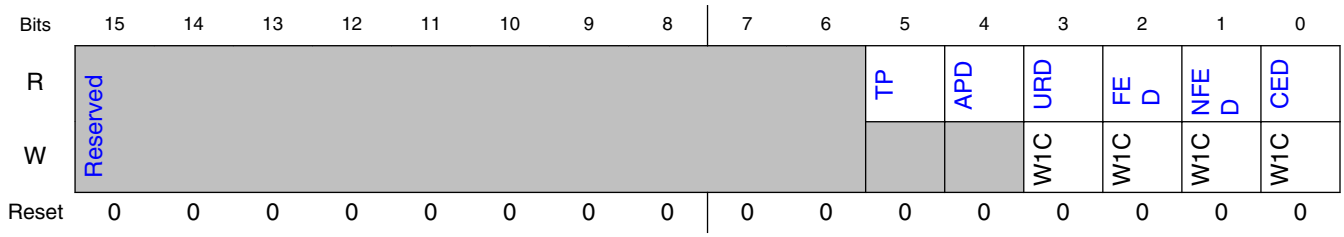
Field	Function
15 IFLR	Initiate Function Level Reset
14-12 MAX_READ_SIZE	Maximum_Read_Request_Size
11 ENS	Enable No Snoop
10 APE	AUX Power PM Enable
9 PFE	Phantom Functions Enable
8 ETE	Extended Tag Field Enable
7-5 MAX_PAYLOAD_SIZE	Max_Payload_Size Maximum payload size
4 RO	Enable Relaxed Ordering
3 URR	Unsupported Request Reporting Enable
2 FER	Fatal Error Reporting Enable
1 NFER	Non-Fatal Error Reporting Enable
0 CER	Correctable Error Reporting Enable

25.4.2.54 PCI Express Device Status Register (Device_Status_Register)

25.4.2.54.1 Offset

Register	Offset
Device_Status_Register	7Ah

25.4.2.54.2 Diagram



25.4.2.54.3 Fields

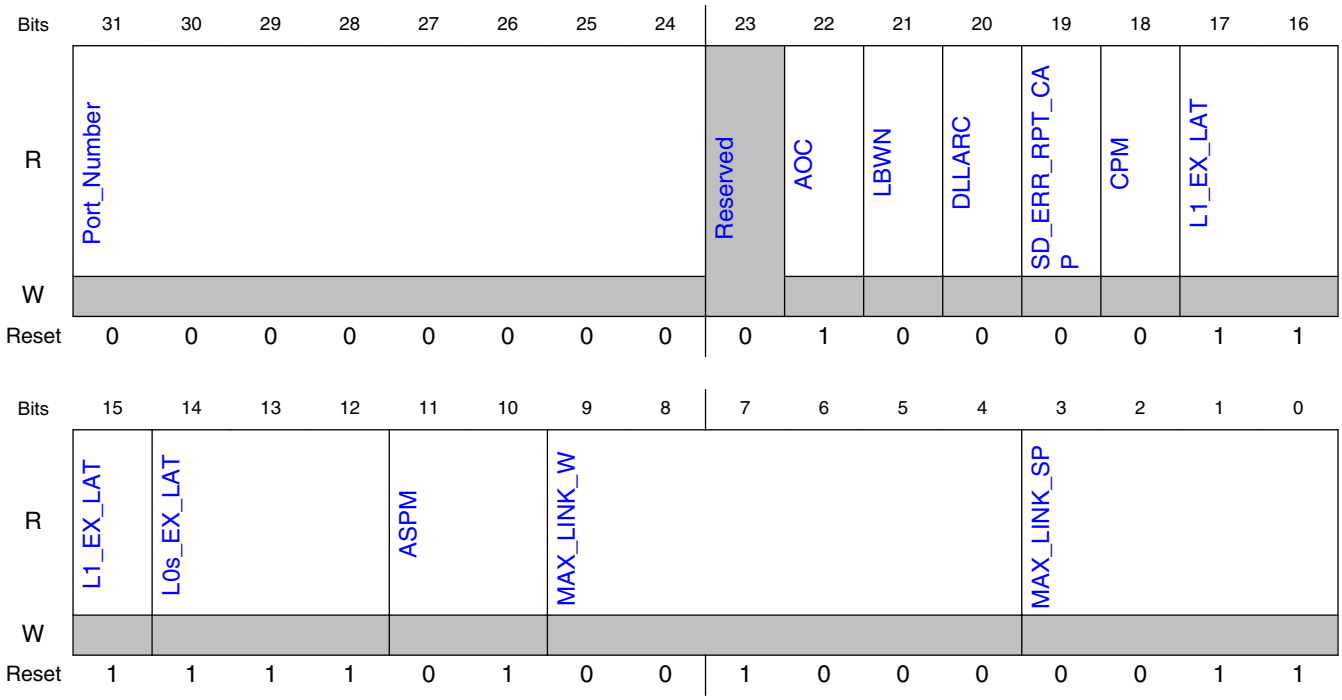
Field	Function
15-6 —	Reserved
5 TP	Transactions Pending
4 APD	AUX Power Detected
3 URD	Unsupported Request Detected
2 FED	Fatal Error Detected
1 NFED	Non-Fatal Error Detected
0 CED	Correctable Error Detected

25.4.2.55 PCI Express Link Capabilities Register (Link_Capabilities_ Register)

25.4.2.55.1 Offset

Register	Offset
Link_Capabilities_Register	7Ch

25.4.2.55.2 Diagram



25.4.2.55.3 Fields

Field	Function
31-24 Port_Number	Port Number This field indicates the PCI Express Port number for the given PCI Express Link
23 —	Reserved
22 AOC	ASPM Optionality Compliance Software is permitted to use the value of this bit to help determine whether to enable ASPM or whether to run ASPM compliance tests.
21 LBWN	Link Bandwidth Notification Capability In EP-mode, this bit is hardwired to 0. In RC-mode it is hardwired to 1.
20 DLLARC	Data Link Layer Active Reporting Capable Set to 1 when in RC. Set to 0 when in EP.
19 SD_ERR_RPT_CAP	Surprise Down Error Reporting Capable
18 CPM	Clock Power Management
17-15 L1_EX_LAT	L1 Exit Latency

Table continues on the next page...

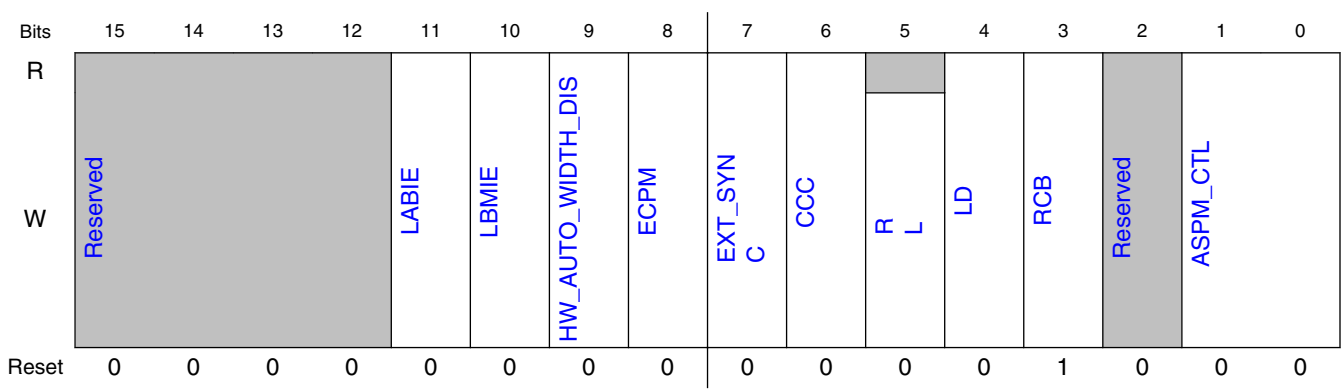
Field	Function
14-12 L0s_EX_LAT	L0s Exit Latency
11-10 ASPM	Active State Power Management (ASPM) Support
9-4 MAX_LINK_W	Maximum Link Width
3-0 MAX_LINK_SP	Maximum Link Speed 0001b - 2.5 GT/s link 0010b - 5.0 GT/s 0011b - 8.0 GT/s

25.4.2.56 PCI Express Link Control Register (Link_Control_Register)

25.4.2.56.1 Offset

Register	Offset
Link_Control_Register	80h

25.4.2.56.2 Diagram



25.4.2.56.3 Fields

Field	Function
15-12 —	Reserved

Table continues on the next page...

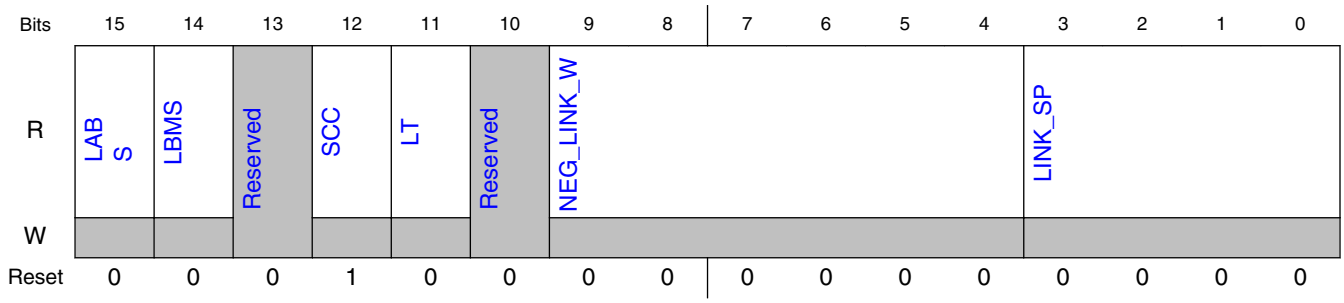
Field	Function
11 LABIE	Link Autonomous Bandwidth Interrupt Enable
10 LBMIE	Link Bandwidth Management Interrupt Enable
9 HW_AUTO_WIDTH_DIS	Hardware Autonomous Width Disable
8 ECPM	Enable Clock Power Management
7 EXT_SYNC	Extended Synch
6 CCC	Common Clock Configuration
5 RL	Retrain Link (Reserved for EP devices). In RC mode, setting this bit initiates link retraining by directing the Physical Layer LTSSM to the Recovery state; reads of this bit always return 0.
4 LD	Link Disable (Reserved for EP devices)
3 RCB	Read Completion Boundary (RCB)
2 —	Reserved
1-0 ASPM_CTL	Active State Power Management (ASPM) Control

25.4.2.57 PCI Express Link Status Register (Link_Status_Register)

25.4.2.57.1 Offset

Register	Offset
Link_Status_Register	82h

25.4.2.57.2 Diagram



25.4.2.57.3 Fields

Field	Function
15 LABS	Link Autonomous Bandwidth Status **NOTE:** This bit is write-1-clear in RC mode; it is read-only in EP mode
14 LBMS	Link Bandwidth Management Status **NOTE:** This bit is write-1-clear in RC mode; it is read-only in EP mode.
13 —	Reserved
12 SCC	Slot Clock Configuration
11 LT	Link Training
10 —	Reserved.
9-4 NEG_LINK_W	Negotiated link width All other encodings are reserved. The value in this field is undefined when the link is not up. 000001b - x1 000010b - x2
3-0 LINK_SP	Current Link Speed 0001b - 2.5 GT/s 0010b - 5.0 GT/s 0011b - 8.0 GT/s

25.4.2.58 PCI Express Slot Capabilities Register (Slot_Capabilities_ Register)

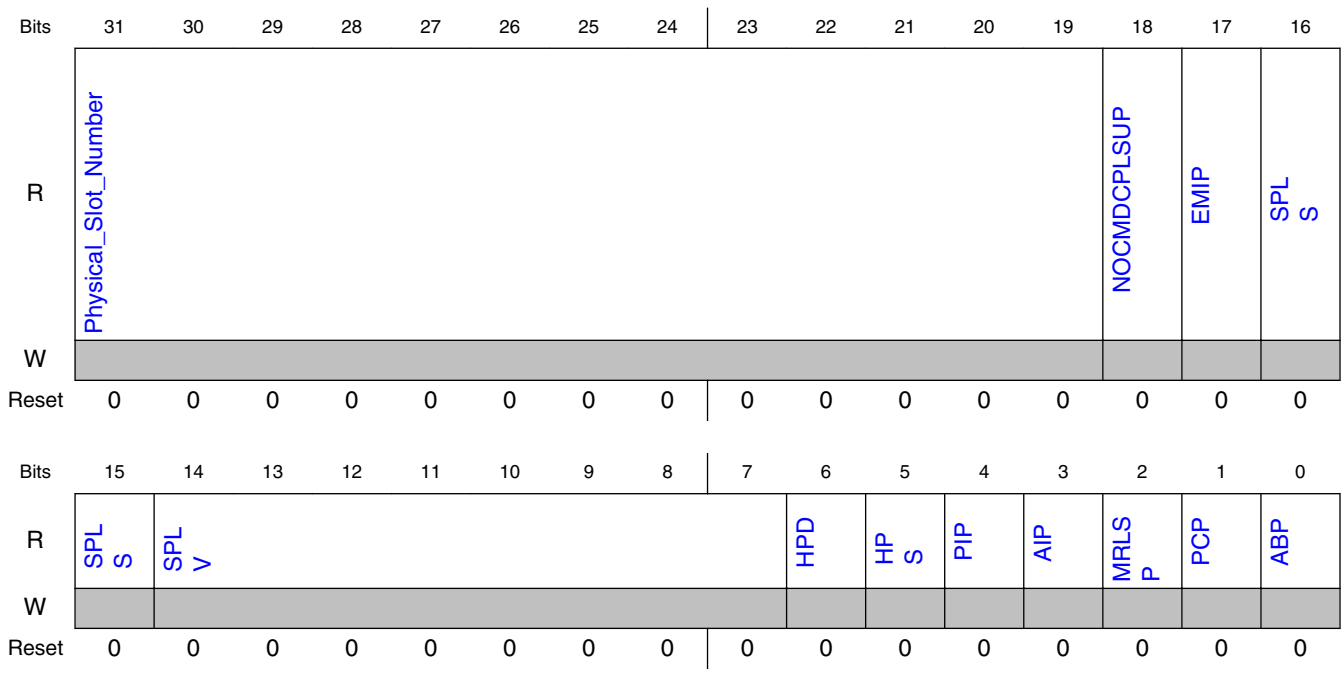
25.4.2.58.1 Offset

Register	Offset
Slot_Capabilities_Register	84h

25.4.2.58.2 Function

This register is supported only for RC mode.

25.4.2.58.3 Diagram



25.4.2.58.4 Fields

Field	Function
31-19 Physical_Slot_Number	Physical Slot Number This field indicates the physical slot number attached to this Port. This field must be hardware initialized to a value that assigns a slot number that is globally unique within the chassis. This field must be initialized to 0 for Ports connected to devices that are either integrated on the system board or integrated within the same silicon as the Switch device or Root Port.
18 NOCMDCPLSUP	No Command Completed Support

Table continues on the next page...

Field	Function
17 EMIP	Electromechanical Interlock Present
16-15 SPLS	Slot Power Limit Scale
14-7 SPLV	Slot Power Limit Value
6 HPD	Hot-Plug Capable **Note:** This chip does not support hot-plug capabilities.
5 HPS	Hot-Plug Surprise **Note:** This chip does not support hot-plug capabilities.
4 PIP	Power Indicator Present **Note:** This chip does not support hot-plug capabilities.
3 AIP	Attention Indicator Present **Note:** This chip does not support hot-plug capabilities.
2 MRLSP	MRL Sensor Present **Note:** This chip does not support hot-plug capabilities.
1 PCP	Power Controller Present **Note:** This chip does not support hot-plug capabilities.
0 ABP	Attention Button Present **Note:** This chip does not support hot-plug capabilities.

25.4.2.59 PCI Express Slot Control Register (Slot_Control_Register)

25.4.2.59.1 Offset

Register	Offset
Slot_Control_Register	88h

25.4.2.59.2 Function

This register is supported only for RC mode.

25.4.2.59.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved			DLLSTCHGEN	EMICTL	PCC	PIC		AIC		HPIE	CCIE	PDCE	MRLSCE	PFDE	ABPE
W																
Reset	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0

25.4.2.59.4 Fields

Field	Function
15-13 —	Reserved
12 DLLSTCHGEN	Data Link Layer State Changed Enable **Note:** This chip does not support hot-plug capabilities.
11 EMICTL	Electromechanical Interlock Control **Note:** This chip does not support hot-plug capabilities.
10 PCC	Power Controller Control **Note:** This chip does not support hot-plug capabilities.
9-8 PIC	Power Indicator Control **Note:** This chip does not support hot-plug capabilities.
7-6 AIC	Attention Indicator Control **Note:** This chip does not support hot-plug capabilities.
5 HPIE	Hot-Plug Interrupt Enable **Note:** This chip does not support hot-plug capabilities.
4 CCIE	Command Completed Interrupt Enable **Note:** This chip does not support hot-plug capabilities.
3 PDCE	Presence Detect Changed Enable **Note:** This chip does not support hot-plug capabilities.
2 MRLSCE	MRL Sensor Changed Enable **Note:** This chip does not support hot-plug capabilities.
1 PFDE	Power Fault Detected Enable **Note:** This chip does not support hot-plug capabilities.
0 ABPE	Attention Button Pressed Enable **Note:** This chip does not support hot-plug capabilities.

25.4.2.60 PCI Express Slot Status Register (Slot_Status_Register)

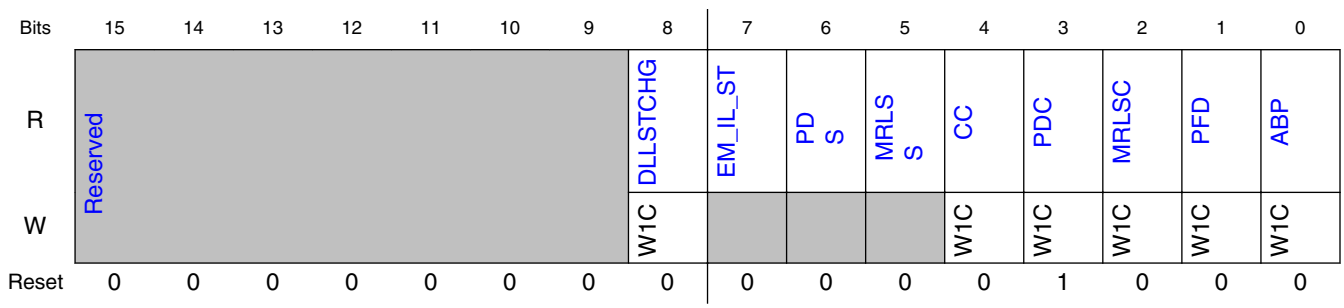
25.4.2.60.1 Offset

Register	Offset
Slot_Status_Register	8Ah

25.4.2.60.2 Function

This register is supported only for RC mode.

25.4.2.60.3 Diagram



25.4.2.60.4 Fields

Field	Function
15-9 —	Reserved
8 DLLSTCHG	Data Link Layer State Changed
7 EM_IL_ST	Electromechanical Interlock Status
6 PDS	Presence Detect State This bit indicates the presence of an adapter in the slot, reflected by the logical OR of the Physical Layer in-band presence detect mechanism and, if present, any out-of-band presence detect mechanism defined for the slotâs corresponding form factor. Note that the in-band presence detect mechanism requires that power be applied to an adapter for its presence to be detected. Consequently, form factors that require a power controller for hot-plug must implement a physical pin presence detect mechanism. This register must be implemented on all Downstream Ports that implement slots. For Downstream Ports not connected to slots (where the Slot Implemented bit of the PCI Express Capabilities register is 0b), this bit must return 1b. Defined encodings are:

Table continues on the next page...

Field	Function
	0b - Slot Empty 1b - Card Present in slot
5 MRLSS	MRL Sensor State 0b - MRL closed 1b - MRL open
4 CC	Command Completed
3 PDC	Presence Detect Changed
2 MRLSC	MRL Sensor Changed
1 PFD	Power Fault Detected
0 ABP	Attention Button Pressed

25.4.2.61 PCI Express Root Control Register (Root_Control_Register)

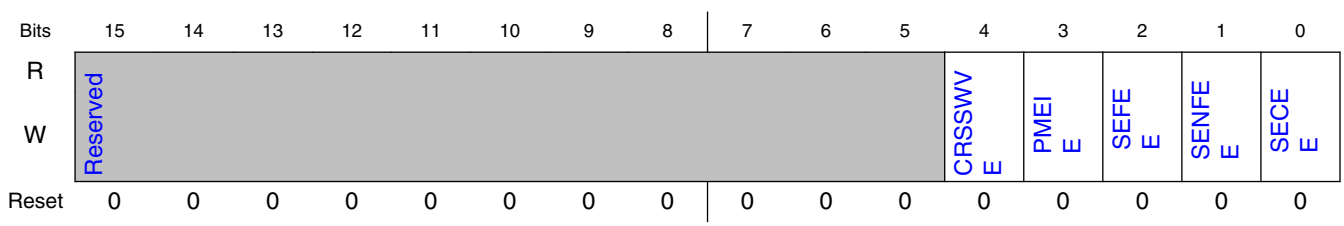
25.4.2.61.1 Offset

Register	Offset
Root_Control_Register	8Ch

25.4.2.61.2 Function

This register is supported only for RC mode.

25.4.2.61.3 Diagram



25.4.2.61.4 Fields

Field	Function
15-5 —	Reserved
4 CRSSWVE	CRS Software Visibility Enable
3 PMEIE	PME Interrupt Enable
2 SEFEE	System Error on Fatal Error Enable
1 SENFEE	System Error on Non-Fatal Error Enable
0 SECEE	System Error on Correctable Error Enable

25.4.2.62 PCI Express Root Capabilities Register (Root_Capabilities_ Register)

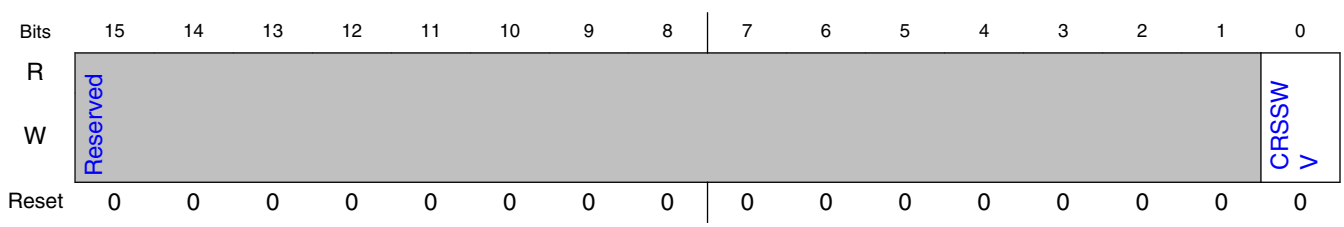
25.4.2.62.1 Offset

Register	Offset
Root_Capabilities_Register	8Eh

25.4.2.62.2 Function

This register is supported only for RC mode.

25.4.2.62.3 Diagram



25.4.2.62.4 Fields

Field	Function
15-1 —	Reserved
0 CRSSWV	CRS Software Visibility

25.4.2.63 PCI Express Root Status Register (Root_Status_Register)

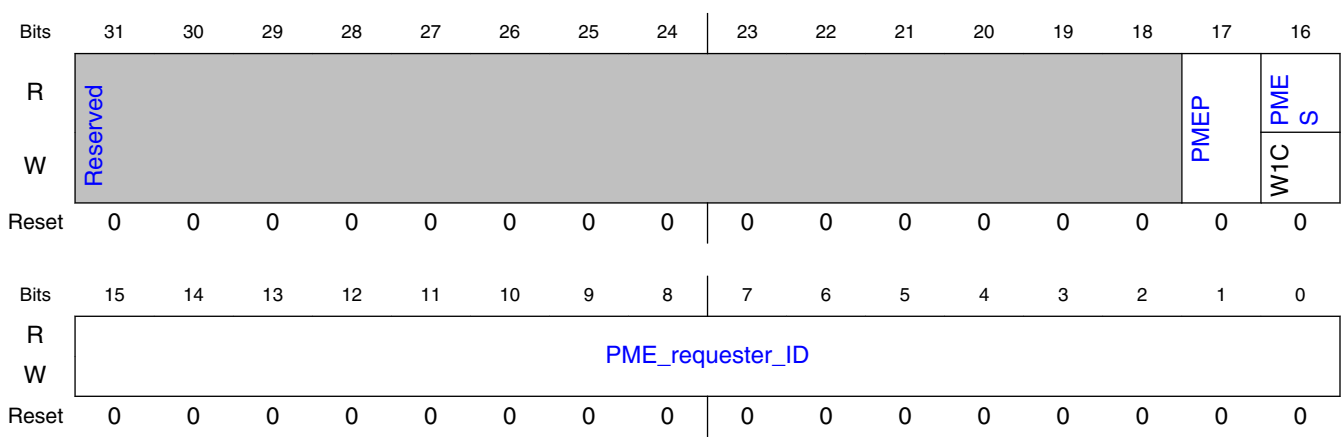
25.4.2.63.1 Offset

Register	Offset
Root_Status_Register	90h

25.4.2.63.2 Function

This register is supported only for RC mode.

25.4.2.63.3 Diagram



25.4.2.63.4 Fields

Field	Function
31-18	Reserved

Table continues on the next page...

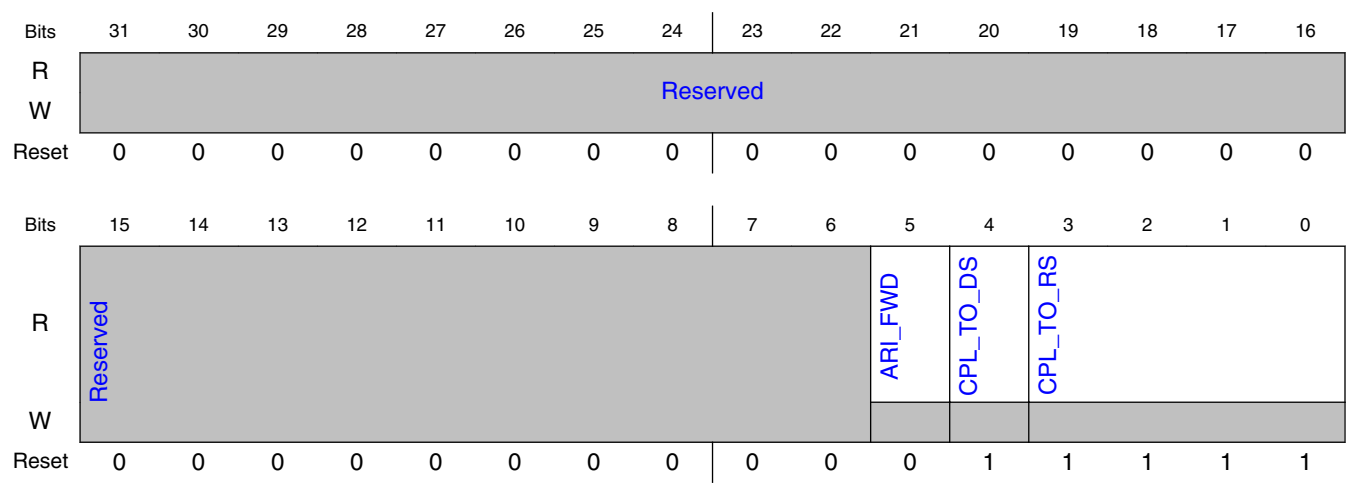
Field	Function
—	
17 PMEP	PME Pending
16 PMES	PME Status
15-0 PME_requester_ID	PME Requester ID

25.4.2.64 PCI Express Device Capabilities 2 Register (Device_Capabilities_2_Register)

25.4.2.64.1 Offset

Register	Offset
Device_Capabilities_2_Register	94h

25.4.2.64.2 Diagram



25.4.2.64.3 Fields

Field	Function
31-6	Reserved

Table continues on the next page...

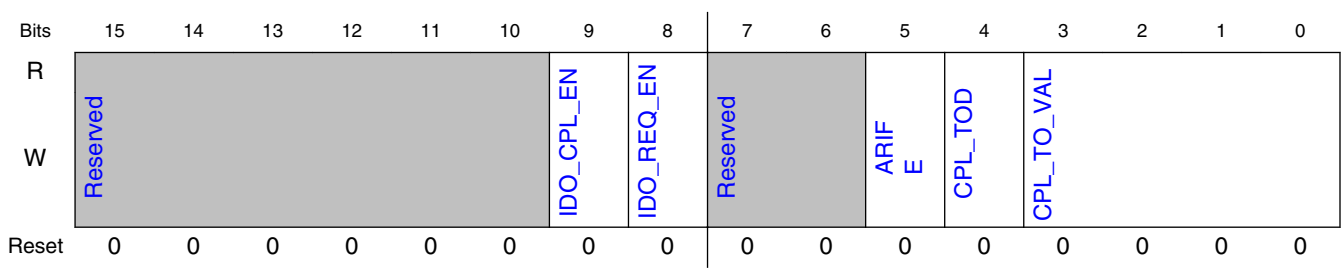
Field	Function
—	
5 ARI_FWD	ARI Forwarding Supported
4 CPL_TO_DS	Completion Timeout Disable Supported
3-0 CPL_TO_RS	Completion Timeout Ranges Supported

25.4.2.65 PCI Express Device Control 2 Register (Device_Control_2_ Register)

25.4.2.65.1 Offset

Register	Offset
Device_Control_2_Register	98h

25.4.2.65.2 Diagram



25.4.2.65.3 Fields

Field	Function
15-10 —	Reserved
9 IDO_CPL_EN	IDO Completion Enable
8 IDO_REQ_EN	IDO Request Enable

Table continues on the next page...

Field	Function
7-6 —	Reserved
5 ARIFE	ARI Forwarding Enable Must be set when RC is communicating with ARI devices. When set will allow RC to issue configuration type 0 cycles with device number != 0.
4 CPL_TOD	Completion Timeout Disable
3-0 CPL_TO_VAL	Completion Timeout Value

25.4.2.66 PCI Express Link Capabilities 2 Register (Link_Capabilities_2_Register)

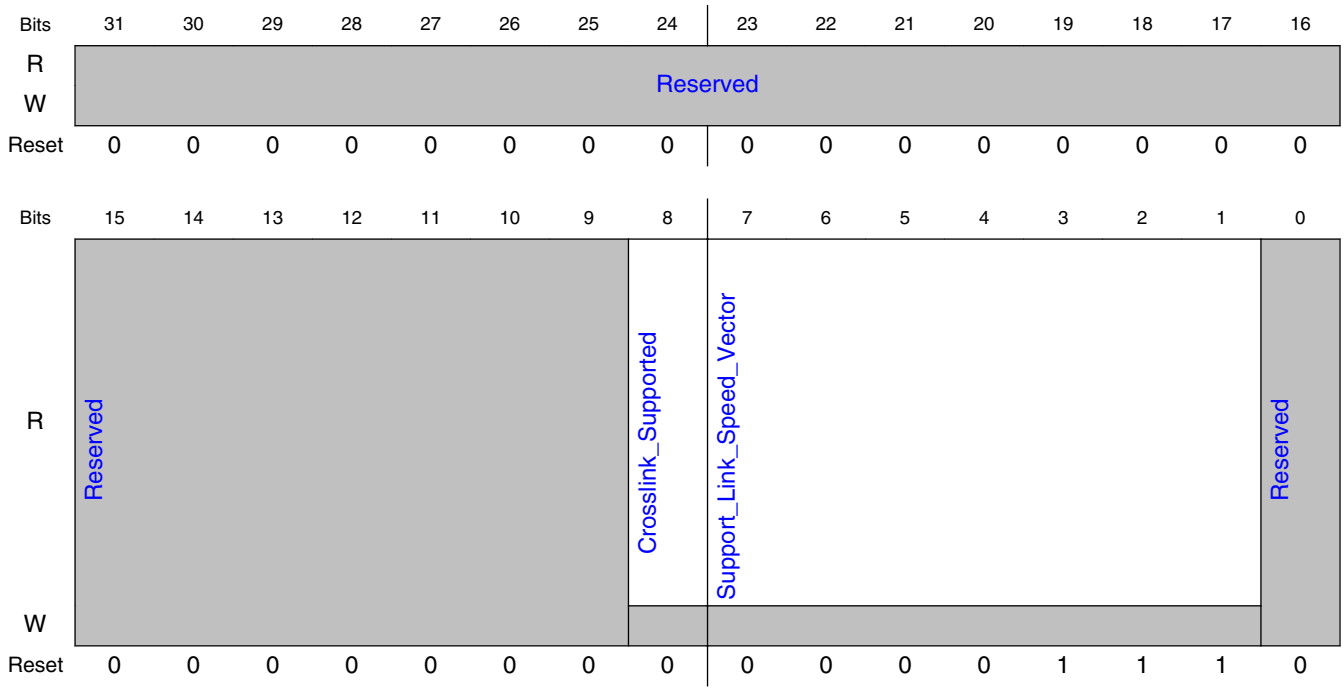
25.4.2.66.1 Offset

Register	Offset
Link_Capabilities_2_Register	9Ch

25.4.2.66.2 Function

The PCI Express link capability 2 register is shown below.

25.4.2.66.3 Diagram



25.4.2.66.4 Fields

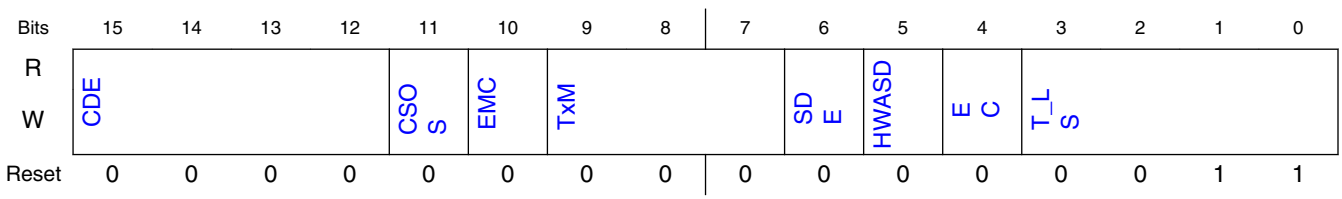
Field	Function
31-9 —	Reserved
8 Crosslink_Supported	Crosslink Supported
7-1 Support_Link_Speed_Vector	Supported Link Speeds Vector This field indicates the supported Link speed(s) of the associated Port. For each bit, a value of 1 indicates that the corresponding Link speed is supported; otherwise, 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
0 —	Reserved

25.4.2.67 PCI Express Link Control 2 Register (Link_Control_2_Register)

25.4.2.67.1 Offset

Register	Offset
Link_Control_2_Register	A0h

25.4.2.67.2 Diagram



25.4.2.67.3 Fields

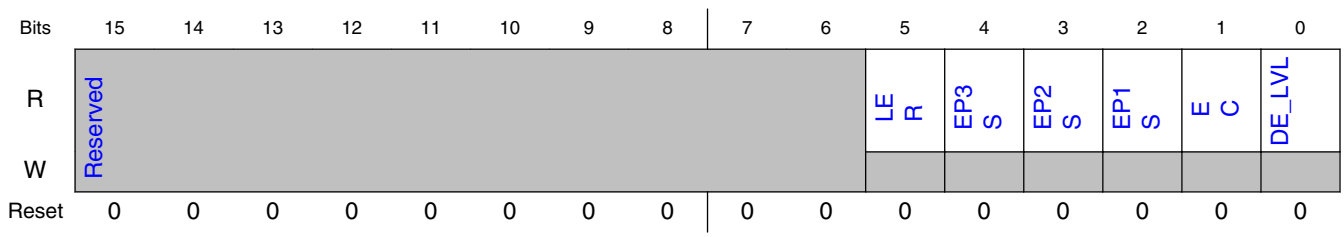
Field	Function
15-12 CDE	Compliance Preset/De-emphasis
11 CSOS	Compliance SOS
10 EMC	Enter Modified Compliance
9-7 TxM	Transmit Margin
6 SDE	Selectable De-emphasis
5 HWASD	Hardware Autonomous Speed Disable
4 EC	Enter Compliance
3-0 T_LS	Target Link Speed

25.4.2.68 PCI Express Link Status 2 Register (Link_Status_2_Register)

25.4.2.68.1 Offset

Register	Offset
Link_Status_2_Register	A2h

25.4.2.68.2 Diagram



25.4.2.68.3 Fields

Field	Function
15-6 —	Reserved
5 LER	Link Equalization Request
4 EP3S	Equalization Phase 3 Successful
3 EP2S	Equalization Phase 2 Successful
2 EP1S	Equalization Phase 1 Successful
1 EC	Equalization Complete
0 DE_LVL	Current De-emphasis Level

25.4.2.69 PCI Express Advanced Error Reporting Capability ID Register (Advanced_Error_Reporting_Capability_ID_Register)

25.4.2.69.1 Offset

Register	Offset
Advanced_Error_Reporting_Capability_ID_Register	100h

25.4.2.69.2 Diagram

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

25.4.2.69.3 Fields

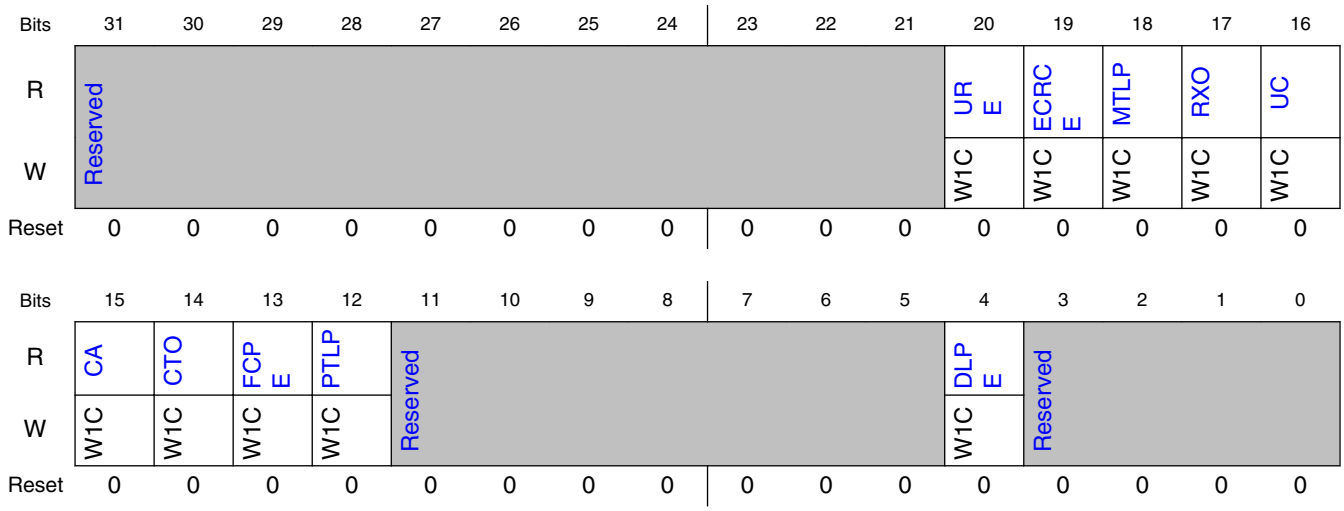
Field	Function
15-0 Capability_ID	PCI Express Extended Capability ID 0x0001 = Advanced error reporting capability

25.4.2.70 PCI Express Uncorrectable Error Status Register (Uncorrectable_Error_Status_Register)

25.4.2.70.1 Offset

Register	Offset
Uncorrectable_Error_Status_Register	104h

25.4.2.70.2 Diagram



25.4.2.70.3 Fields

Field	Function
31-21 —	Reserved
20 URE	Unsupported request error status.
19 ECRCE	ECRC error status.
18 MTLP	Malformed TLP status.
17 RXO	Receiver overflow status.
16 UC	Unexpected completion status.
15 CA	Completer abort status.
14 CTO	Completion timeout status. Note that a completion timeout error is a fatal error. If a completion timeout error is detected, the system has become unstable. Hot reset is recommended to restore stability of the system.
13 FCPE	Flow control protocol error status.
12 PTLP	Poisoned TLP status.
11-5 —	Reserved

Table continues on the next page...

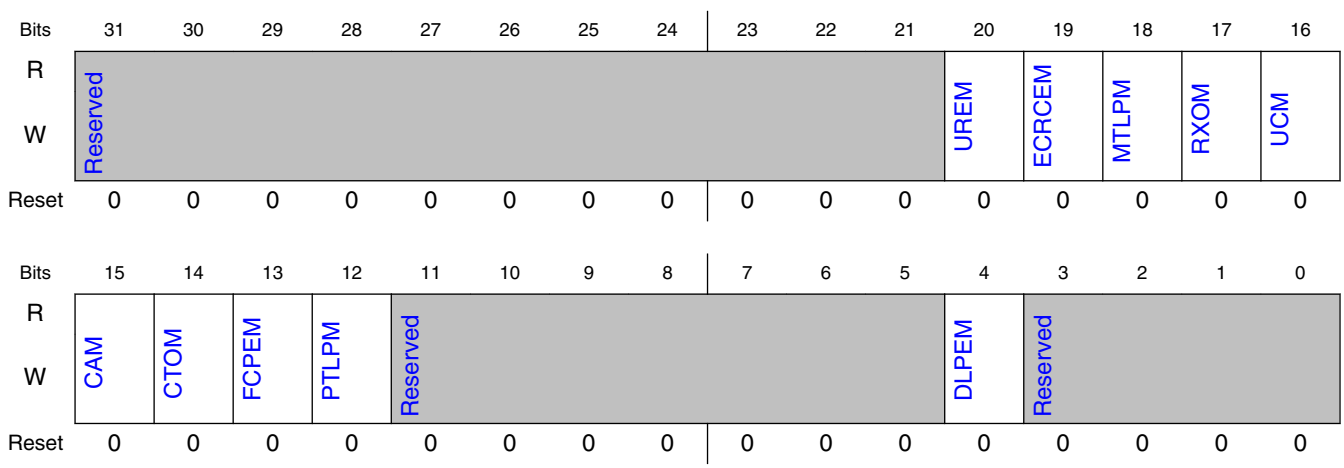
Field	Function
4 DLPE	Data link protocol error status.
3-0 —	Reserved

25.4.2.71 PCI Express Uncorrectable Error Mask Register (Uncorrectable_Error_Mask_Register)

25.4.2.71.1 Offset

Register	Offset
Uncorrectable_Error_Mask_Register	108h

25.4.2.71.2 Diagram



25.4.2.71.3 Fields

Field	Function
31-21 —	Reserved
20 UREM	Unsupported request error mask.

Table continues on the next page...

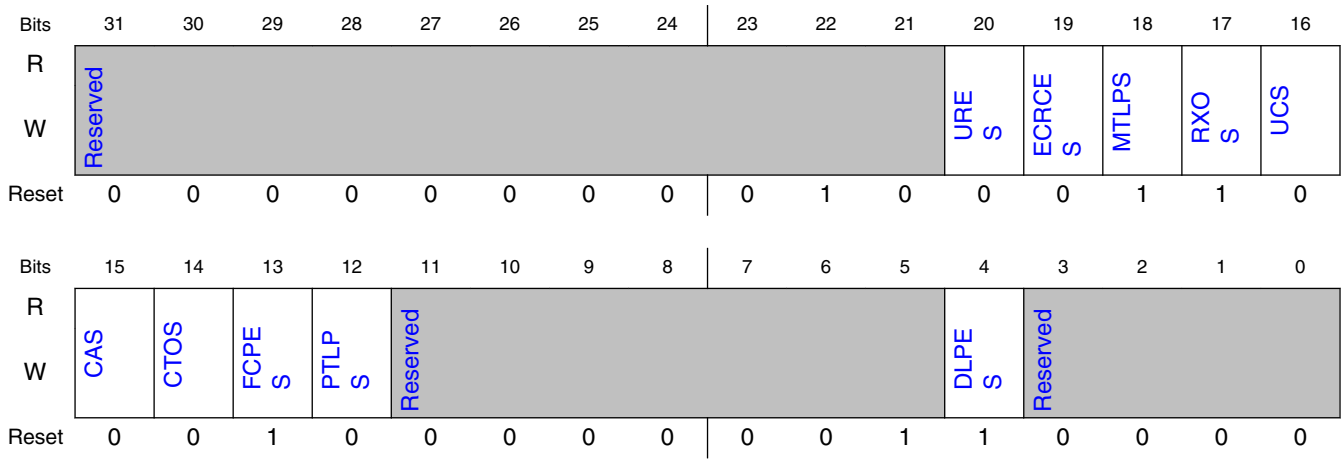
Field	Function
19 ECRCEM	ECRC error mask.
18 MTLPM	Malformed TLP mask.
17 RXOM	Receiver overflow mask.
16 UCM	Unexpected completion mask.
15 CAM	Completer abort mask.
14 CTOM	Completion timeout mask.
13 FCPEM	Flow control protocol error mask.
12 PTLPM	Poisoned TLP mask.
11-5 —	Reserved
4 DLPEM	Data link protocol error mask.
3-0 —	Reserved

25.4.2.72 PCI Express Uncorrectable Error Severity Register (Uncorrectable_Error_Severity_Register)

25.4.2.72.1 Offset

Register	Offset
Uncorrectable_Error_Severity_Register	10Ch

25.4.2.72.2 Diagram



25.4.2.72.3 Fields

Field	Function
31-21 —	Reserved
20 URES	Unsupported request error severity.
19 ECRCES	ECRC error severity.
18 MTLPS	Malformed TLP severity.
17 RXOS	Receiver overflow severity.
16 UCS	Unexpected completion severity.
15 CAS	Completer abort severity.
14 CTOS	Completion timeout severity.
13 FCPES	Flow control protocol error severity.
12 PTLPS	Poisoned TLP severity.
11-5 —	Reserved
4	Data link protocol error severity.

Table continues on the next page...

Field	Function
DLPES	
3-0 —	Reserved

25.4.2.73 PCI Express Correctable Error Status Register (Correctable_Error_Status_Register)

25.4.2.73.1 Offset

Register	Offset
Correctable_Error_Status_Register	110h

25.4.2.73.2 Diagram

Bits	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

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved		ADVNFE	RTTO	Reserved			RNR	BDLLP	BTLP	Reserved					RXE
W			W1C	W1C				W1C	W1C	W1C						W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

25.4.2.73.3 Fields

Field	Function
31-14 —	Reserved
13 ADVNFE	Advisory Non-Fatal Error Status indicates the occurrence of the advisory error, and the Advisory Non-Fatal Error Mask bit in the PCI Express Correctable Error Mask Register (Correctable_Error_Mask_Register) is checked to determine whether to proceed further with logging and signaling
12	Replay timer timeout status

Table continues on the next page...

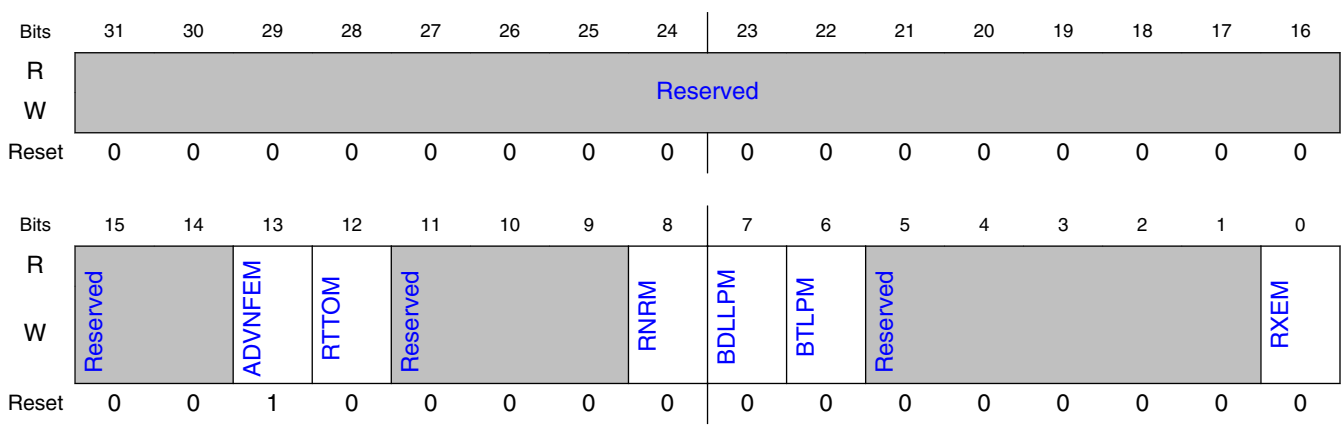
Field	Function
RTTO	
11-9 —	Reserved
8 RNR	REPLAY_NUM Rollover status
7 BDLLP	Bad DLLP status
6 BTLP	Bad TLP status
5-1 —	Reserved
0 RXE	Receiver error status

25.4.2.74 PCI Express Correctable Error Mask Register (Correctable_Error_Mask_Register)

25.4.2.74.1 Offset

Register	Offset
Correctable_Error_Mask_Register	114h

25.4.2.74.2 Diagram



25.4.2.74.3 Fields

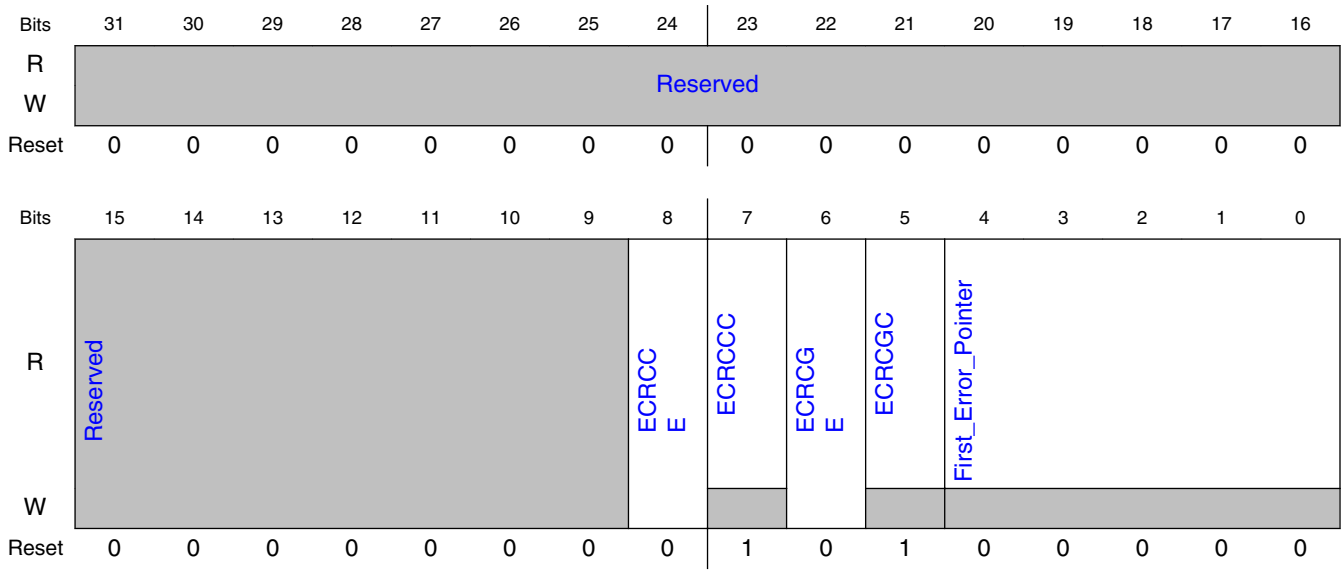
Field	Function
31-14 —	Reserved
13 ADVNFEM	Advisory non-fatal error mask. This bit is Set by default to enable compatibility with software that does not comprehend Role-Based Error Reporting
12 RTTOM	Replay timer timeout mask
11-9 —	Reserved
8 RNRM	REPLAY_NUM Rollover mask
7 BDLLPM	Bad DLLP mask
6 BTLPM	Bad TLP mask
5-1 —	Reserved
0 RXEM	Receiver error mask

25.4.2.75 PCI Express Advanced Error Capabilities and Control Register (Advanced_Error_Capabilities_and_Control_Register)

25.4.2.75.1 Offset

Register	Offset
Advanced_Error_Capabilities_and_Control_Register	118h

25.4.2.75.2 Diagram



25.4.2.75.3 Fields

Field	Function
31-9 —	Reserved
8 ECRCCE	ECRC checking enable.
7 ECRCCC	ECRC checking capable.
6 ECRCGE	ECRC generation enable.
5 ECRCGC	ECRC generation capable.
4-0 First_Error_Pointer	The First Error Pointer is a read-only register that identifies the bit position of the first error reported in the Uncorrectable Error Status register.

25.4.2.76 PCI Express Header Log Register 1 (Header_Log_Register_DWORD1)

25.4.2.76.1 Offset

Register	Offset
Header_Log_Register_ DWORD1	11Ch

25.4.2.76.2 Function

The first DWORD of the PCI Express header log register is shown in the figure below.

25.4.2.76.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Byte_0								Byte_1							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Byte_2								Byte_3							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

25.4.2.76.4 Fields

Field	Function
31-24 Byte_0	Byte n of the TLP header associated with the error.
23-16 Byte_1	Byte n of the TLP header associated with the error.
15-8 Byte_2	Byte n of the TLP header associated with the error.
7-0 Byte_3	Byte n of the TLP header associated with the error.

25.4.2.77 PCI Express Header Log Register 2 (Header_Log_Register_ DWORD2)

25.4.2.77.1 Offset

Register	Offset
Header_Log_Register_ DWORD2	120h

25.4.2.77.2 Function

The second DWORD of the PCI Express header log register is shown in the figure below.

25.4.2.77.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Byte_4								Byte_5							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Byte_6								Byte_7							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

25.4.2.77.4 Fields

Field	Function
31-24 Byte_4	Byte n of the TLP header associated with the error.
23-16 Byte_5	Byte n of the TLP header associated with the error.
15-8 Byte_6	Byte n of the TLP header associated with the error.
7-0 Byte_7	Byte n of the TLP header associated with the error.

25.4.2.78 PCI Express Header Log Register 3 (Header_Log_Register_ DWORD3)

25.4.2.78.1 Offset

Register	Offset
Header_Log_Register_ DWORD3	124h

25.4.2.78.2 Function

The third DWORD of the PCI Express header log register is shown in the figure below.

25.4.2.78.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Byte_8								Byte_9							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Byte_A								Byte_B							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

25.4.2.78.4 Fields

Field	Function
31-24 Byte_8	Byte n of the TLP header associated with the error.
23-16 Byte_9	Byte n of the TLP header associated with the error.
15-8 Byte_A	Byte n of the TLP header associated with the error.
7-0 Byte_B	Byte n of the TLP header associated with the error.

25.4.2.79 PCI Express Header Log Register 4 (Header_Log_Register_ DWORD4)

25.4.2.79.1 Offset

Register	Offset
Header_Log_Register_DWORD4	128h

25.4.2.79.2 Function

The fourth DWORD of the PCI Express header log register is shown in the figure below.

25.4.2.79.3 Diagram

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

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Byte_E								Byte_F							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

25.4.2.79.4 Fields

Field	Function
31-24 Byte_C	Byte n of the TLP header associated with the error.
23-16 Byte_D	Byte n of the TLP header associated with the error.
15-8 Byte_E	Byte n of the TLP header associated with the error.
7-0 Byte_F	Byte n of the TLP header associated with the error.

25.4.2.80 PCI Express Root Error Command Register (Root_Error_Command_Register)

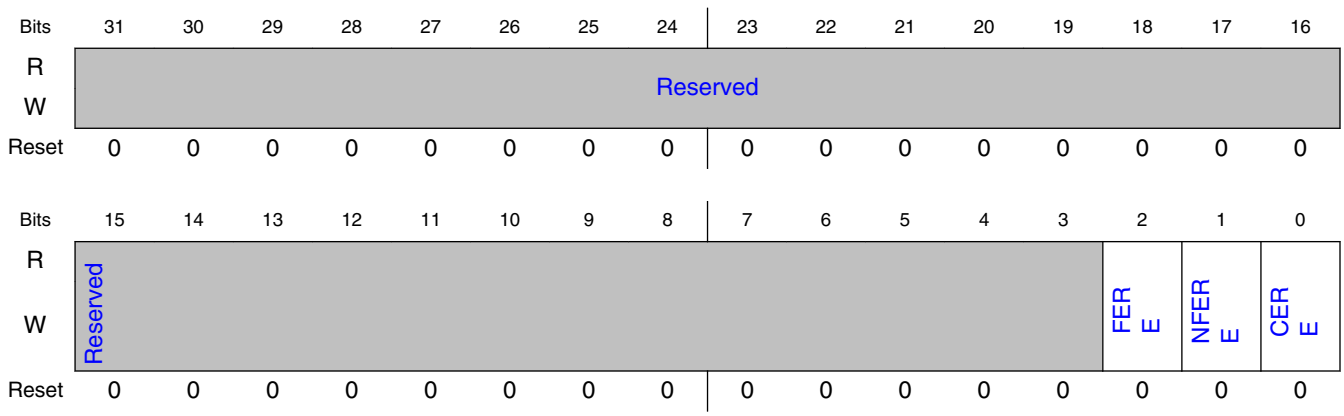
25.4.2.80.1 Offset

Register	Offset
Root_Error_Command_Register	12Ch

25.4.2.80.2 Function

This register is supported only for RC mode.

25.4.2.80.3 Diagram



25.4.2.80.4 Fields

Field	Function
31-3 —	Reserved
2 FERE	Fatal error reporting enable.
1 NFERE	Non-fatal error reporting enable
0 CERE	Correctable error reporting enable

25.4.2.81 PCI Express Root Error Status Register (Root_Error_Status_Register)

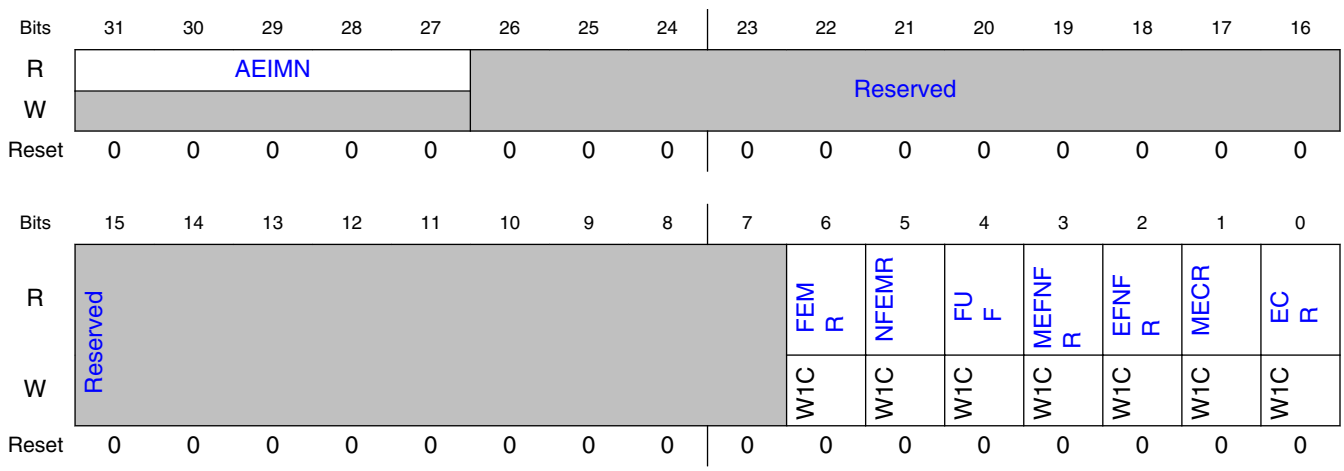
25.4.2.81.1 Offset

Register	Offset
Root_Error_Status_Register	130h

25.4.2.81.2 Function

This register is supported only for RC mode.

25.4.2.81.3 Diagram



25.4.2.81.4 Fields

Field	Function
31-27 AEIMN	Advanced error interrupt message number.
26-7 —	Reserved
6 FEMR	Fatal error messages received.
5 NFEMR	Non-fatal error messages received.

Table continues on the next page...

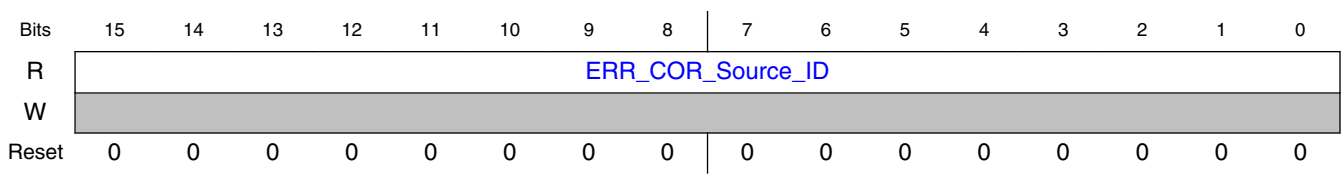
Field	Function
4 FUF	First uncorrectable fatal.
3 MEFNFR	Multiple ERR_FATAL/NONFATAL received.
2 EFNFR	ERR_FATAL/NONFATAL received.
1 MECR	Multiple ERR_COR received.
0 ECR	ERR_COR received.

25.4.2.82 PCI Express Correctable Error Source ID Register (Correctable_Error_Source_ID_Register)

25.4.2.82.1 Offset

Register	Offset
Correctable_Error_Source_ID_Register	134h

25.4.2.82.2 Diagram



25.4.2.82.3 Fields

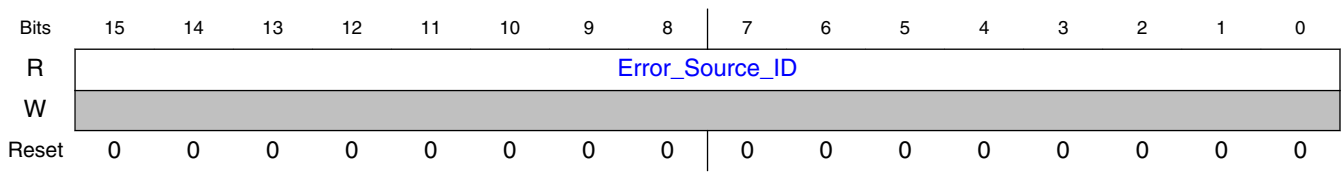
Field	Function
15-0 ERR_COR_Source_ID	Loaded with the requester ID indicated in the received ERR_COR Message when the ERR_COR Received register is not already set. Default value of this field is 0.

25.4.2.83 PCI Express Error Source ID Register (Error_Source_ID_ Register)

25.4.2.83.1 Offset

Register	Offset
Error_Source_ID_Register	136h

25.4.2.83.2 Diagram



25.4.2.83.3 Fields

Field	Function
15-0 Error_Source_ID	ERR_FATAL/NONFATAL source ID

25.4.2.84 Secondary PCI Express Extended Capability Header (SPCIE_CAP_HEADER_REG)

25.4.2.84.1 Offset

Register	Offset
SPCIE_CAP_HEADER_REG	148h

25.4.2.84.2 Diagram

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

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

25.4.2.84.3 Fields

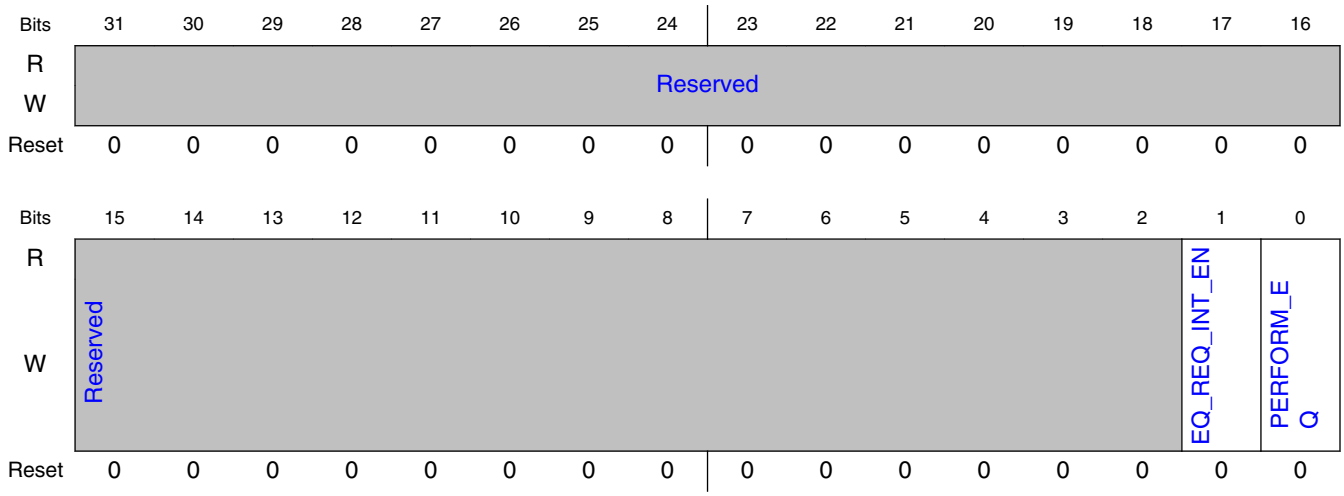
Field	Function
31-20 NEXT_OFFSET	Next Capability Offset This field is a pointer to the next capability structure. **NOTE:** The reset value of this field depends on the instantiation of the module. See PCI Express configuration registers for the specific values.
19-16 CAP_VERSION	Capability Version Indicates the version of the Capability structure present.
15-0 EXTENDED_CAP_ID	PCI Express Extended Capability ID 0019h = Secondary PCI Express extended capability

25.4.2.85 Link Control 3 Register (LINK_CONTROL3_REG)

25.4.2.85.1 Offset

Register	Offset
LINK_CONTROL3_REG	14Ch

25.4.2.85.2 Diagram



25.4.2.85.3 Fields

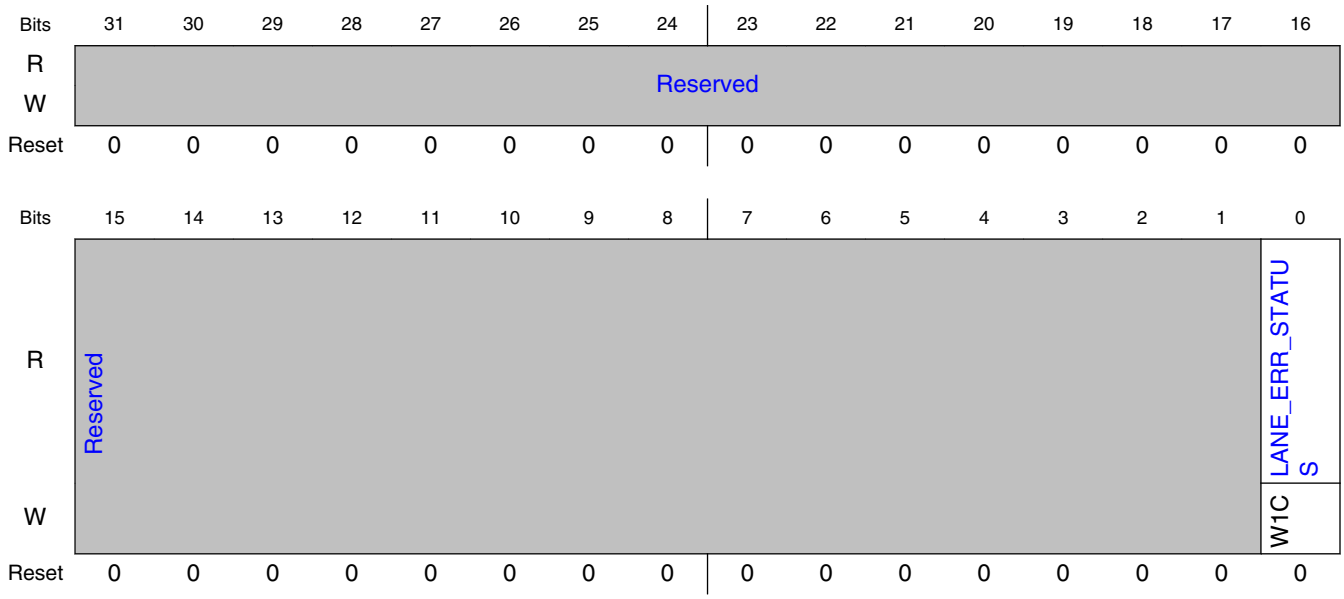
Field	Function
31-2 —	Reserved
1 EQ_REQ_INT_EN	Link Equalization Request Interrupt Enable This bit is RW for Downstream Ports only.
0 PERFORM_EQ	Perform Equalization This bit is RW for Downstream Ports only.

25.4.2.86 Lane Error Status Register (LANE_ERR_STATUS_REG)

25.4.2.86.1 Offset

Register	Offset
LANE_ERR_STATUS_REG	150h

25.4.2.86.2 Diagram



25.4.2.86.3 Fields

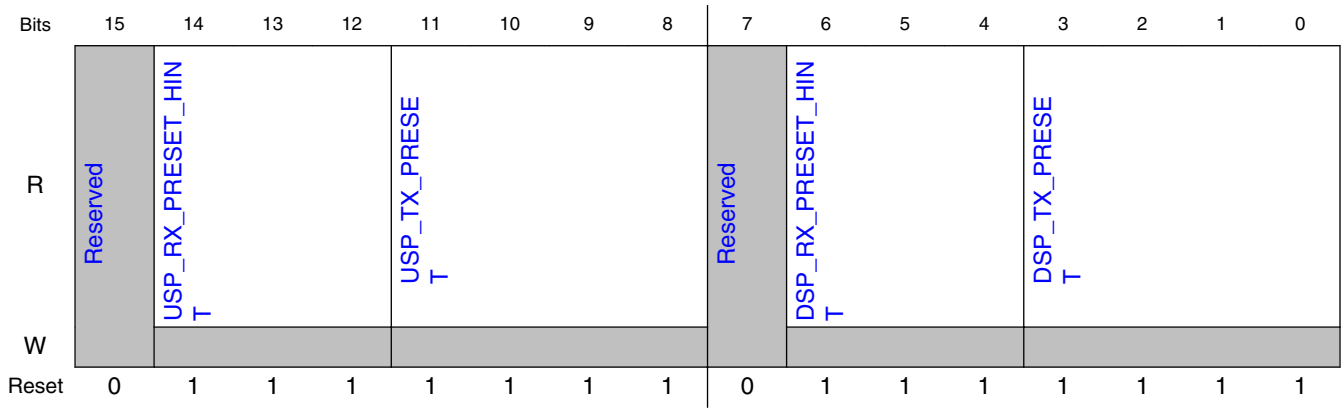
Field	Function
31-1 —	Reserved
0 LANE_ERR_STATUS	Lane Error Status Each bit indicates if the corresponding Lane detected a Lane-based error.

25.4.2.87 Lane Equalization Control Register (LANE0_EQUALIZATION_CONTROL)

25.4.2.87.1 Offset

Register	Offset
LANE0_EQUALIZATION_CONTROL	154h

25.4.2.87.2 Diagram



25.4.2.87.3 Fields

Field	Function
15 —	Reserved
14-12 USP_RX_PRESET_HINT	Upstream Port Receiver Preset Hint
11-8 USP_TX_PRESET	Upstream Port Transmitter Preset
7 —	Reserved
6-4 DSP_RX_PRESET_HINT	Downstream Port Receiver Preset Hint
3-0 DSP_TX_PRESET	Downstream Port Transmitter Preset

25.4.2.88 Symbol Timer Register and Filter Mask 1 Register (SYMBOL_TIMER_FILTER_1_OFF)

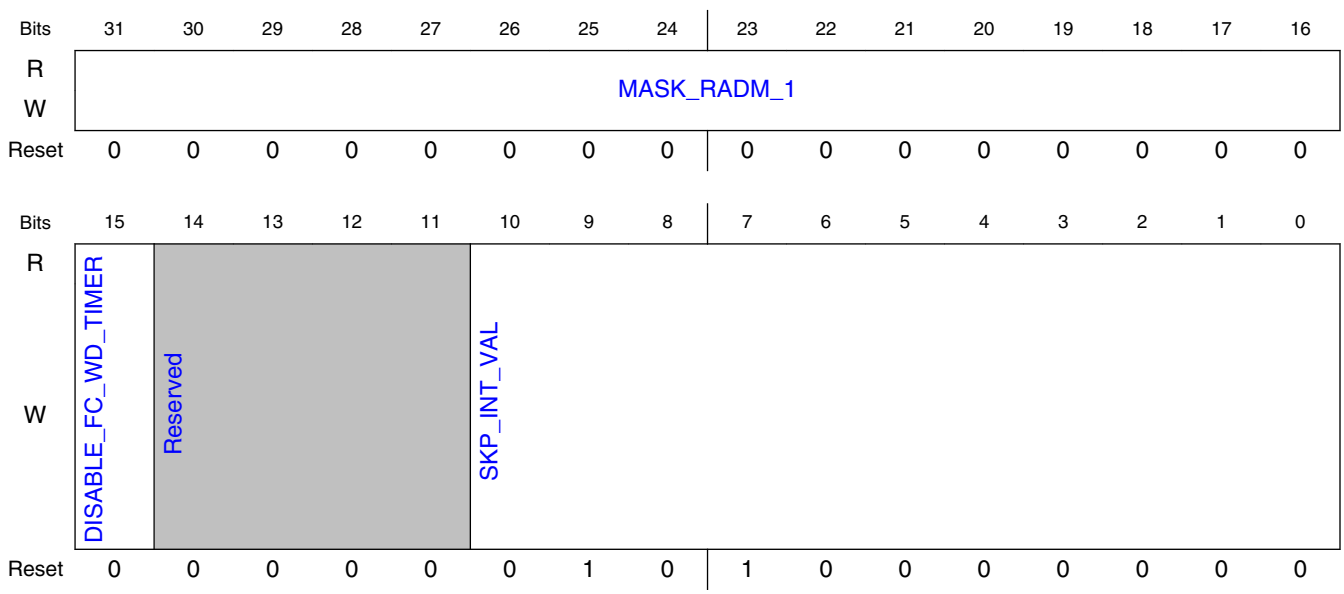
25.4.2.88.1 Offset

Register	Offset
SYMBOL_TIMER_FILTER_1_OFF	71Ch

25.4.2.88.2 Function

The Filter Mask 1 Register modifies the receive filtering and error handling rules. In each case, '0' applies the associated filtering rule and '1' masks the associated filtering rule

25.4.2.88.3 Diagram



25.4.2.88.4 Fields

Field	Function
31-16	Filter Mask 1
MASK_RADM_1	The Filter Mask 1 Register modifies the receive filtering and error handling rules. In each case, '0' applies the associated filtering rule and '1' masks the associated filtering rule. [31]: CX_FLT_MASK_RC_CFG_DISCARD 0: For RC filter to not allow CFG transaction being received 1: For RC filter to allow CFG transaction being received [30]: CX_FLT_MASK_RC_IO_DISCARD 0: For RC filter to not allow IO transaction being received 1: For RC filter to allow IO transaction being received

Table continues on the next page...

Field	Function
	[29]: CX_FLT_MASK_MSG_DROP 0: Drop MSG TLP (except for Vendor MSG). 1: Do not Drop MSG (except for Vendor MSG). Send message TLPs to your application. This bit only controls message TLPs other than Vendor MSGs. Vendor MSGs are controlled by Filter Mask Register 2, bits [1:0]. **NOTE:** The forwarding of message TLPs may be undesirable. If so, write a 0 to this bit to disable forwarding of message TLPs. [28]: CX_FLT_MASK_CPL_ECRC_DISCARD Only used when completion queue is advertized with infinite credits and is in store-and-forward mode. 0: Discard completions with ECRC errors 1: Allow completions with ECRC errors to be passed up Reserved field for SW. [27]: 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 0: Enforce length match for completions; a violation results in cpl_abort, and possibly AER of unexp_cpl_err 1: MASK length match for completions [25]: CX_FLT_MASK_CPL_ATTR_MATCH 0: Enforce attribute match for completions; 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 completions [24]: CX_FLT_MASK_CPL_TC_MATCH 0: Enforce Traffic Class match for completions; 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 completions [23]: CX_FLT_MASK_CPL_FUNC_MATCH 0: Enforce function match for completions; a violation results in cpl_abort, and possibly AER of unexp_cpl_err, cpl_rcvd_ur, cpl_rcvd_ca 1: Mask function match for completions [22]: CX_FLT_MASK_CPL_REQID_MATCH 0: Enforce Req. Id match for completions; 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 completions [21]: CX_FLT_MASK_CPL_TAGERR_MATCH 0: Enforce Tag Error Rules for completions; 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 completions [20]: CX_FLT_MASK_LOCKED_RD_AS_UR 0: Treat locked Read TLPs as UR for EP. Treat locked Read TLPs as supported for RC.

Table continues on the next page...

Field	Function
	1: Treat locked Read TLPs as Supported for EP. Treat locked Read TLPs as UR for RC. [19]: CX_FLT_MASK_CFG_TYPE1_RE_AS_UR 0: Treat CFG type1 TLPs as UR for EP. Treat CFG type1 TLPs as supported for RC. 1: Treat CFG type1 TLPs as Supported for EP. Treat CFG type1 TLPs as UR for RC. [18]: CX_FLT_MASK_UR_OUTSIDE_BAR 0: Treat out-of-bar TLPs as UR 1: Do not treat out-of-bar TLPs as UR [17]: CX_FLT_MASK_UR_POIS 0: Treat poisoned request TLPs as UR 1: Do not treat poisoned request TLPs as UR The native PEX controller always passes poisoned completions to your application. [16]: CX_FLT_MASK_UR_FUNC_MISMATCH 0: Treat Function MisMatched TLPs as UR 1: Do not treat Function MisMatched TLPs as UR *Note*: This register field is sticky.
15 DISABLE_FC_WD_TIMER	Disable FC Watchdog Timer Disable FC Watchdog Timer. *Note*: This register field is sticky.
14-11 —	Reserved
10-0 SKP_INT_VAL	SKP Interval Value The number of symbol times to wait between transmitting SKP ordered sets. Note that the PEX controller actually waits the number of symbol times in this register plus 1 between transmitting SKP ordered sets. Your application must program this register accordingly. For example, if 1536 were programmed into this register (in a 250 MHz PEX controller), then the PEX controller actually transmits SKP ordered sets once every 1537 symbol times. The value programmed to this register is actually clock ticks and not symbol times. In a 125 MHz PEX controller, programming the value programmed to this register should be scaled down by a factor of 2 (because one clock tick = two symbol times in this case). *Note:*This value is not used at Gen3 speed; the skip interval is hardcoded to 370 blocks. *Note*: This register field is sticky.

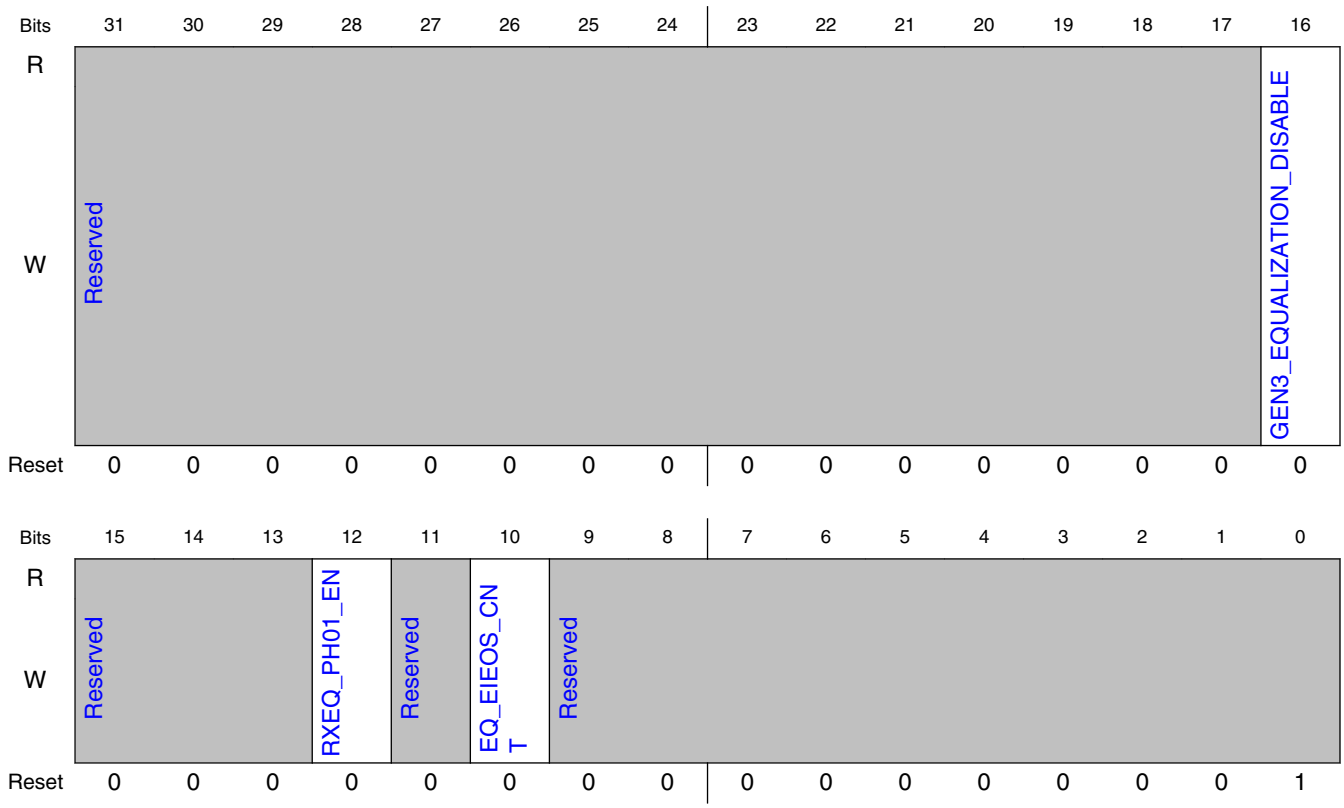
25.4.2.89 Gen3 Control Register (GEN3_RELATED_OFF)

25.4.2.89.1 Offset

Register	Offset
GEN3_RELATED_OFF	890h

25.4.2.89.2 Function

25.4.2.89.3 Diagram



25.4.2.89.4 Fields

Field	Function
31-17 —	Reserved
16 GEN3_EQUALIZATION_DISABLE	Equalization Disable Disable equalization feature. *Note*: This register field is sticky.
15-13 —	Reserved
12 RXEQ_PH01_EN	Rx Equalization Phase 0/Phase 1 Hold Enable When this bit is set the upstream port holds phase 0 (the downstream port holds phase 1) for 10ms. Holding phase 0 or phase 1 can be used to allow sufficient time for Rx Equalization to be performed by the PHY. *Note*: This register field is sticky.
11	Reserved

Table continues on the next page...

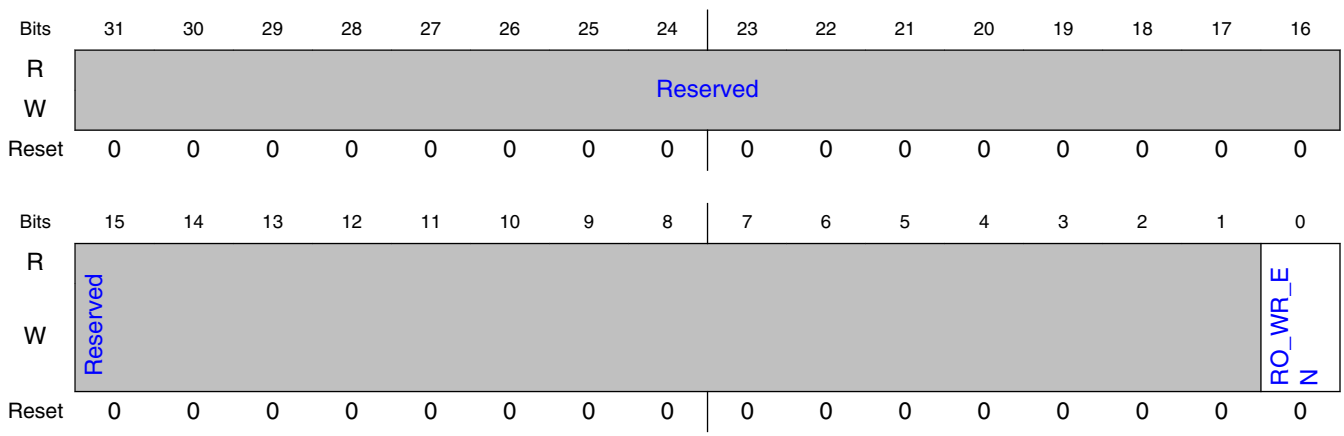
Field	Function
—	
10 EQ_EIEOS_CNT	Equalization EIEOS Count Reset Disable Disable requesting reset of EIEOS count during equalization. *Note*: This register field is sticky.
9-0 —	Reserved

25.4.2.90 DBI Read-only Write Enable Register (MISC_CONTROL_1_OFF)

25.4.2.90.1 Offset

Register	Offset
MISC_CONTROL_1_OFF	8BCh

25.4.2.90.2 Diagram



25.4.2.90.3 Fields

Field	Function
31-1 —	Reserved
0	Read-only write enable

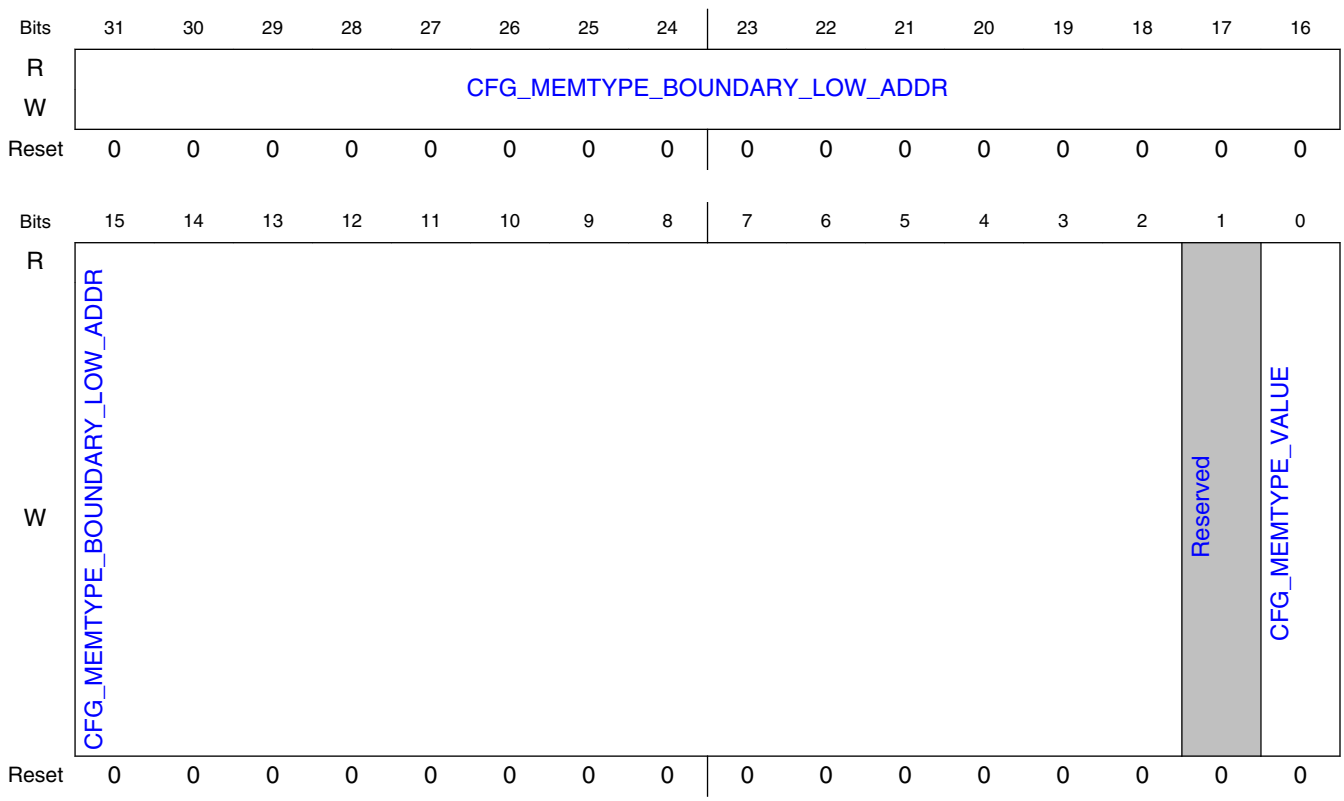
Field	Function
RO_WR_EN	This field enables writing (using internal accesses only) to some read-only and hardware-initialized bits in the configuration registers. 0b - Read-only fields are read only. 1b - Enables writing to some read-only fields

25.4.2.91 Coherency Control Register 1 (COHERENCY_CONTROL_1_OFF)

25.4.2.91.1 Offset

Register	Offset
COHERENCY_CONTROL_1_OFF	8E0h

25.4.2.91.2 Diagram



25.4.2.91.3 Fields

Field	Function
31-2 CFG_MEMTYPE_BOUNDARY_LOW_ADDR	Boundary lower address for memory type Bits [31:0] of dword-aligned address of the boundary for memory type. The two lower address least-significant bits must be 00. Addresses up to but not including this value are in the lower address space region; addresses equal or greater than this value are in the upper address space region. *Note*: This register field is sticky.
1 —	Reserved
0 CFG_MEMTYPE_VALUE	Memory type Sets the memory type for the lower and upper regions of the address space: *Note*: This register field is sticky. 0b - lower = CCSR; upper = Memory 1b - Reserved

25.4.2.92 Coherency Control Register 2 (COHERENCY_CONTROL_2_OFF)

25.4.2.92.1 Offset

Register	Offset
COHERENCY_CONTROL_2_OFF	8E4h

25.4.2.92.2 Diagram

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

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

25.4.2.92.3 Fields

Field	Function
31-0 CFG_MEMTYPE_BOUNDARY_HIGH_ADDR	Boundary upper address for memory type Bits [63:32] of the 64-bit dword-aligned address of the boundary for memory type. *Note*: This register field is sticky.

25.4.2.93 Coherency Control Register 3 (COHERENCY_CONTROL_3_OFF)

25.4.2.93.1 Offset

Register	Offset
COHERENCY_CONTROL_3_OFF	8E8h

25.4.2.93.2 Diagram

Bits	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

Bits	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

25.4.2.93.3 Fields

Field	Function
31-0 —	Reserved. Must be 0000_0000h.

25.4.2.94 iATU Index Register (IATU_VIEWPORT_OFF)

25.4.2.94.1 Offset

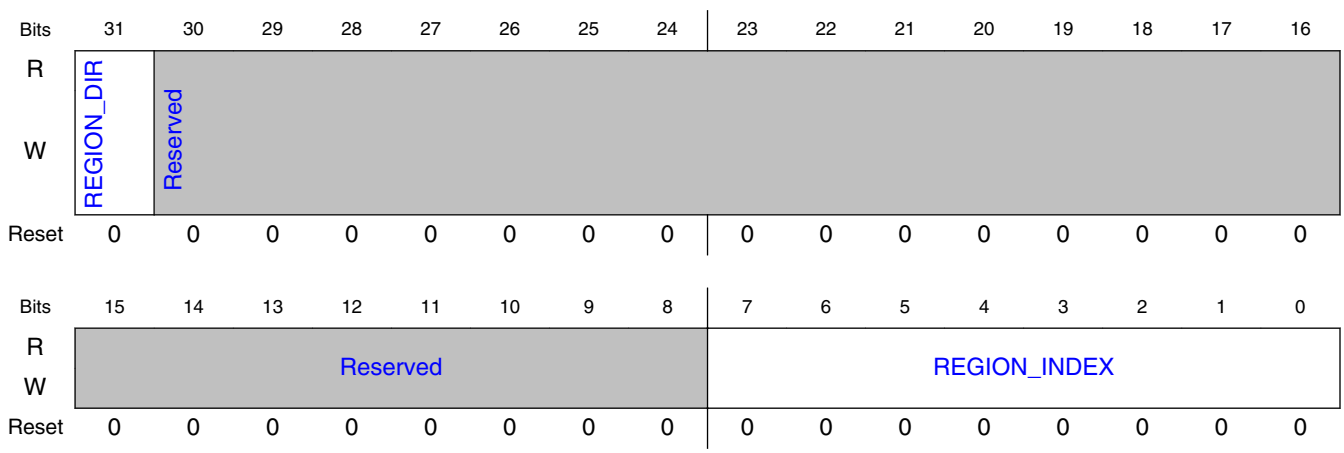
Register	Offset
IATU_VIEWPORT_OFF	900h

25.4.2.94.2 Function

The iATU registers are programmed through an indirect addressing scheme (using this index register) to reduce the address footprint in the PCI Express extended configuration space.

The iATU can remap 2 outbound and 2 inbound address regions. This index register determines the memory region to be accessed.

25.4.2.94.3 Diagram



25.4.2.94.4 Fields

Field	Function
31 REGION_DIR	Region Direction Defines the region being accessed as either 0: Outbound 1: Inbound
30-8 —	Reserved

Table continues on the next page...

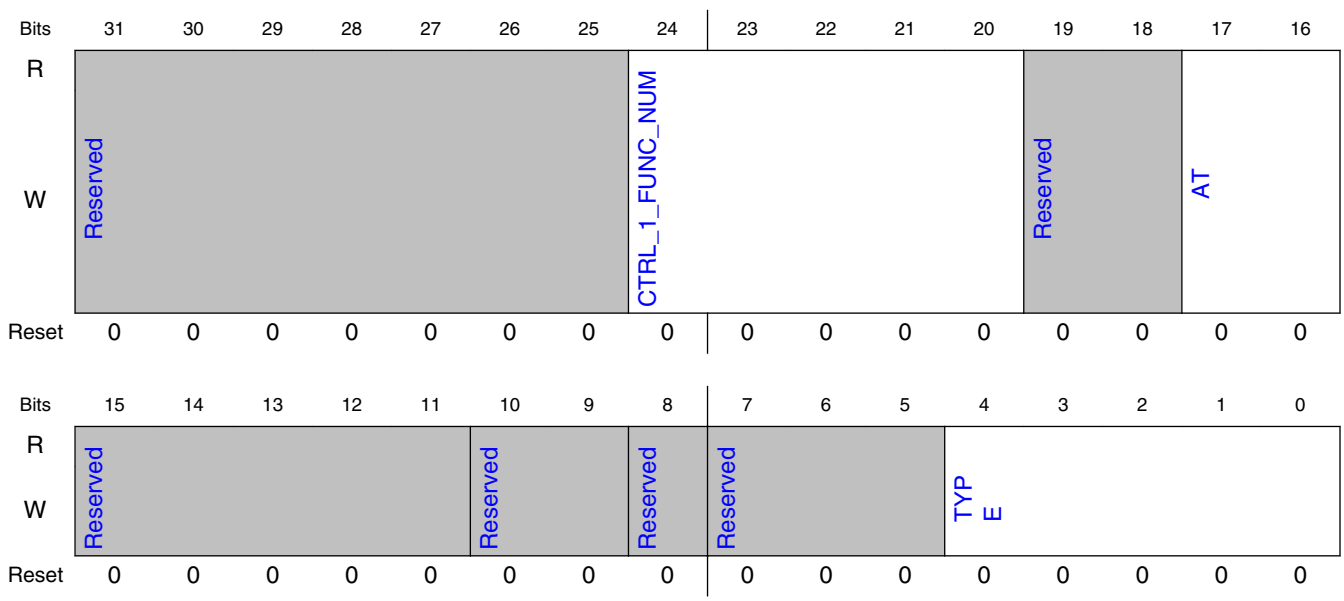
Field	Function
7-0 REGION_INDEX	Region Index Defines which region is being accessed when writing to the control, base, limit and target registers. Must not be set to a number greater than 1 when an outbound region is being accessed. Must not be set to a value greater than 1 when an inbound region is being accessed.

25.4.2.95 iATU Region Control 1 Register (IATU_REGION_CTRL_1_OFF_INBOUND_0)

25.4.2.95.1 Offset

Register	Offset
IATU_REGION_CTRL_1_OFF_INBOUND_0	904h

25.4.2.95.2 Diagram



25.4.2.95.3 Fields

Field	Function
31-25	Reserved

Table continues on the next page...

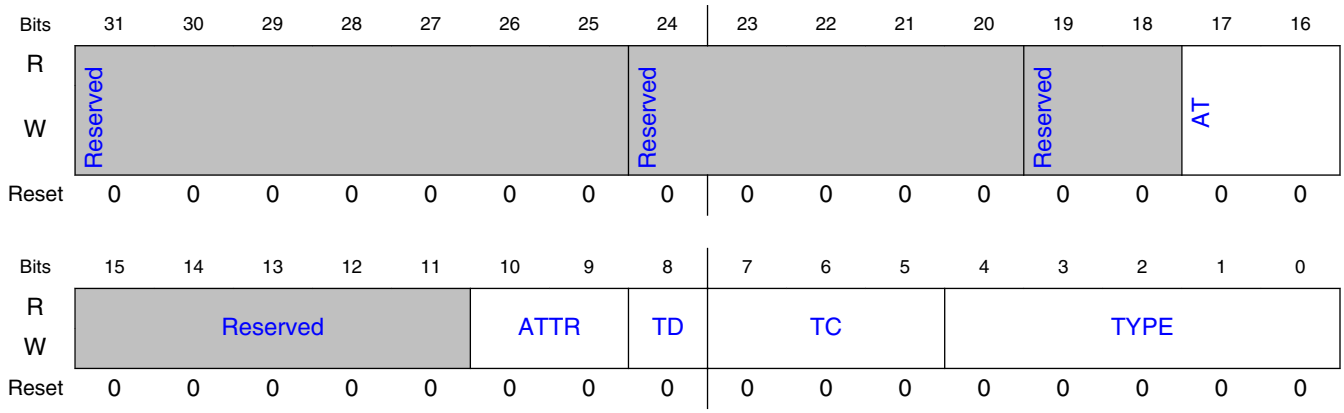
Field	Function
—	
24-20 CTRL_1_FUNC_NUM	Function Number. **MEM-I/O:** DM/EP/SW: When the Address and BAR matching logic in the PEX controller indicate that a MEM-I/O transaction matches a BAR in the function corresponding to this value, then address translation proceeds. This check is only performed if the "Function Number Match Enable" bit of the "iATU Control 2 Register" is set. **CFG0/CFG1:** DM/EP/SW: When the destination function number as specified in the routing ID of the TLP header matches the function, then address translation proceeds. This check is only performed if the "Function Number Match Enable" bit of the "iATU Control 2 Register" is set. RC: Reserved. You must set this bit to "0"
19-18 —	Reserved
17-16 AT	Address Translation (AT) When the TYPE field of an inbound TLP is matched to this value, then address translation proceeds (when all other enabled field-matches are successful). This check is only performed if the "AT Match Enable" bit of the iATU Control 2 Register is set.
15-11 —	Reserved
10-9 —	Reserved
8 —	Reserved
7-5 —	Reserved
4-0 TYPE	Type When the TYPE field of an inbound TLP is matched to this value, then address translation proceeds (when all other enabled field-matches are successful).

25.4.2.96 iATU Region Control 1 Register (IATU_REGION_CTRL_1_OFF_OUTBOUND_0)

25.4.2.96.1 Offset

Register	Offset
IATU_REGION_CTRL_1_OFF_OUTBOUND_0	904h

25.4.2.96.2 Diagram



25.4.2.96.3 Fields

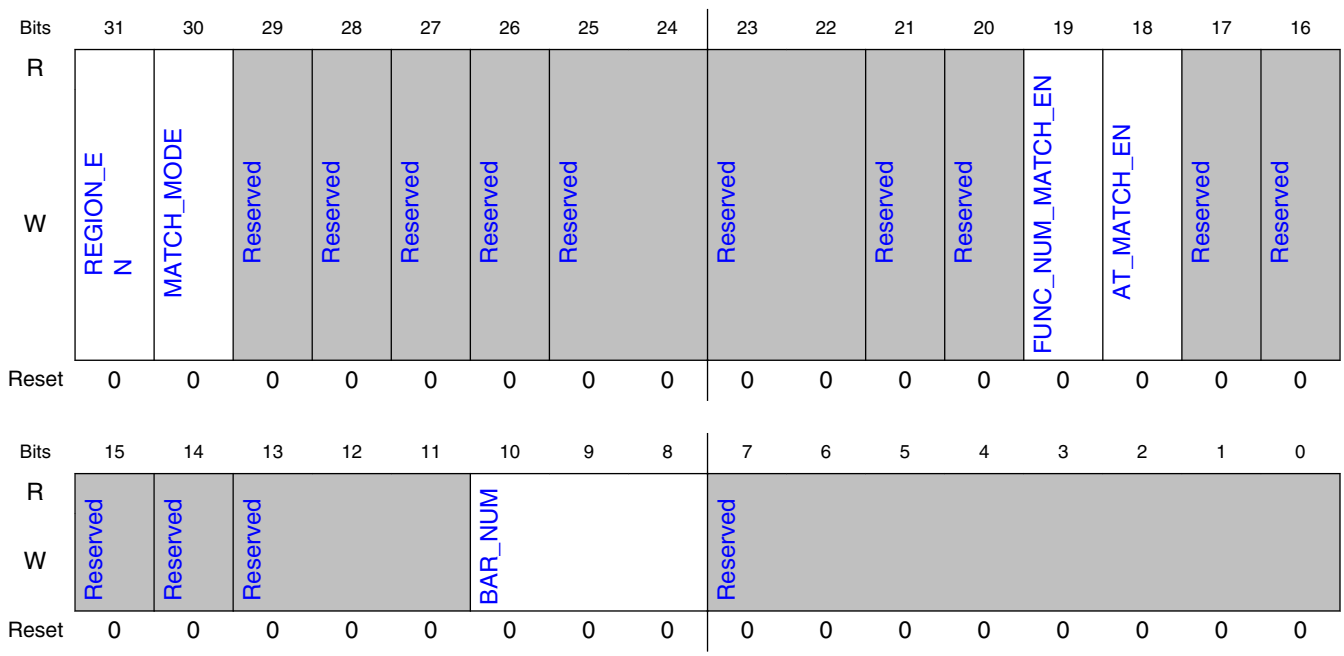
Field	Function
31-25 —	Reserved
24-20 —	Reserved
19-18 —	Reserved
17-16 AT	AT When the address of an outbound TLP is matched to this region, then the AT field of the TLP is changed to the value in this register.
15-11 —	Reserved
10-9 ATTR	Attribute (Attr) When the address of an outbound TLP is matched to this region, then the ATTR field of the TLP is changed to the value in this register.
8 TD	TLP Digest (TD) 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.
7-5 TC	Traffic class (TC) 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.
4-0 TYPE	Type 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.

25.4.2.97 iATU Region Control 2 Register (IATU_REGION_CTRL_2_OFF_INBOUND_0)

25.4.2.97.1 Offset

Register	Offset
IATU_REGION_CTRL_2_OFF_INBOUND_0	908h

25.4.2.97.2 Diagram



25.4.2.97.3 Fields

Field	Function
31 REGION_EN	Region Enable This bit must be set to "1" for address translation to take place.
30 MATCH_MODE	Match Mode. Determines Inbound matching mode for TLPs. The mode depends on the type of TLP that is received as follows: For MEM-I/O TLPs, this field is interpreted as follows: 0: Address Match Mode. The iATU operates using addresses as in the outbound direction. The Region Base and Limit Registers must be setup.

Table continues on the next page...

Field	Function
	1:BAR Match Mode. BAR matching is used. The "BAR Number" field is relevant. Not used for RC. For CFG0 TLPs, this field is interpreted as follows: 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-I/O 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. For MSG/MSGD TLPs, this field is interpreted as follows: 0: Address Match Mode. The iATU treats the third dword and fourth 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 third dword of the TLP header, but matches against the Vendor ID in bits [15:0] of the third 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 fourth dword of the TLP header.
29 —	Reserved
28 —	Reserved
27 —	Reserved
26 —	Reserved
25-24 —	Reserved
23-22 —	Reserved
21 —	Reserved
20 —	Reserved
19 FUNC_NUM_MATCH_EN	Function Number Match Enable **RC/SW:** Reserved. You must set this bit to "0". **DM/EP:** Function Number Match Enable. Ensures that a successful Function Number TLP field comparison match (see Function Number field of the "iATU Control 1 Register") occurs (in MEM-I/O and CFG0/CFG1 transactions) for address translation to proceed.
18 AT_MATCH_EN	AT Match Enable Ensures that a successful AT TLP field comparison match (see AT field of the "iATU Control 1 Register") occurs for address translation to proceed.
17 —	Reserved

Table continues on the next page...

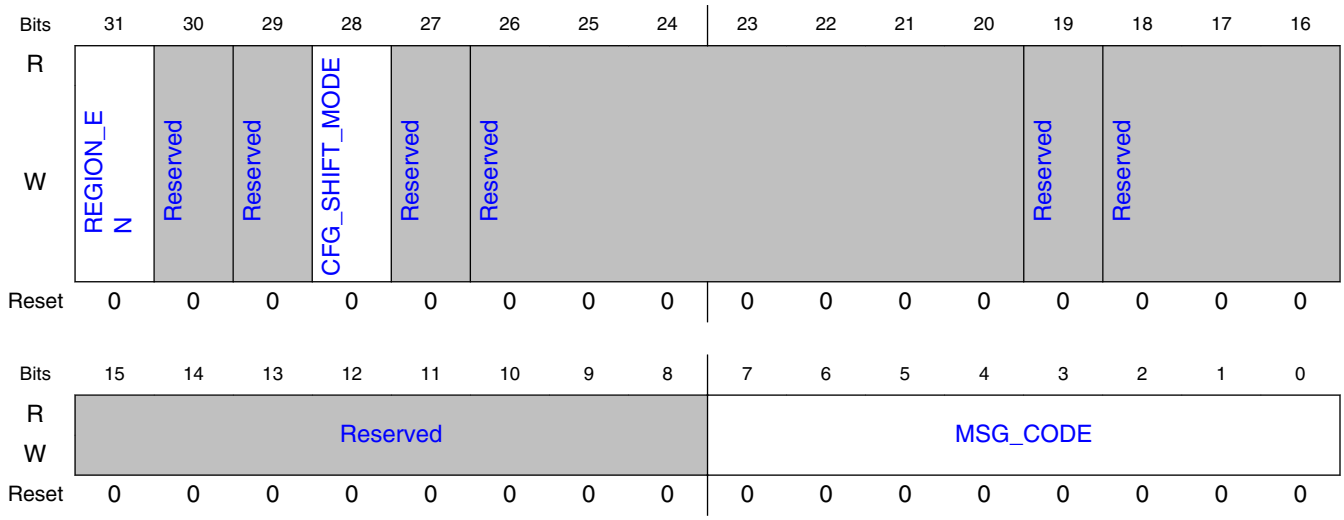
Field	Function
16 —	Reserved
15 —	Reserved
14 —	Reserved
13-11 —	Reserved
10-8 BAR_NUM	BAR Number **DM/EP/SW:** When the BAR number of an inbound MEM or IO TLP that is matched by the normal internal BAR address matching mechanism is the same as this field, address translation proceeds (when all other enabled field-matches are successful). This check is only performed if the "Match Mode" bit of the "iATU Control 2 Register" is set. 000b - BAR0 001b - BAR1 010b - BAR2 011b - BAR3 100b - BAR4 101b - BAR5 110b - ROM 111b - reserved 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. **RC:** Reserved. Must be 0.
7-0 —	Reserved

25.4.2.98 iATU Region Control 2 Register (IATU_REGION_CTRL_2_OFF_OUTBOUND_0)

25.4.2.98.1 Offset

Register	Offset
IATU_REGION_CTRL_2_OFF_OUTBOUND_0	908h

25.4.2.98.2 Diagram



25.4.2.98.3 Fields

Field	Function
31 REGION_EN	Region Enable This bit must be set to "1" for address translation to take place.
30 —	Reserved
29 —	Reserved
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 your application memory space using a 28-bit effective address.Shifts bits [27:12] of the untranslated address to form bits [31:16] of the translated address.
27 —	Reserved
26-20 —	Reserved
19 —	Reserved
18-8 —	Reserved
7-0 MSG_CODE	Message Code Message TLPs: 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.

25.4.2.99 iATU Lower Base Address Register (IATU_LWR_BASE_ADDR_OFF_INBOUND_0)

25.4.2.99.1 Offset

Register	Offset
IATU_LWR_BASE_ADDR_OFF_INBOUND_0	90Ch

25.4.2.99.2 Diagram

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

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	LWR_BASE_RW				LWR_BASE_HW											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

25.4.2.99.3 Fields

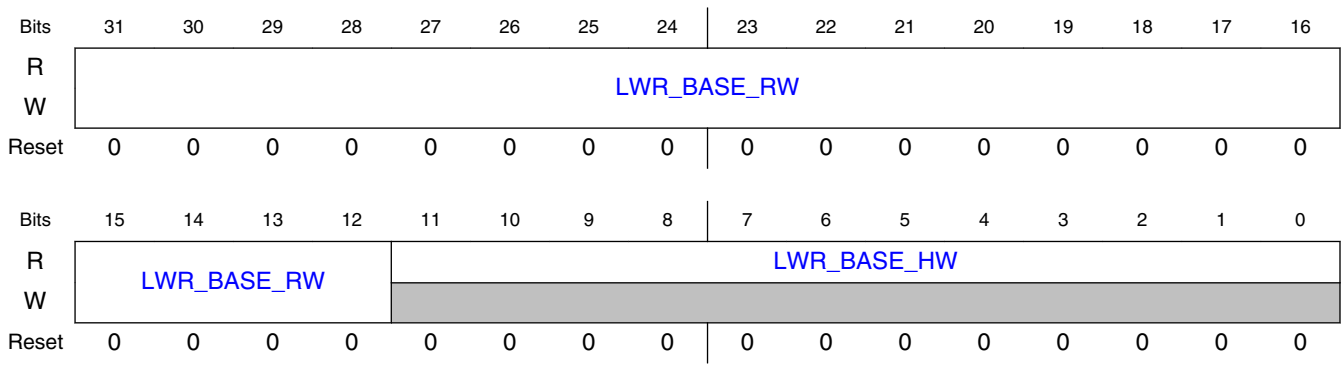
Field	Function
31-12 LWR_BASE_RW	Lower base address bits—programmable Forms bits [31:12] of the start address of the address region to be translated.
11-0 LWR_BASE_HW	Lower base address bits—hardwired Forms bits [11:0] of the start address of the address region to be translated. The start address must be aligned to a 4 KB boundary, so these bits are always 0. A write to this location is ignored by the PEX controller.

25.4.2.100 iATU Lower Base Address Register (IATU_LWR_BASE_ADDR_OFF_OUTBOUND_0)

25.4.2.100.1 Offset

Register	Offset
IATU_LWR_BASE_ADDR_OFF_OUTBOUND_0	90Ch

25.4.2.100.2 Diagram



25.4.2.100.3 Fields

Field	Function
31-12 LWR_BASE_RW	Lower base address bits—programmable Forms bits [31:12] of the start address of the address region to be translated.
11-0 LWR_BASE_HW	Lower base address bits—hardwired Forms bits [11:0] of the start address of the address region to be translated. The start address must be aligned to a 4 KB boundary, so these bits are always 0. A write to this location is ignored by the PEX controller.

25.4.2.101 iATU Upper Base Address Register (IATU_UPPER_BASE_ADDR_OFF_INBOUND_0)

25.4.2.101.1 Offset

Register	Offset
IATU_UPPER_BASE_ADDR_OFF_INBOUND_0	910h

25.4.2.101.2 Diagram

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

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

25.4.2.101.3 Fields

Field	Function
31-0 UPPER_BASE_RW	Upper base address bits Forms bits [63:32] of the start (and end) address of the address region to be translated.

25.4.2.102 iATU Upper Base Address Register (IATU_UPPER_BASE_ADDR_OFF_OUTBOUND_0)

25.4.2.102.1 Offset

Register	Offset
IATU_UPPER_BASE_ADDR_OFF_OUTBOUND_0	910h

25.4.2.102.2 Diagram

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

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

25.4.2.102.3 Fields

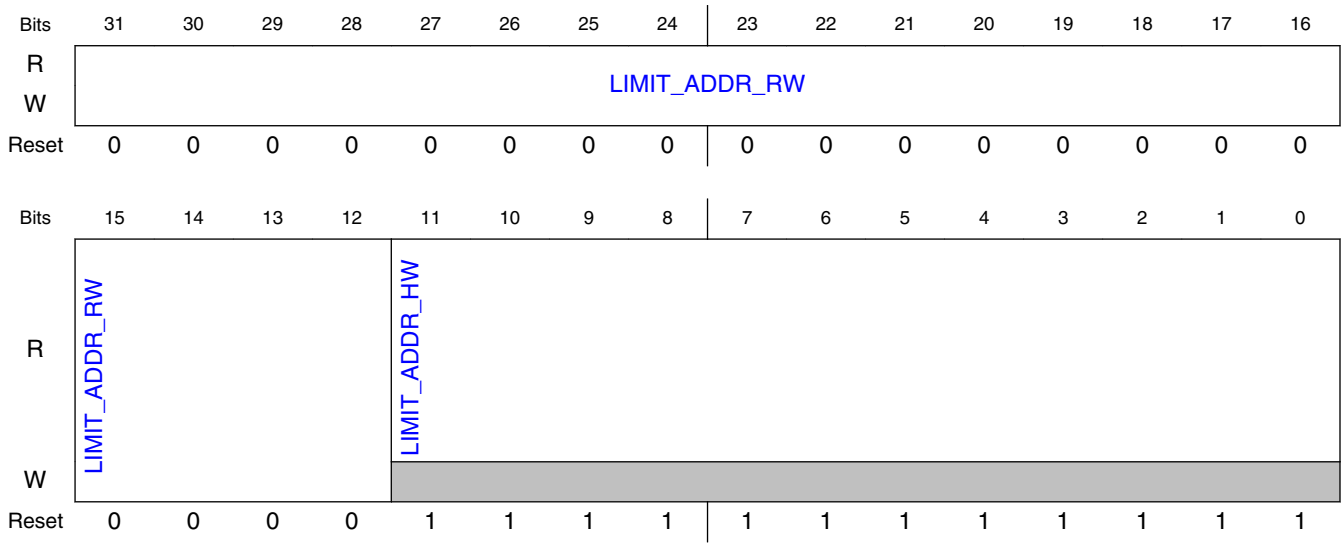
Field	Function
31-0 UPPER_BASE_RW	Upper base address bits Forms bits [63:32] of the start (and end) address of the address region to be translated. In systems with a 32-bit address space, this register is not used and therefore writing to this register has no effect.

25.4.2.103 iATU Limit Address Register (IATU_LIMIT_ADDR_OFF_INBOUND_0)

25.4.2.103.1 Offset

Register	Offset
IATU_LIMIT_ADDR_OFF_INBOUND_0	914h

25.4.2.103.2 Diagram



25.4.2.103.3 Fields

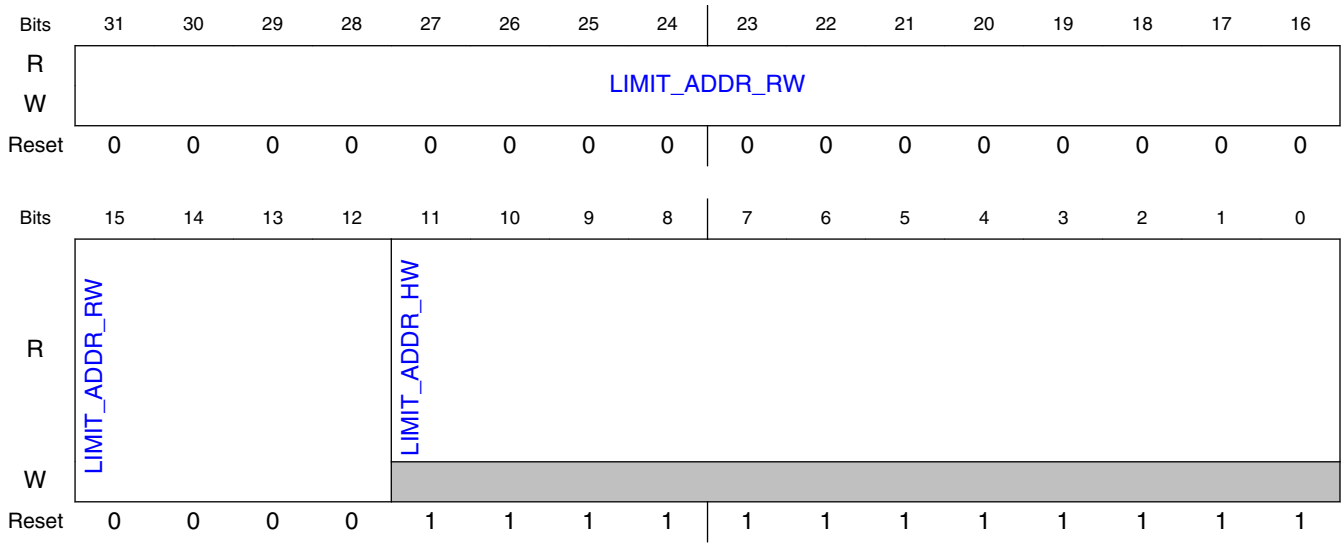
Field	Function
31-12 LIMIT_ADDR_RW	Limit address bits—programmable Forms bits [31:12] of the end address of the address region to be translated.
11-0 LIMIT_ADDR_HW	Limit address bits—hardwired Forms bits [11:0] of the end address of the address region to be translated. Address regions must be aligned to a 4 KB boundary, so these bits are always 1. A write to this location is ignored by the PEX controller.

25.4.2.104 iATU Limit Address Register (IATU_LIMIT_ADDR_OFF_OUTBOUND_0)

25.4.2.104.1 Offset

Register	Offset
IATU_LIMIT_ADDR_OFF_OUTBOUND_0	914h

25.4.2.104.2 Diagram



25.4.2.104.3 Fields

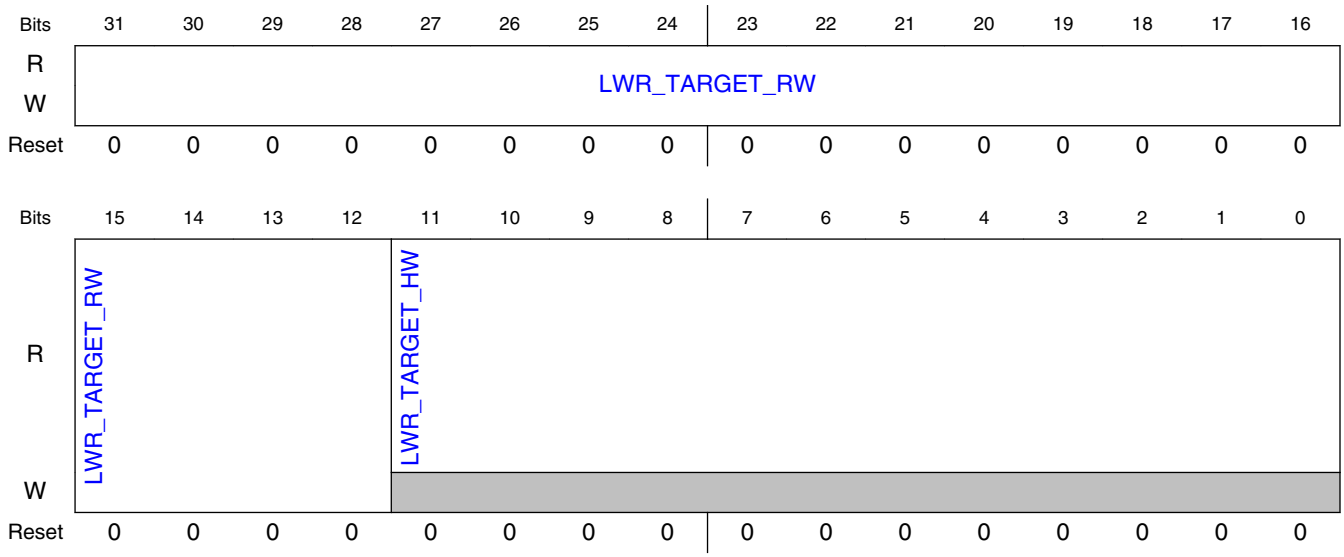
Field	Function
31-12 LIMIT_ADDR_RW	Limit address bits—programmable Forms bits [31:12] of the end address of the address region to be translated.
11-0 LIMIT_ADDR_HW	Limit address bits—hardwired Forms bits [11:0] of the end address of the address region to be translated. Address regions must be aligned to a 4 KB boundary, so these bits are always 1. A write to this location is ignored by the PEX controller.

25.4.2.105 iATU Region#N Lower Offset Address Register (IATU_LWR_TARGET_ADDR_OFF_INBOUND_0)

25.4.2.105.1 Offset

Register	Offset
IATU_LWR_TARGET_ADDR_OFF_INBOUND_0	918h

25.4.2.105.2 Diagram



25.4.2.105.3 Fields

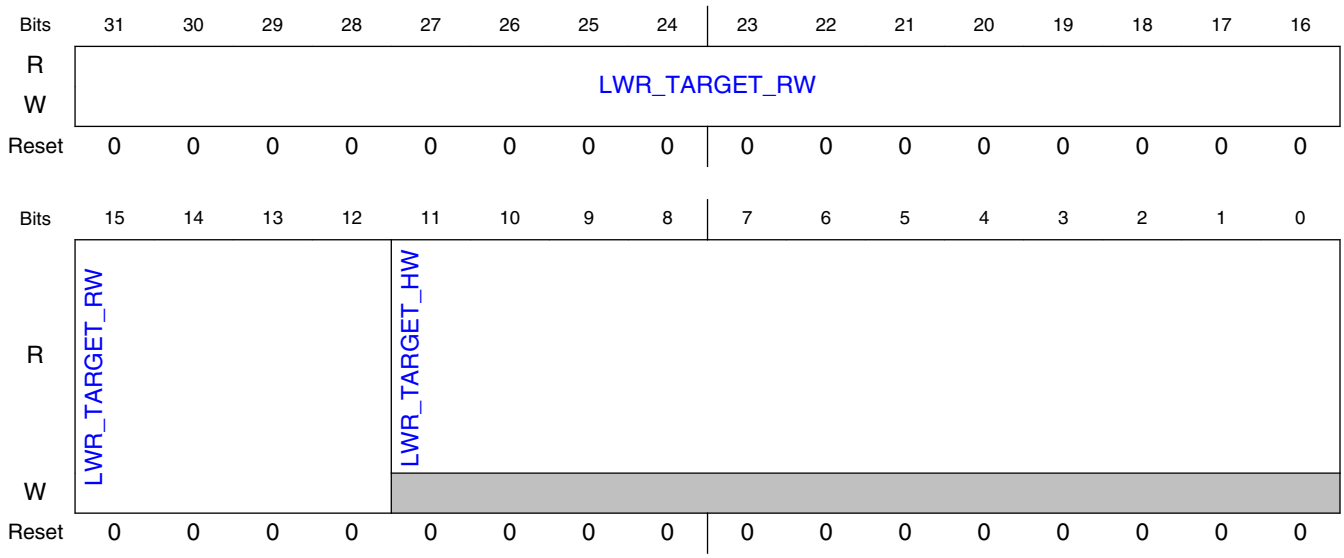
Field	Function
31-12 LWR_TARGET_RW	Lower target address bits—programmable Forms bits [31:12] of the of the new address of the translated region.
11-0 LWR_TARGET_HW	Lower target address bits—hardwired Forms bits [11:0] of the start address of the new address of the translated region. The start address must be aligned to a 4 KB boundary (in address match mode); and to the BAR size boundary (in BAR match mode) so these bits are always 0. If the BAR is smaller than the iATU region size, then the iATU target address must align to the iATU region size; otherwise it must align to the BAR size. A write to this location is ignored by the PEX controller.

25.4.2.106 iATU Outbound Region#N Lower Offset Address Register (IATU_LWR_TARGET_ADDR_OFF_OUTBOUND_0)

25.4.2.106.1 Offset

Register	Offset
IATU_LWR_TARGET_ADDR_OFF_OUTBOUND_0	918h

25.4.2.106.2 Diagram



25.4.2.106.3 Fields

Field	Function
31-12 LWR_TARGET_RW	Lower target address bits—programmable Forms bits [31:12] of the of the new address of the translated region.
11-0 LWR_TARGET_HW	Lower target address bits—hardwired Forms bits [11:0] of the start address of the new address of the translated region. The start address must be aligned to a 4 KB boundary, so these bits are always 0. A write to this location is ignored by the PEX controller.

25.4.2.107 iATU Upper Target Address Register (IATU_UPPER_TARGET_ADDR_OFF_INBOUND_0)

25.4.2.107.1 Offset

Register	Offset
IATU_UPPER_TARGET_ADDR_OFF_INBOUND_0	91Ch

25.4.2.107.2 Diagram

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

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

25.4.2.107.3 Fields

Field	Function
31-0 UPPER_TARGET_RW	Upper target address bits Forms bits [63:32] of the start address of the new address of the translated region. In systems with a 32-bit address space, this register is not used and therefore writing to this register has no effect.

25.4.2.108 iATU Upper Target Address Register (IATU_UPPER_TARGET_ADDR_OFF_OUTBOUND_0)

25.4.2.108.1 Offset

Register	Offset
IATU_UPPER_TARGET_ADDR_OFF_OUTBOUND_0	91Ch

25.4.2.108.2 Diagram

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

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

25.4.2.108.3 Fields

Field	Function
31-0 UPPER_TARGET_RW	Upper target address bits Forms bits [63:32] of the start address of the new address of the translated region.

25.4.2.109 Base Address Register 0 Mask (BAR0_MASK)

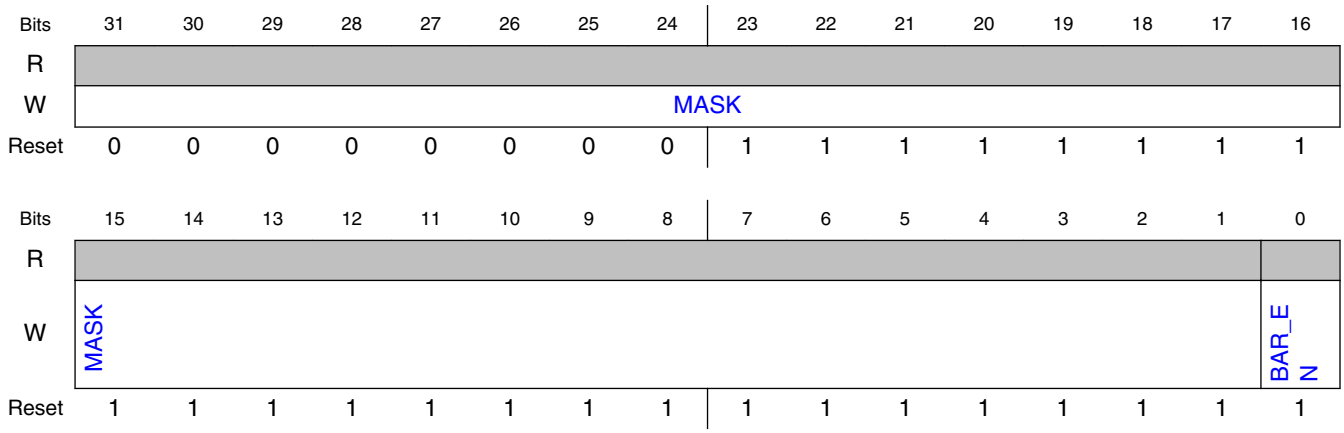
25.4.2.109.1 Offset

Register	Offset
BAR0_MASK	1010h

25.4.2.109.2 Function

The Base Address Register Mask is used to control size of memory requested by the corresponding PCI Express base address register (BAR) during initialization. When a mask bit is set to 1, the corresponding address bit in the BAR is masked and cannot be written, returning a 0 when read.

25.4.2.109.3 Diagram



25.4.2.109.4 Fields

Field	Function
31-1 MASK	Mask
0 BAR_EN	BAR enable. 0b - Disable BAR 1b - Enable BAR

25.4.2.110 Base Address Register 1 Mask (BAR1_MASK)

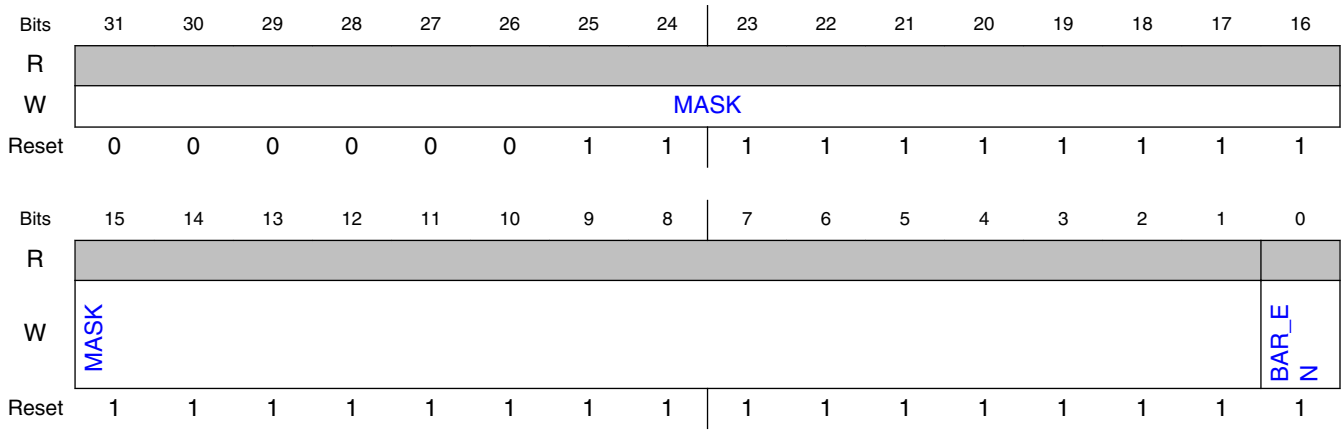
25.4.2.110.1 Offset

Register	Offset
BAR1_MASK	1014h

25.4.2.110.2 Function

The Base Address Register Mask is used to control size of memory requested by the corresponding PCI Express base address register (BAR) during initialization. When a mask bit is set to 1, the corresponding address bit in the BAR is masked and cannot be written, returning a 0 when read.

25.4.2.110.3 Diagram



25.4.2.110.4 Fields

Field	Function
31-1 MASK	Mask
0 BAR_EN	BAR enable. 0b - Disable BAR 1b - Enable BAR

25.4.2.111 Base Address Register 2 Mask (BAR2_MASK)

25.4.2.111.1 Offset

Register	Offset
BAR2_MASK	1018h

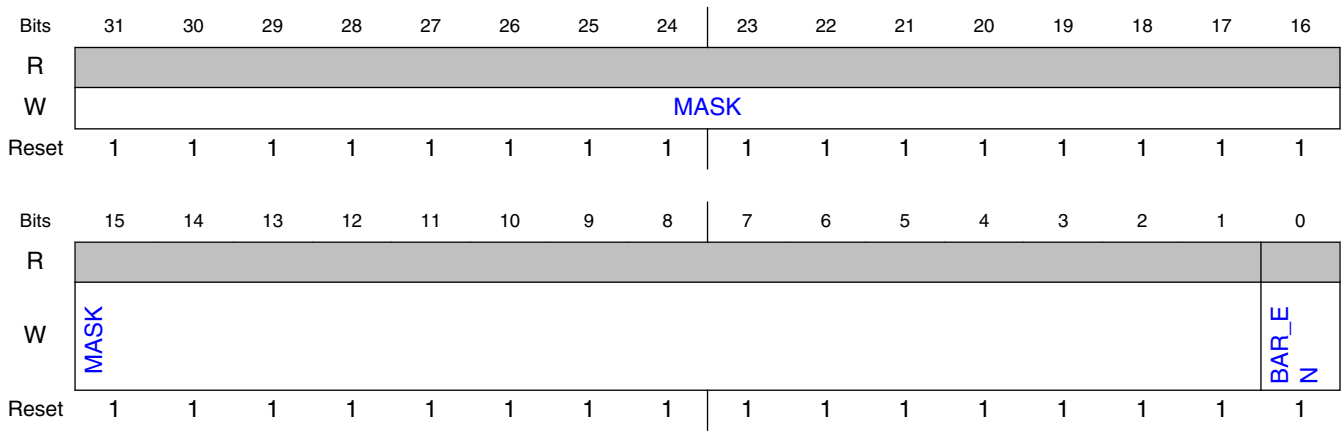
25.4.2.111.2 Function

NOTE

This register is supported only in EP mode; it does not exist in RC mode.

The Base Address Register Mask is used to control size of memory requested by the corresponding PCI Express base address register (BAR) during initialization. When a mask bit is set to 1, the corresponding address bit in the BAR is masked and cannot be written, returning a 0 when read.

25.4.2.111.3 Diagram



25.4.2.111.4 Fields

Field	Function
31-1 MASK	Mask
0 BAR_EN	BAR enable. 0b - Disable BAR 1b - Enable BAR

25.4.2.112 Base Address Register 3 Mask (BAR3_MASK)

25.4.2.112.1 Offset

Register	Offset
BAR3_MASK	101Ch

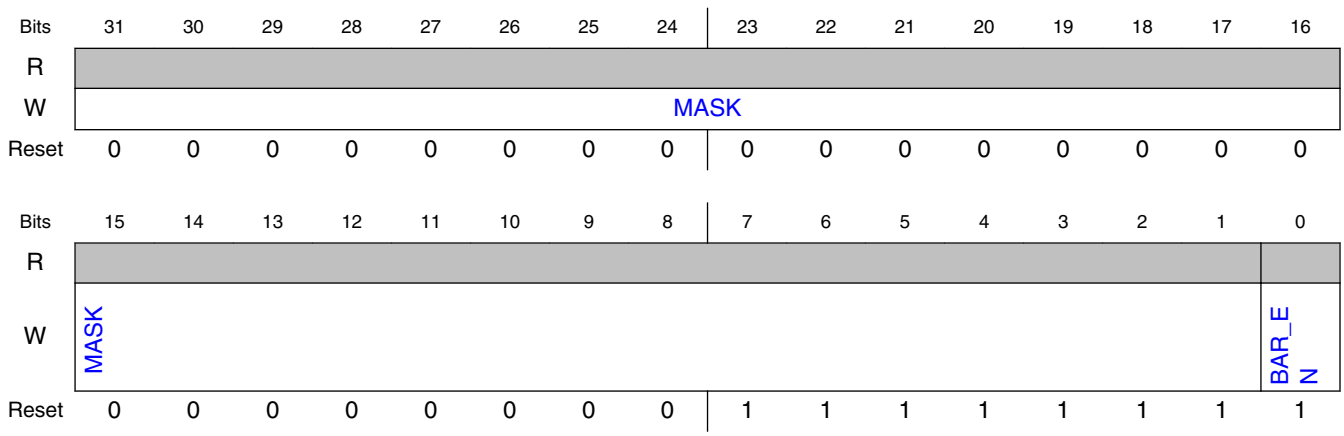
25.4.2.112.2 Function

NOTE

This register is supported only in EP mode; it does not exist in RC mode.

The Base Address Register Mask is used to control size of memory requested by the corresponding PCI Express base address register (BAR) during initialization. When a mask bit is set to 1, the corresponding address bit in the BAR is masked and cannot be written, returning a 0 when read.

25.4.2.112.3 Diagram



25.4.2.112.4 Fields

Field	Function
31-1 MASK	Mask
0 BAR_EN	BAR enable. 0b - Disable BAR 1b - Enable BAR

25.4.2.113 Base Address Register 4 Mask (BAR4_MASK)

25.4.2.113.1 Offset

Register	Offset
BAR4_MASK	1020h

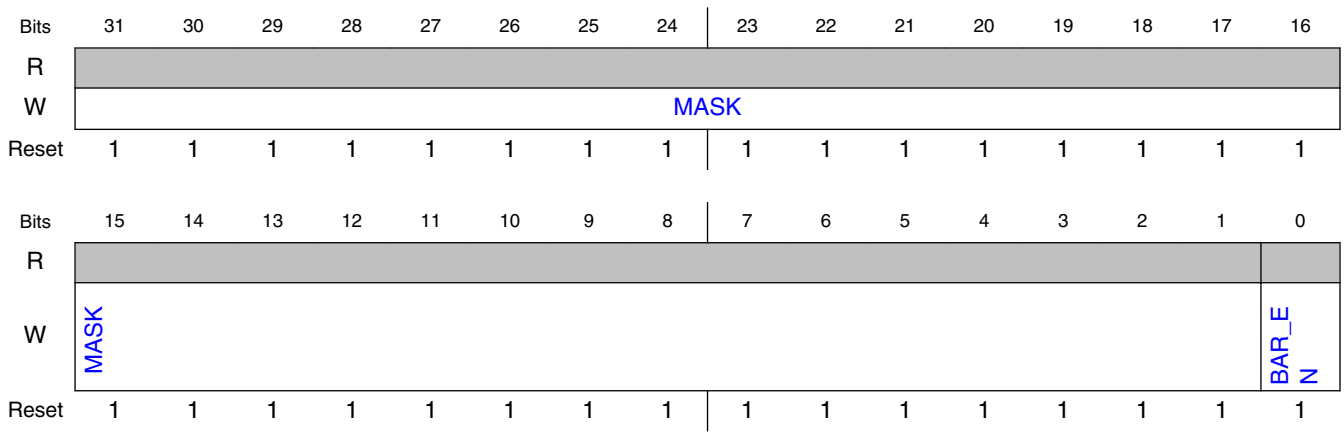
25.4.2.113.2 Function

NOTE

This register is supported only in EP mode; it does not exist in RC mode.

The Base Address Register Mask is used to control size of memory requested by the corresponding PCI Express base address register (BAR) during initialization. When a mask bit is set to 1, the corresponding address bit in the BAR is masked and cannot be written, returning a 0 when read.

25.4.2.113.3 Diagram



25.4.2.113.4 Fields

Field	Function
31-1 MASK	Mask
0 BAR_EN	BAR enable. 0b - Disable BAR 1b - Enable BAR

25.4.2.114 Base Address Register 5 Mask (BAR5_MASK)

25.4.2.114.1 Offset

Register	Offset
BAR5_MASK	1024h

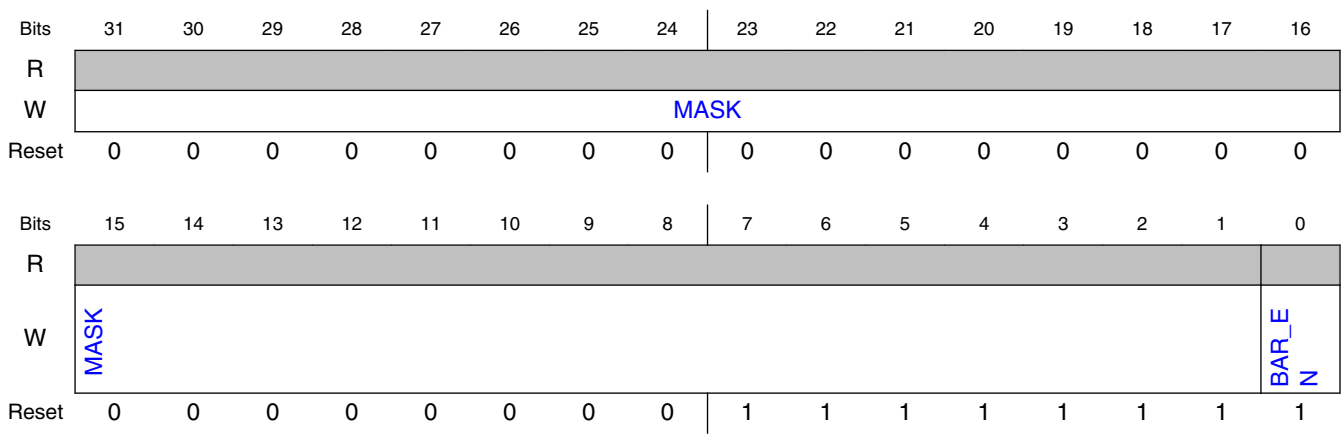
25.4.2.114.2 Function

NOTE

This register is supported only in EP mode; it does not exist in RC mode.

The Base Address Register Mask is used to control size of memory requested by the corresponding PCI Express base address register (BAR) during initialization. When a mask bit is set to 1, the corresponding address bit in the BAR is masked and cannot be written, returning a 0 when read.

25.4.2.114.3 Diagram



25.4.2.114.4 Fields

Field	Function
31-1 MASK	Mask
0 BAR_EN	BAR enable. 0b - Disable BAR 1b - Enable BAR

25.4.2.115 Expansion ROM Base Address Register Mask (EP mode) (EXP_ROM_BAR_MASK_EP)

25.4.2.115.1 Offset

Register	Offset
EXP_ROM_BAR_MASK_EP	1030h

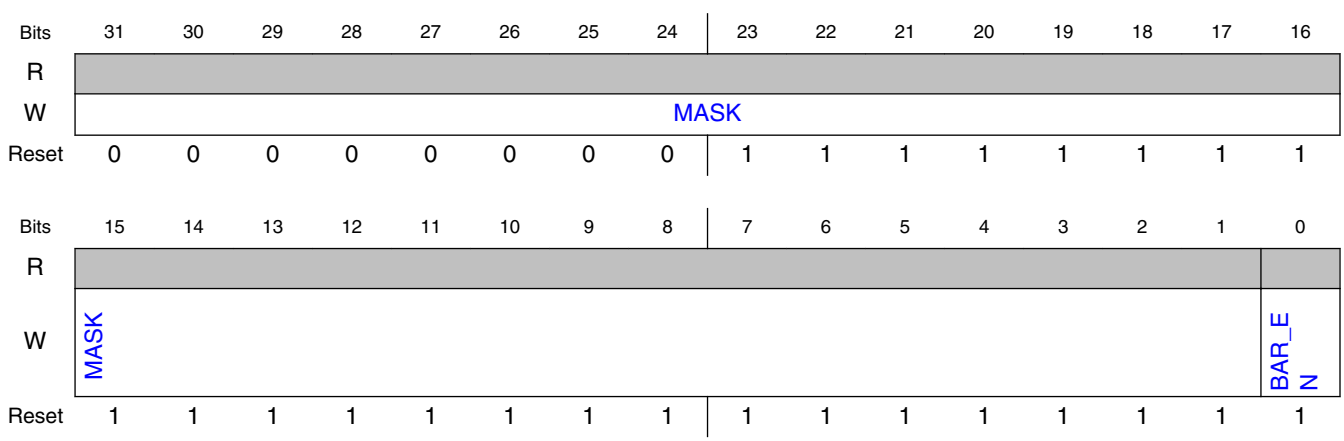
25.4.2.115.2 Function

NOTE

This register is supported only in EP mode; the Expansion ROM Base Address Register Mask for RC mode is located at offset 0x1038.

The Base Address Register Mask is used to control size of memory requested by the corresponding PCI Express Expansion ROM Base Address Register (EP-Mode) during initialization. When a mask bit is set to 1, the corresponding address bit in the BAR is masked and cannot be written, returning a 0 when read.

25.4.2.115.3 Diagram



25.4.2.115.4 Fields

Field	Function
31-1 MASK	Expansion ROM BAR Mask

Table continues on the next page...

Field	Function
0 BAR_EN	Expansion ROM BAR enable. 0b - Disable Expansion ROM BAR 1b - Enable Expansion ROM BAR

25.4.2.116 Expansion ROM Base Address Register Mask (RC mode) (EXP_ROM_BAR_MASK_RC)

25.4.2.116.1 Offset

Register	Offset
EXP_ROM_BAR_MASK_RC	1038h

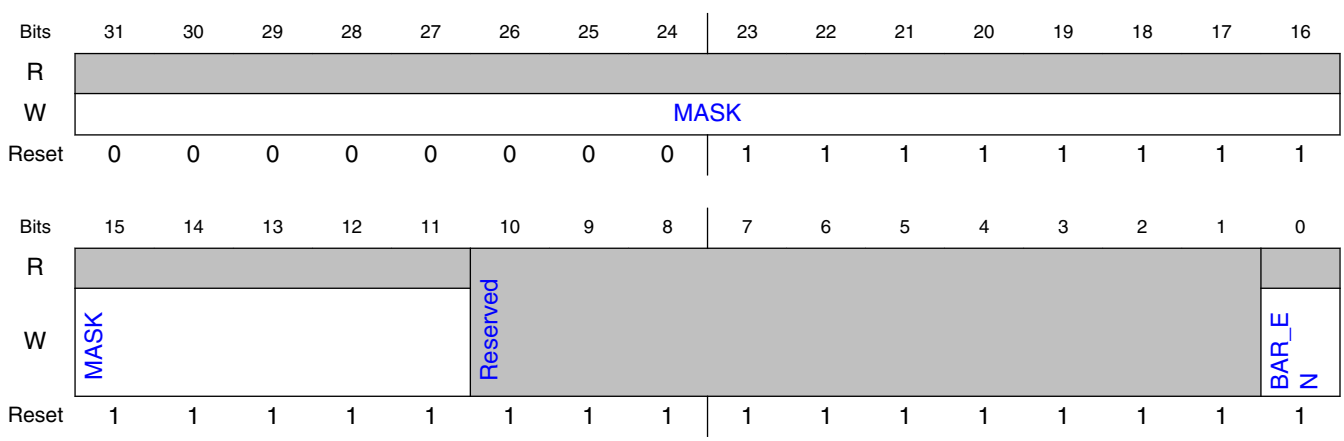
25.4.2.116.2 Function

NOTE

This register is supported only in RC mode; the Expansion ROM Base Address Register Mask for EP mode is located at offset 0x1030.

The Expansion ROM Base Address Register Mask is used to control size of memory requested by the corresponding PCI Express Expansion ROM Base Address Register (RC-Mode) during initialization. When a mask bit is set to 1, the corresponding address bit in the BAR is masked and cannot be written, returning a 0 when read.

25.4.2.116.3 Diagram



25.4.2.116.4 Fields

Field	Function
31-11 MASK	Expansion ROM BAR Mask
10-1 —	Reserved.
0 BAR_EN	Expansion ROM BAR enable. 0b - Disable Expansion ROM BAR 1b - Enable Expansion ROM BAR

25.5 PEX_LUT memory map/registers overview

PEX_LUT provides registers for:

- mapping AMQ attributes to request IDs.
- generating INTx and PME messages
- controlling errors (disable, detect, and interrupt enable)
- generating a soft-reset
- determining the current LTSSM state

25.5.1 PEX_LUT register descriptions

25.5.1.1 PEX_LUT Memory map

PEX1_LUT base address: 348_0000h

Offset	Register	Width (In bits)	Access	Reset value
20h	PEX LUT Status Register (PEXLSR)	32	W1C	0000_0000h
24h	PEX LUT Control Register (PEXLCR)	32	RW	0000_0000h
800h	PEX LUT Entry a Upper Data Register (PEXL0UDR)	32	RW	0000_FFFFh
804h	PEX LUT Entry a Lower Data Register (PEXL0LDR)	32	RW	8000_0000h
808h	PEX LUT Entry a Upper Data Register (PEXL1UDR)	32	RW	0000_0000h
80Ch	PEX LUT Entry a Lower Data Register (PEXL1LDR)	32	RW	0000_0000h
810h	PEX LUT Entry a Upper Data Register (PEXL2UDR)	32	RW	0000_0000h
814h	PEX LUT Entry a Lower Data Register (PEXL2LDR)	32	RW	0000_0000h

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
818h	PEX LUT Entry a Upper Data Register (PEXL3UDR)	32	RW	0000_0000h
81Ch	PEX LUT Entry a Lower Data Register (PEXL3LDR)	32	RW	0000_0000h
820h	PEX LUT Entry a Upper Data Register (PEXL4UDR)	32	RW	0000_0000h
824h	PEX LUT Entry a Lower Data Register (PEXL4LDR)	32	RW	0000_0000h
828h	PEX LUT Entry a Upper Data Register (PEXL5UDR)	32	RW	0000_0000h
82Ch	PEX LUT Entry a Lower Data Register (PEXL5LDR)	32	RW	0000_0000h
830h	PEX LUT Entry a Upper Data Register (PEXL6UDR)	32	RW	0000_0000h
834h	PEX LUT Entry a Lower Data Register (PEXL6LDR)	32	RW	0000_0000h
838h	PEX LUT Entry a Upper Data Register (PEXL7UDR)	32	RW	0000_0000h
83Ch	PEX LUT Entry a Lower Data Register (PEXL7LDR)	32	RW	0000_0000h
840h	PEX LUT Entry a Upper Data Register (PEXL8UDR)	32	RW	0000_0000h
844h	PEX LUT Entry a Lower Data Register (PEXL8LDR)	32	RW	0000_0000h
848h	PEX LUT Entry a Upper Data Register (PEXL9UDR)	32	RW	0000_0000h
84Ch	PEX LUT Entry a Lower Data Register (PEXL9LDR)	32	RW	0000_0000h
850h	PEX LUT Entry a Upper Data Register (PEXL10UDR)	32	RW	0000_0000h
854h	PEX LUT Entry a Lower Data Register (PEXL10LDR)	32	RW	0000_0000h
858h	PEX LUT Entry a Upper Data Register (PEXL11UDR)	32	RW	0000_0000h
85Ch	PEX LUT Entry a Lower Data Register (PEXL11LDR)	32	RW	0000_0000h
860h	PEX LUT Entry a Upper Data Register (PEXL12UDR)	32	RW	0000_0000h
864h	PEX LUT Entry a Lower Data Register (PEXL12LDR)	32	RW	0000_0000h
868h	PEX LUT Entry a Upper Data Register (PEXL13UDR)	32	RW	0000_0000h
86Ch	PEX LUT Entry a Lower Data Register (PEXL13LDR)	32	RW	0000_0000h
870h	PEX LUT Entry a Upper Data Register (PEXL14UDR)	32	RW	0000_0000h
874h	PEX LUT Entry a Lower Data Register (PEXL14LDR)	32	RW	0000_0000h
878h	PEX LUT Entry a Upper Data Register (PEXL15UDR)	32	RW	0000_0000h
87Ch	PEX LUT Entry a Lower Data Register (PEXL15LDR)	32	RW	0000_0000h
880h	PEX LUT Entry a Upper Data Register (PEXL16UDR)	32	RW	0000_0000h
884h	PEX LUT Entry a Lower Data Register (PEXL16LDR)	32	RW	0000_0000h
888h	PEX LUT Entry a Upper Data Register (PEXL17UDR)	32	RW	0000_0000h
88Ch	PEX LUT Entry a Lower Data Register (PEXL17LDR)	32	RW	0000_0000h
890h	PEX LUT Entry a Upper Data Register (PEXL18UDR)	32	RW	0000_0000h
894h	PEX LUT Entry a Lower Data Register (PEXL18LDR)	32	RW	0000_0000h
898h	PEX LUT Entry a Upper Data Register (PEXL19UDR)	32	RW	0000_0000h
89Ch	PEX LUT Entry a Lower Data Register (PEXL19LDR)	32	RW	0000_0000h
8A0h	PEX LUT Entry a Upper Data Register (PEXL20UDR)	32	RW	0000_0000h
8A4h	PEX LUT Entry a Lower Data Register (PEXL20LDR)	32	RW	0000_0000h
8A8h	PEX LUT Entry a Upper Data Register (PEXL21UDR)	32	RW	0000_0000h
8ACh	PEX LUT Entry a Lower Data Register (PEXL21LDR)	32	RW	0000_0000h
8B0h	PEX LUT Entry a Upper Data Register (PEXL22UDR)	32	RW	0000_0000h
8B4h	PEX LUT Entry a Lower Data Register (PEXL22LDR)	32	RW	0000_0000h

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
8B8h	PEX LUT Entry a Upper Data Register (PEXL23UDR)	32	RW	0000_0000h
8BCh	PEX LUT Entry a Lower Data Register (PEXL23LDR)	32	RW	0000_0000h
8C0h	PEX LUT Entry a Upper Data Register (PEXL24UDR)	32	RW	0000_0000h
8C4h	PEX LUT Entry a Lower Data Register (PEXL24LDR)	32	RW	0000_0000h
8C8h	PEX LUT Entry a Upper Data Register (PEXL25UDR)	32	RW	0000_0000h
8CCh	PEX LUT Entry a Lower Data Register (PEXL25LDR)	32	RW	0000_0000h
8D0h	PEX LUT Entry a Upper Data Register (PEXL26UDR)	32	RW	0000_0000h
8D4h	PEX LUT Entry a Lower Data Register (PEXL26LDR)	32	RW	0000_0000h
8D8h	PEX LUT Entry a Upper Data Register (PEXL27UDR)	32	RW	0000_0000h
8DCh	PEX LUT Entry a Lower Data Register (PEXL27LDR)	32	RW	0000_0000h
8E0h	PEX LUT Entry a Upper Data Register (PEXL28UDR)	32	RW	0000_0000h
8E4h	PEX LUT Entry a Lower Data Register (PEXL28LDR)	32	RW	0000_0000h
8E8h	PEX LUT Entry a Upper Data Register (PEXL29UDR)	32	RW	0000_0000h
8ECh	PEX LUT Entry a Lower Data Register (PEXL29LDR)	32	RW	0000_0000h
8F0h	PEX LUT Entry a Upper Data Register (PEXL30UDR)	32	RW	0000_0000h
8F4h	PEX LUT Entry a Lower Data Register (PEXL30LDR)	32	RW	0000_0000h
8F8h	PEX LUT Entry a Upper Data Register (PEXL31UDR)	32	RW	0000_0000h
8FCh	PEX LUT Entry a Lower Data Register (PEXL31LDR)	32	RW	0000_0000h
4_0014h	PEX PFa Config Register (PEX_PF0_CONFIG)	32	RW	0000_0000h
4_0018h	PEX PFa Interrupt status Register (PEX_PF0_INT_STAT)	32	RO	0000_0000h
4_0020h	PEX PFa PCIE pme and message detect register (PEX_PF0_PME_MES_DR)	32	W1C	0000_0000h
4_0024h	PEX PFa PCIE pme and message disable register (PEX_PF0_PME_MES_DISR)	32	RW	0000_0000h
4_0028h	PEX PFa PCIE pme and message interrupt enable register (PEX_PF0_PME_MES_IER)	32	RW	0000_0000h
4_002Ch	PEX PFa PCIE message command register (PEX_PF0_MCR)	32	RW	0000_0000h
4_0140h	PEX PFa Route By Port Address Upper register (PEX_PF0_RBP_ADDR_U)	32	RW	0000_0000h
4_0200h	PEX PFa PCIE error detect register (PEX_PF0_ERR_DR)	32	W1C	0000_0000h
4_0208h	PEX PFa PCIE error interrupt enable register (PEX_PF0_ERR_EN)	32	RW	0000_0000h
4_0210h	PEX PFa PCIE error disable register (PEX_PF0_ERR_DISR)	32	RW	0000_0000h
4_07FCh	PEX PF0 Debug register (PEX_PF0_DBG)	32	RW	0000_0000h

25.5.1.2 PEX LUT Status Register (PEXLSR)

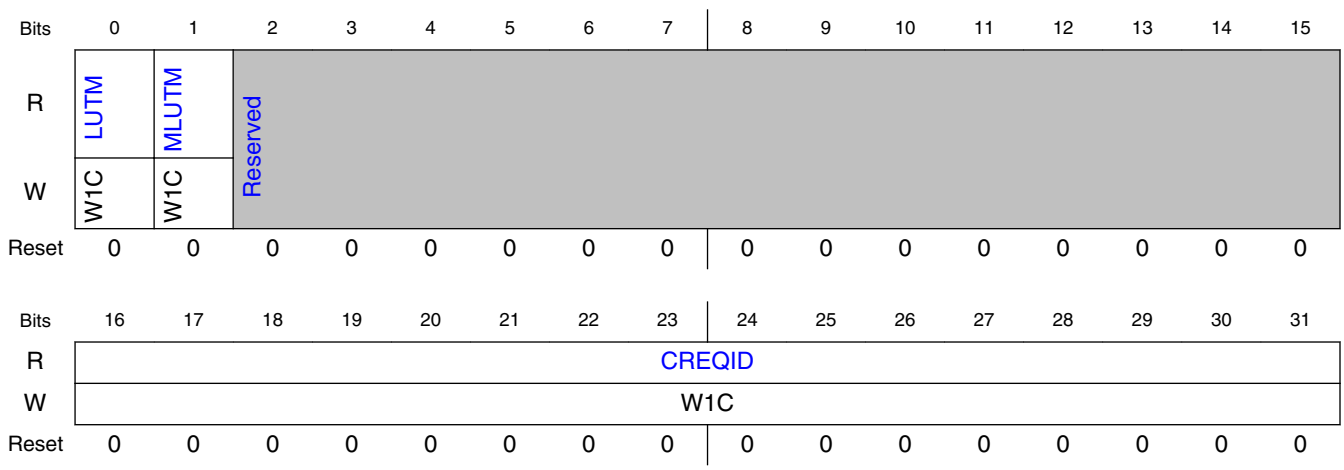
25.5.1.2.1 Offset

Register	Offset
PEXLSR	20h

25.5.1.2.2 Function

The PEX LUT Status Register (PEXLSR) logs the status of a lookup request, through the PEXLSR[LUTM] field. When no match is detected for all the lookup table entries, the lookup table miss LUTM field will be set. PEXLSR[LUTM] can be cleared by writing a 1 to it.

25.5.1.2.3 Diagram



25.5.1.2.4 Fields

Field	Function
0 LUTM	LUTM: Lookup table miss. This field is set when the AXI REQID does not match any of the lookup table entries. This field remains set from the time of an LUT miss until cleared by writing 1 to it.
1 MLUTM	MLUTM: Multiple lookup table miss. This field is set when more than one lookup failed. This field is set from the time of the second LUT miss until cleared by writing 1 to it. This bit is also set if a single miss occurs on both write and read channels simultaneously.
2-15 —	Reserved
16-31 CREQID	CREQID: Captured REQID. This field is the request ID of the first lookup table miss. If a simultaneous LUT miss is detected on both write and read channels, capture priority is given to the write channel REQID and multiple miss is reported in the MLUTM bit. This field remains unchanged from the time of an LUT miss until cleared by writing all 1s to it.

25.5.1.3 PEX LUT Control Register (PEXLCR)

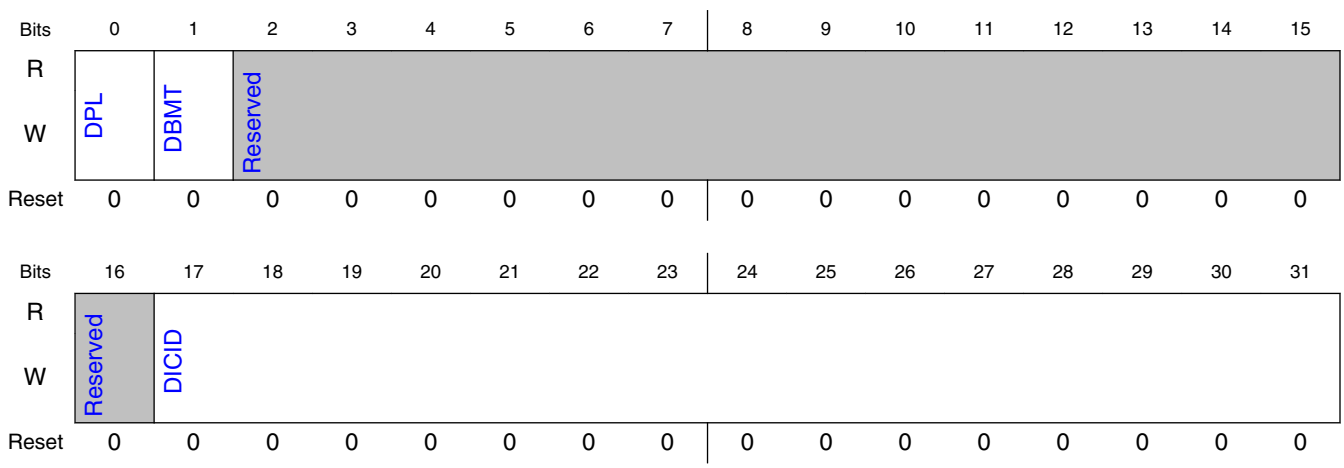
25.5.1.3.1 Offset

Register	Offset
PEXLCR	24h

25.5.1.3.2 Function

The PEX LUT Control Register (PEXLCR) represents the Address Management Qualifier (AMQ) attributes, PL, BMT, 14'h0, ICID[14:0], assigned to the REQID translation when there is a lookup table miss.

25.5.1.3.3 Diagram



25.5.1.3.4 Fields

Field	Function
0 DPL	DPL: Default privilege Level. This field is the default AMQ privilege level attribute for a lookup table miss.
1 DBMT	DBMT: Default bypass memory translation. This field is the default AMQ bypass memory translation attribute for a lookup table miss.
2-16 —	Reserved

Table continues on the next page...

Field	Function
17-31 DICID	DICID: Default isolation context ID. This field is the default AMQ isolation context ID attribute for a lookup table miss

25.5.1.4 PEX LUT Entry a Upper Data Register (PEXL0UDR)

25.5.1.4.1 Offset

Register	Offset
PEXL0UDR	800h

25.5.1.4.2 Function

The PEX LUT Entry *a* Upper Data Register (PEXL*a*UDR) represents the upper data fields of entry *a* within the PEX lookup table.

25.5.1.4.3 Diagram

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

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

25.5.1.4.4 Fields

Field	Function
0-15 REQID	REQID: The Request ID of entry *a* of the PEX lookup table. This field is qualified by the MASK field PEXL*a*UDR[MASK] of the same PEX lookup table entry when attempting to match.
16-31 MASK	MASK: The mask field of entry *a* of the PEX lookup table. This field is qualified with the REQID field of the same entry when attempting to match. It is used to bitwise disable matches against PEXL*a*UDR[REQID] and a received packet's REQID. Note: The reset value is 0xFFFF for entry 0 and 0x0000 for all other entries of the PEX lookup table.

25.5.1.5 PEX LUT Entry a Lower Data Register (PEXL0LDR)

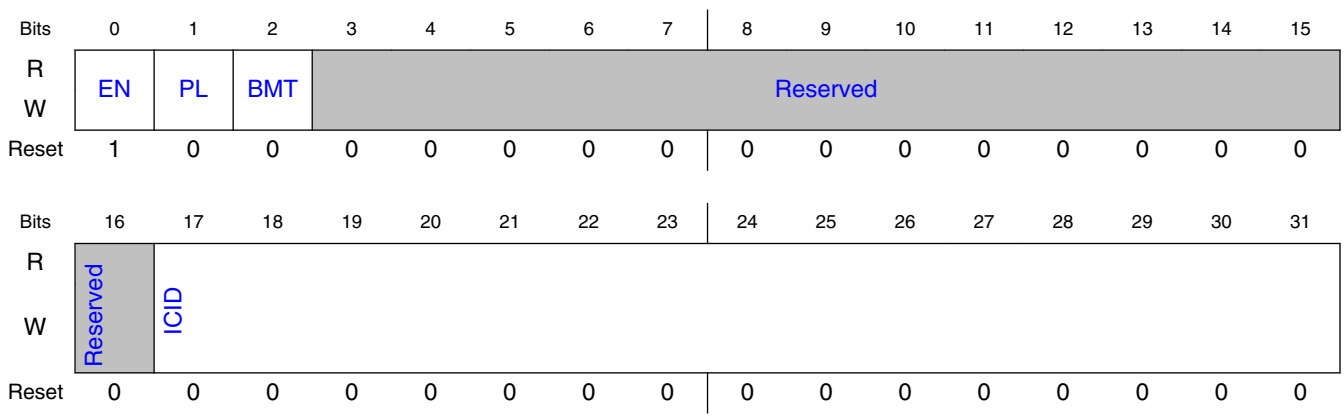
25.5.1.5.1 Offset

Register	Offset
PEXL0LDR	804h

25.5.1.5.2 Function

The PEX LUT Entry *a* Lower Data Register (PEXL*a*LDR) represents the lower data fields of entry *a* within the PEX lookup table.

25.5.1.5.3 Diagram



25.5.1.5.4 Fields

Field	Function
0 EN	Enable: The enable field of entry *a* within the PEX lookup table. This field indicates whether the entry participates in matching. Note: The reset value is 0x1 for entry 0 and 0x0 for all other entries of the PEX lookup table. 0b - This entry does not participate in matching. 1b - This entry participates in matching.
1 PL	Priviledge Level: The priviledge level field of entry *a* within the PEX lookup table. If this entry matches, this field will be added to the outgoing AXI request sideband.
2	Bypass memory translation: The bypass memory translation of entry *a* within the PEX lookup table. If this entry matches, this field will be added to the outgoing AXI request sideband.

Table continues on the next page...

Field	Function
BMT	
3-16 —	Reserved
17-31 ICID	Isolation context ID: The isolation context ID field of entry *a* within the PEX lookup table. If this entry matches, this field will be added to the outgoing AXI request sideband.

25.5.1.6 PEX LUT Entry a Upper Data Register (PEXL1UDR - PEXL 31UDR)

25.5.1.6.1 Offset

For a = 1 to 31:

Register	Offset
PEXLaUDR	800h + (a × 8h)

25.5.1.6.2 Function

The PEX LUT Entry *a* Upper Data Register (PEXL*a*UDR) represents the upper data fields of entry *a* within the PEX lookup table.

25.5.1.6.3 Diagram

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

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

25.5.1.6.4 Fields

Field	Function
0-15 REQID	REQID: The Request ID of entry *a* of the PEX lookup table. This field is qualified by the MASK field PEXL*a*UDR[MASK] of the same PEX lookup table entry when attempting to match.
16-31 MASK	MASK: The mask field of entry *a* of the PEX lookup table. This field is qualified with the REQID field of the same entry when attempting to match. It is used to bitwise disable matches against PEXL*a*UDR[REQID] and a received packet's REQID. Note: The reset value is 0xFFFF for entry 0 and 0x0000 for all other entries of the PEX lookup table.

25.5.1.7 PEX LUT Entry a Lower Data Register (PEXL1LDR - PEXL 31LDR)

25.5.1.7.1 Offset

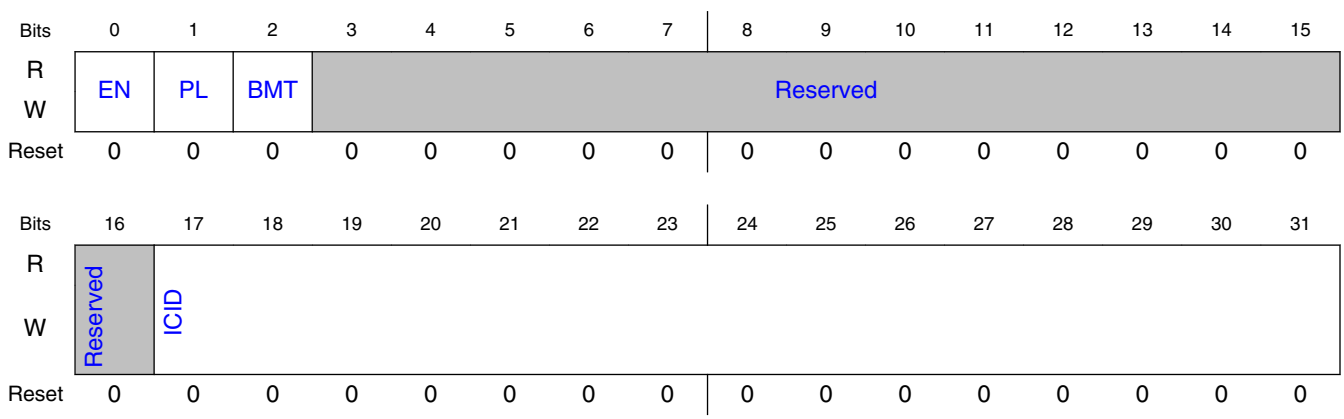
For a = 1 to 31:

Register	Offset
PEXLaLDR	804h + (a × 8h)

25.5.1.7.2 Function

The PEX LUT Entry *a* Lower Data Register (PEXL*a*LDR) represents the lower data fields of entry *a* within the PEX lookup table.

25.5.1.7.3 Diagram



25.5.1.7.4 Fields

Field	Function
0 EN	Enable: The enable field of entry *a* within the PEX lookup table. This field indicates whether the entry participates in matching. Note: The reset value is 0x1 for entry 0 and 0x0 for all other entries of the PEX lookup table. 0b - This entry does not participate in matching. 1b - This entry participates in matching.
1 PL	Priviledge Level: The priviledge level field of entry *a* within the PEX lookup table. If this entry matches, this field will be added to the outgoing AXI request sideband.
2 BMT	Bypass memory translation: The bypass memory translation of entry *a* within the PEX lookup table. If this entry matches, this field will be added to the outgoing AXI request sideband.
3-16 —	Reserved
17-31 ICID	Isolation context ID: The isolation context ID field of entry *a* within the PEX lookup table. If this entry matches, this field will be added to the outgoing AXI request sideband.

25.5.1.8 PEX PFa Config Register (PEX_PF0_CONFIG)

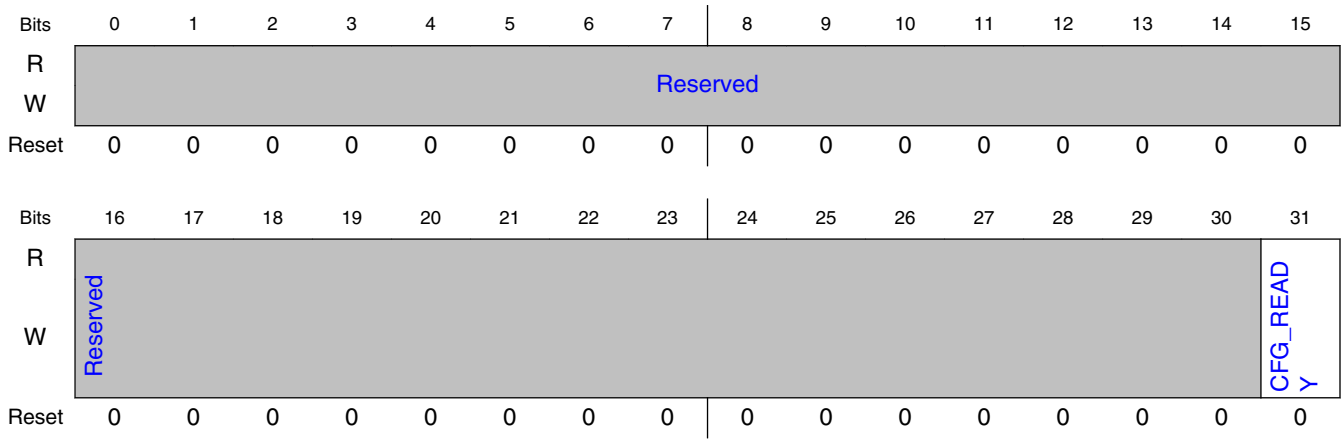
25.5.1.8.1 Offset

Register	Offset
PEX_PF0_CONFIG	4_0014h

25.5.1.8.2 Function

The PEX PF*a* Config register is used for setting the config ready control of the PCIE device.

25.5.1.8.3 Diagram



25.5.1.8.4 Fields

Field	Function
0-30 —	Reserved
31 CFG_READY	Config Ready: This bit is cleared by default to enable EP application software configure the PCIe controller. Any config attempts from the Root Complex will be returned with config retry status (CRS). After application software finishes configuring the PCIe controller it sets the CFG_READY bit to enable accesses from the Root Complex.

25.5.1.9 PEX PFa Interrupt status Register (PEX_PF0_INT_STAT)

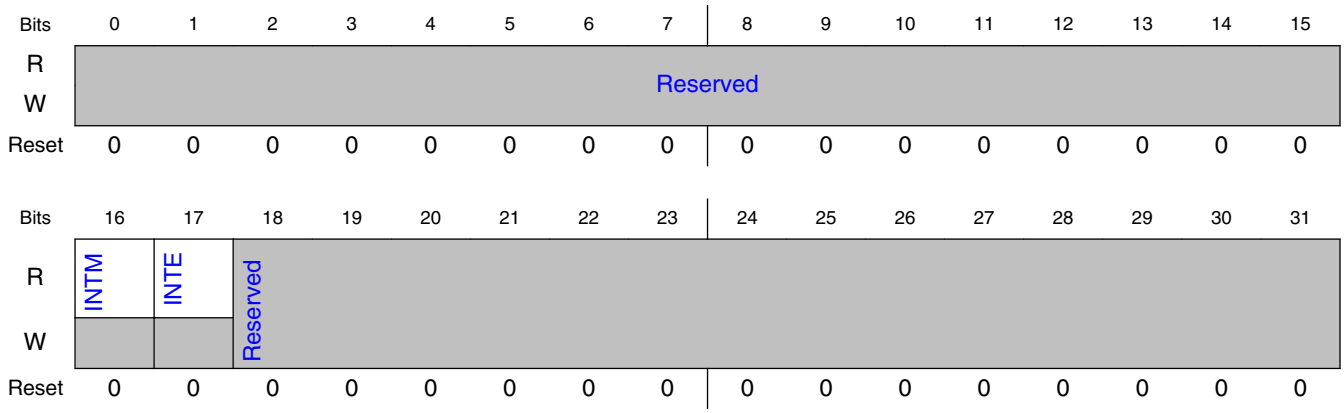
25.5.1.9.1 Offset

Register	Offset
PEX_PF0_INT_STAT	4_0018h

25.5.1.9.2 Function

The PEX PF*a* Interrupt status register is used to indicate that an interrupt is pending.

25.5.1.9.3 Diagram



25.5.1.9.4 Fields

Field	Function
0-15 —	Reserved
16 INTM	Per PF dependent message interrupt is pending.
17 INTE	Per PF dependent error interrupt is pending.
18-31 —	Reserved

25.5.1.10 PEX PFa PCIE pme and message detect register (PEX_PF0_PME_MES_DR)

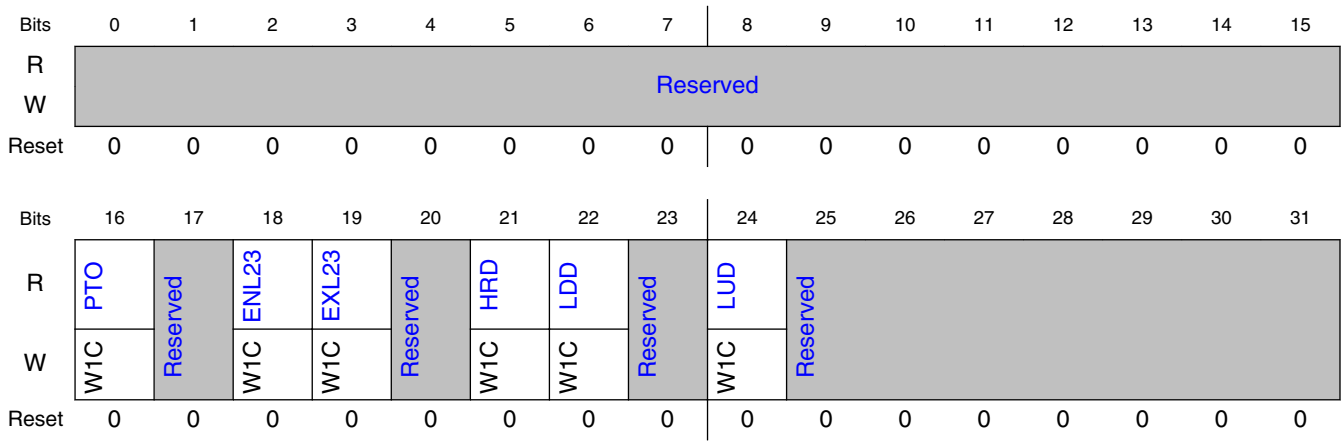
25.5.1.10.1 Offset

Register	Offset
PEX_PF0_PME_MES_DR	4_0020h

25.5.1.10.2 Function

The PEX PF*a* PCIE pme and message detect register is used to detect various power management and error events.

25.5.1.10.3 Diagram



25.5.1.10.4 Fields

Field	Function
0-15 —	Reserved
16 PTO	PTO: Indicates that PME turn off was detected. 1'b1: PME turn off was detected. 1'b0: PME turn off was not detected.
17 —	Reserved
18 ENL23	ENL23: Indicates that PCIe core entered L2/L3 ready state. 1'b1: Entered L2/3 ready state was detected. 1'b0: Entered L2/3 ready state was not detected.
19 EXL23	EXL23: Indicates that PCIe core Exited L2/L3 ready state. 1'b1: Exit L2/3 ready state was detected. 1'b0: Exit L2/3 ready state was not detected.
20 —	Reserved
21 HRD	HRD: Indicates a hot reset was detected. 1'b1: hot reset was detected. 1'b0: hot reset was not detected.
22 LDD	LDD : Indicates a link down was detected. 1'b1: link down was detected. 1'b0: link down was not detected.
23 —	Reserved
24 LUD	LUD : Indicates a link up was detected. 1'b1: link up was detected. 1'b0: link up was not detected.
25-31 —	Reserved

25.5.1.11 PEX PFa PCIE pme and message disable register (PEX_PF0_PME_MES_DISR)

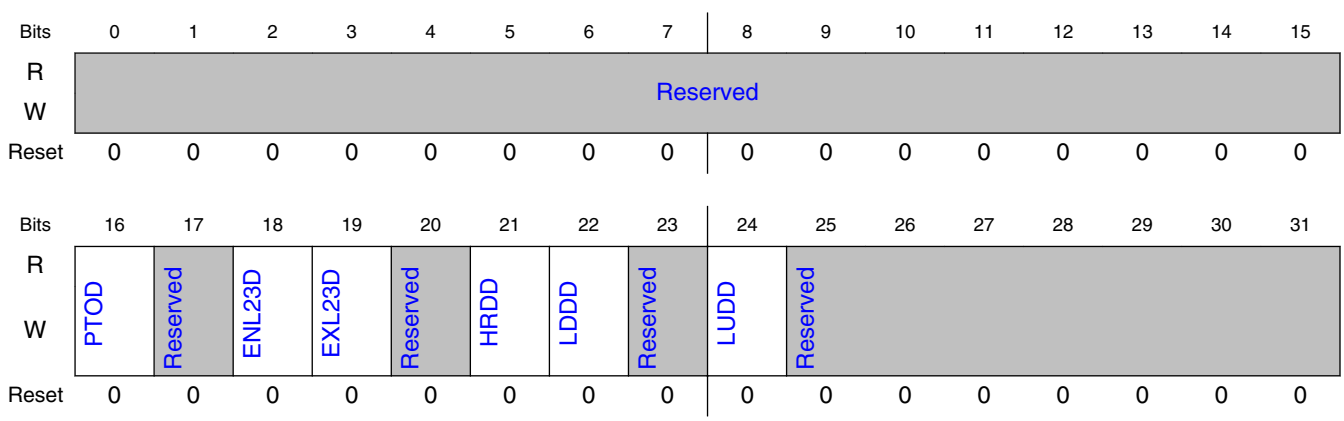
25.5.1.11.1 Offset

Register	Offset
PEX_PF0_PME_MES_DISR	4_0024h

25.5.1.11.2 Function

The PEX PF*a* PCIE pme and message disable register is used to disable the detection of various power management and error events.

25.5.1.11.3 Diagram



25.5.1.11.4 Fields

Field	Function
0-15 —	Reserved
16 PTOD	PTO: PME turn off detect disabled. 1'b1: PME turn off detect disabled. 1'b0: PME turn off detect enabled.
17 —	Reserved

Table continues on the next page...

Field	Function
18 ENL23D	ENL23: Entered L2/L3 ready state detect disabled. 1'b1: Entered L2/L3 ready state detect disabled. 1'b0: Entered L2/L3 ready state detect enabled.
19 EXL23D	EXL23: Exited L2/L3 ready state detect disable. 1'b1: Exit L2/L3 ready state detect disabled. 1'b0: Exit L2/L3 ready state detect enabled.
20 —	Reserved
21 HRDD	HRDD: Hot reset detect disable. 1'b1: hot reset detect disabled. 1'b0: hot reset detect enabled.
22 LDDD	LDDD : Link down detect disable. 1'b1: link down detect disabled. 1'b0: link down detect enabled.
23 —	Reserved
24 LUDD	LUDD : link up detect disable. 1'b1: link up detect disabled. 1'b0: link up detect enabled.
25-31 —	Reserved

25.5.1.12 PEX PFa PCIE pme and message interrupt enable register (PEX_PF0_PME_MES_IER)

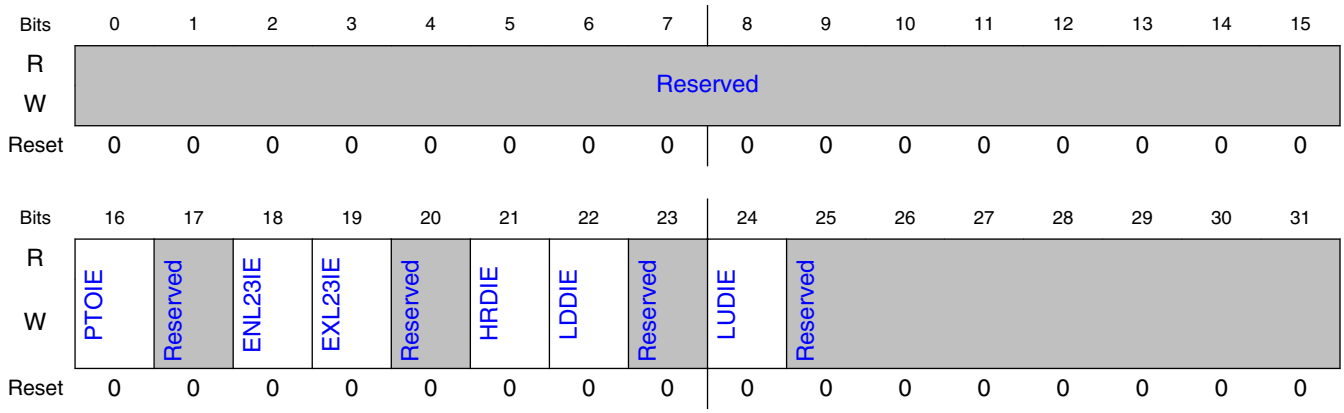
25.5.1.12.1 Offset

Register	Offset
PEX_PF0_PME_MES_IER	4_0028h

25.5.1.12.2 Function

The PEX PF*a* PCIE pme and message disable register is used to enable interrupt generation for the detection of various power management and error events.

25.5.1.12.3 Diagram



25.5.1.12.4 Fields

Field	Function
0-15 —	Reserved
16 PTOIE	PTO: PME turn off detect interrupt enable. 1'b1: PME turn off detect interrupt enabled. 1'b0: PME turn off detect interrupt disabled.
17 —	Reserved
18 ENL23IE	ENL23IE: Entered L2/L3 ready state detect interrupt enabled. 1'b1: Entered L2/L3 ready state detect interrupt enabled. 1'b0: Entered L2/L3 ready state detect interrupt disabled.
19 EXL23IE	EXL23IE: Exited L2/L3 ready state detect interrupt enable. 1'b1: Exit L2/L3 ready state detect interrupt enabled. 1'b0: Exit L2/L3 ready state detect interrupt disabled.
20 —	Reserved
21 HRDIE	HRDD: Hot reset detect interrupt enable. 1'b1: hot reset detect interrupt enabled. 1'b0: hot reset detect interrupt disabled.
22 LDDIE	LDDIE : Link down detect interrupt enable. 1'b1: link down detect interrupt enabled. 1'b0: link down detect interrupt disabled.
23 —	Reserved
24 LUDIE	LUDIE : link up detect interrupt enable. 1'b1: link up detect interrupt enabled. 1'b0: link up detect interrupt disabled.
25-31 —	Reserved

25.5.1.13 PEX PFa PCIE message command register (PEX_PF0_MCR)

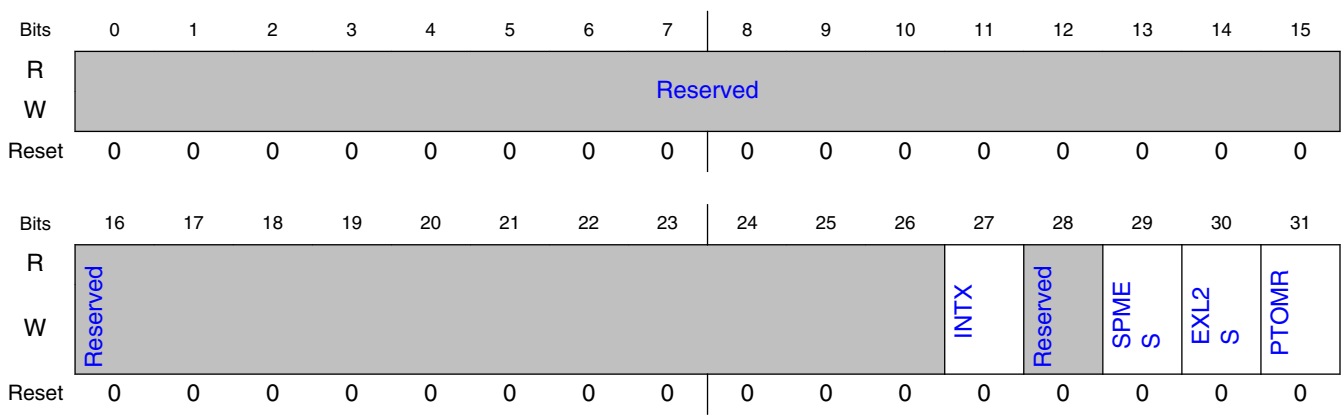
25.5.1.13.1 Offset

Register	Offset
PEX_PF0_MCR	4_002Ch

25.5.1.13.2 Function

The PEX PF*a* PCIE message command register enables the generation of interrupt and power management messages per PF.

25.5.1.13.3 Diagram



25.5.1.13.4 Fields

Field	Function
0-26 —	Reserved
27 INTX	INTX: Assert/de-assert intx command. When sys_int goes from low to high, the core generates an Assert_INTx Message. When sys_int goes from high to low, the core generates a Deassert_INTx Message.The Interrupt Pin register for the corresponding function determines which INTx Message the core generates (INTA).
28 —	Reserved
29 SPMES	SPME: Send PM_PME command. When set to 1 , a PM PME command is generated to wake up the the PCIE power management controller from a D1, D2 or D3 power state. The bit is self clearing. Once the bit is set , S/W needs to wait for the bit to self clear before sending a new command..

Table continues on the next page...

Field	Function
30 EXL2S	EXL2S : Exit L2 state command. When set to 1 , an L2 exit command is generated. The bit is self clearing. Once the bit is set , S/W needs to wait for the bit to self clear before sending a new command.
31 PTOMR	PTOMR : Generate PME turn off message (RC only). When set to 1, a pme turn off message is generated from the RC. The bit is self clearing. Once the bit is set , S/W needs to wait for the bit to self clear before sending a new command.

25.5.1.14 PEX PFa Route By Port Address Upper register (PEX_PF0_RBP_ADDR_U)

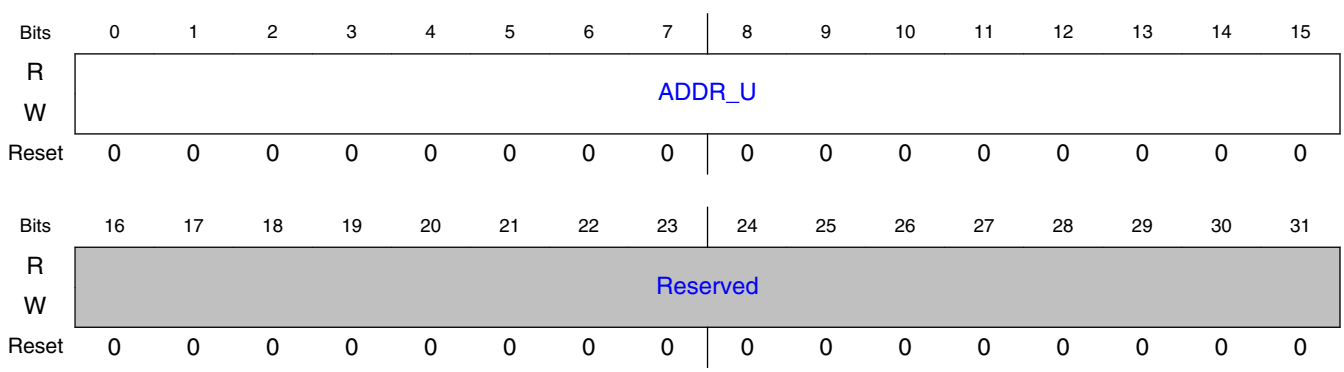
25.5.1.14.1 Offset

Register	Offset
PEX_PF0_RBP_ADDR_U	4_0140h

25.5.1.14.2 Function

The PEX PF*a* Route By Port Address Upper register provides the upper 16 bit out of the 64 bit address of the PCIE slave device used for the outbound transactions.

25.5.1.14.3 Diagram



25.5.1.14.4 Fields

Field	Function
0-15	ADDR_U : addr[63:48] . upper 16 bit of the PCIE slave device used for the outbound transactions.

Table continues on the next page...

Field	Function
ADDR_U	
16-31 —	Reserved

25.5.1.15 PEX PFa PCIE error detect register (PEX_PF0_ERR_DR)

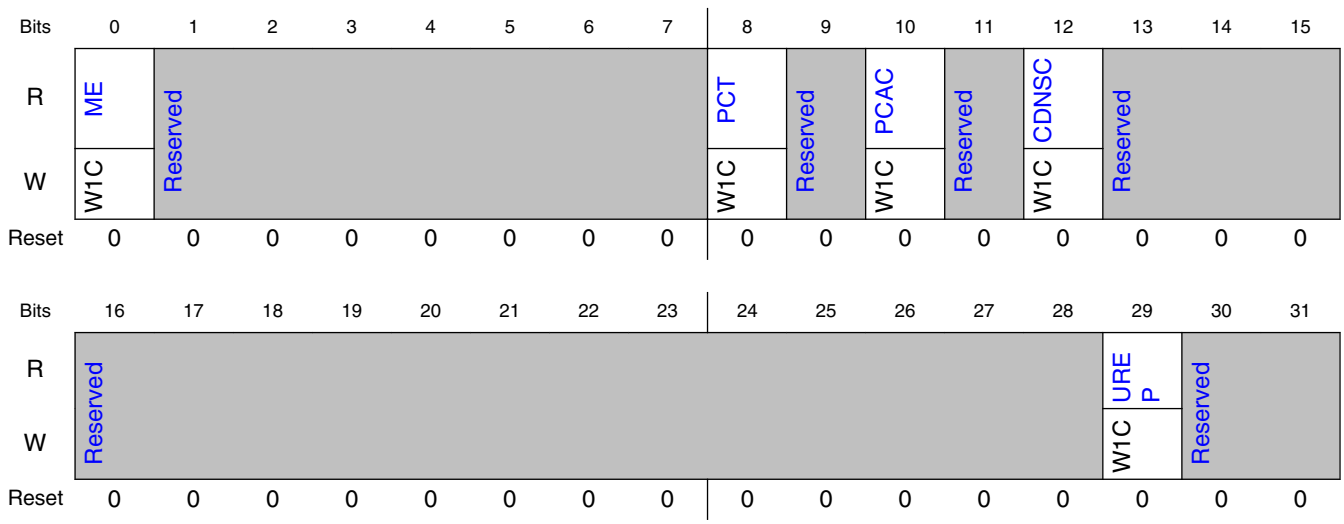
25.5.1.15.1 Offset

Register	Offset
PEX_PF0_ERR_DR	4_0200h

25.5.1.15.2 Function

The PEX PF*a* error detect register is used for capturing various error events .

25.5.1.15.3 Diagram



25.5.1.15.4 Fields

Field	Function
0 ME	ME: Indicates Multiple errors of same type. If any of the detectable errors in PEX_ERR_DET register is detected more than one time the ME bit will be set. 1'b1: Multiple errors of same type was detected. 1'b0: Multiple errors of same type was not detected.
1-7 —	Reserved
8 PCT	PCT: Indicates completion timeout. 1'b1: Completion timeout was detected. 1'b0: Completion timeout was not detected.
9 —	Reserved
10 PCAC	PCAC: Completer abort was detected. 1'b1: Completer abort was detected. 1'b0: Completer abort was not detected.
11 —	Reserved
12 CDNSC	CDNSC : Completion with data not succsessful was detected. 1'b1: Completion with data not succsessful was detected. 1'b0: Completion with data not succsessful was not detected.
13-28 —	Reserved
29 UREP	UREP : Indicates an unsupported request completion was detected. 1'b1: Unsupported request in EP mode was detected. 1'b0: Unsupported request in EP mode not detected.
30-31 —	Reserved

25.5.1.16 PEX PFa PCIE error interrupt enable register (PEX_PF0_ERR_EN)

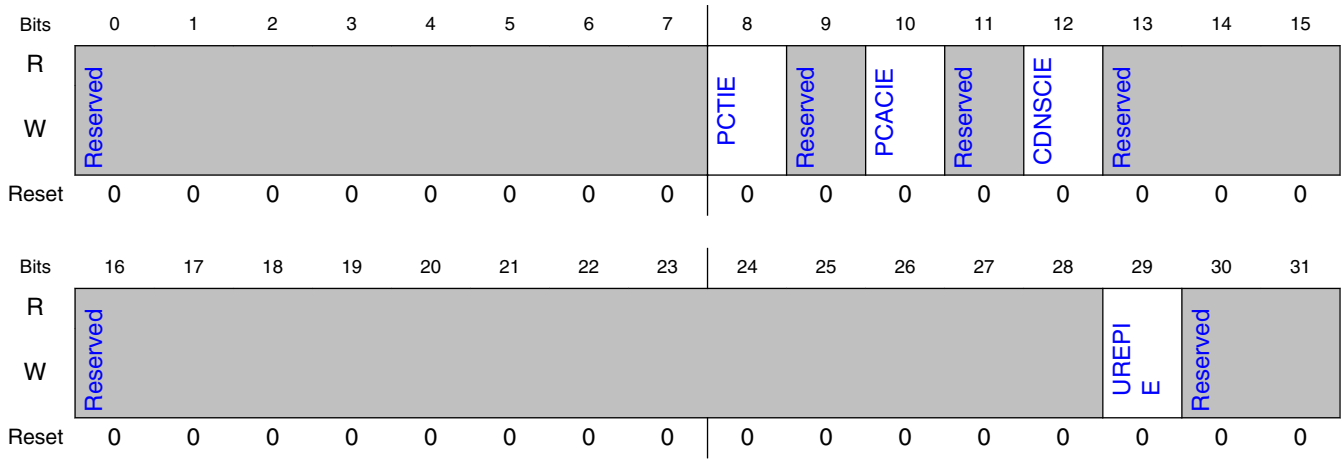
25.5.1.16.1 Offset

Register	Offset
PEX_PF0_ERR_EN	4_0208h

25.5.1.16.2 Function

The PEX PF*a* error interrupt enable register. Enables/disables interrupt for errors detected in PEX_ERR_DET.

25.5.1.16.3 Diagram



25.5.1.16.4 Fields

Field	Function
0-7 —	Reserved
8 PCTIE	PCTIE: completion timeout interrupt enable. 1'b1: Completion timeout interrupt enabled. 1'b0: Completion timeout interrupt disabled.
9 —	Reserved
10 PCACIE	PCACIE: Completer abort interrupt enable. 1'b1: Completer abort interrupt enabled. 1'b0: Completer abort interrupt disabled.
11 —	Reserved
12 CDNSCIE	CDNSCIE : Completion with data not succsessful interrupt enable. 1'b1: Completion with data not succsessful interrupt enabled. 1'b0: Completion with data not succsessful was not detected.
13-28 —	Reserved
29 UREPIE	UREPIE : Unsupported request in EP mode interrupt enable. 1'b1: Unsupported request in EP mode interrupt enabled. 1'b0: Unsupported request in EP mode interrupt disabled.
30-31 —	Reserved

25.5.1.17 PEX PFa PCIE error disable register (PEX_PF0_ERR_DISR)

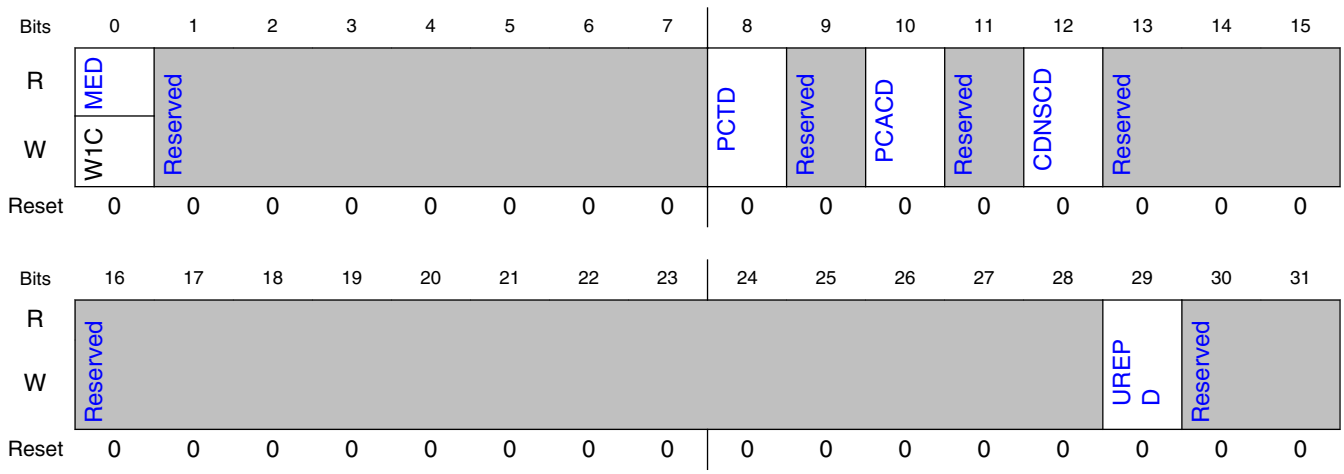
25.5.1.17.1 Offset

Register	Offset
PEX_PF0_ERR_DISR	4_0210h

25.5.1.17.2 Function

The PEX PF*a* error disable register. disables detection of errors in PEX_ERR_DET.

25.5.1.17.3 Diagram



25.5.1.17.4 Fields

Field	Function
0 MED	ME: Multiple errors of same type detection disable. 1'b1: Multiple errors of same type detection disabled. 1'b0: Multiple errors of same type detection enabled.
1-7 —	Reserved
8 PCTD	PCTIE: completion detection disable. 1'b1: Completion timeout detection disabled. 1'b0: Completion timeout detection enabled.
9 —	Reserved
10 PCACD	PCACD: Completer abort detection disable. 1'b1: Completer abort detection disabled. 1'b0: Completer abort detection enabled.
11 —	Reserved

Table continues on the next page...

Field	Function
12 CDNSCD	CDNSCD : Completion with data not succsessful detection disable. 1'b1: Completion with data not succsessful detection disabled. 1'b0: Completion with data not succsessful detection enabled.
13-28 —	Reserved
29 UREPD	UREPD : Unsupported request in EP mode detection disable. 1'b1: Unsupported request in EP mode detection disabled. 1'b0: Unsupported request in EP mode detection enabled.
30-31 —	Reserved

25.5.1.18 PEX PF0 Debug register (PEX_PF0_DBG)

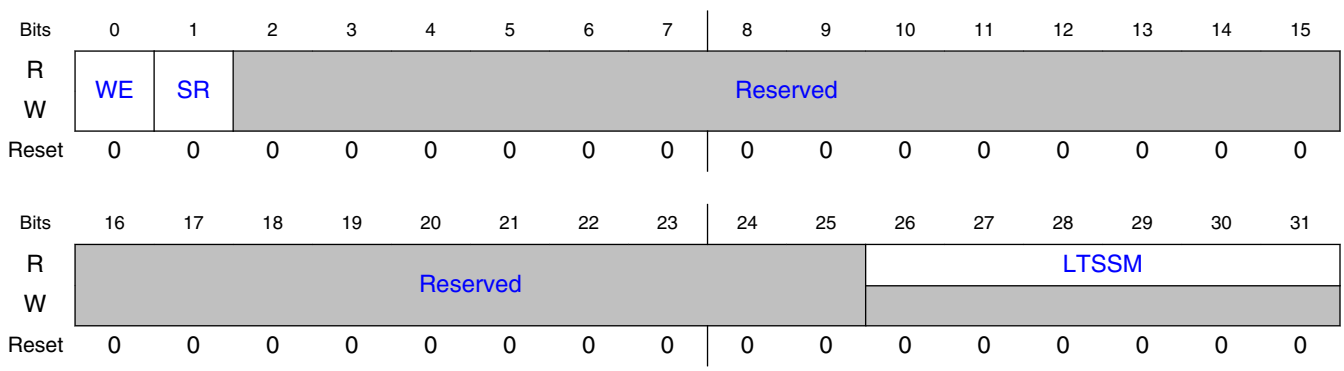
25.5.1.18.1 Offset

Register	Offset
PEX_PF0_DBG	4_07FCh

25.5.1.18.2 Function

By setting the WE field, the user has access to the SR field. Once set, SR will enable soft reset to the PEX module. LTSSM is a read only field that reports the LTSSM status from the PEX module.

25.5.1.18.3 Diagram



25.5.1.18.4 Fields

Field	Function
0 WE	Write Enable This bit when set allows the user to have write access to the PEXDBG[SR] for both setting and clearing the soft reset on the PEX module.
1 SR	Soft Reset When SR is set to 1, the PEX module enters soft reset. The PEX mode remains in soft reset while SR is set to 1. When SR is cleared, the PEX module exits soft reset. If SR is to be set and cleared multiple times back to back, software must wait a few seconds before changing the bit value to allow each new state to take effect.
2-25 —	Reserved
26-31 LTSSM	Link Training Status State Machine (LTSSM) status This field is read only and captures the PEX block LTSSM state at each clock cycle. These bits indicate the link training status. This field is useful for debugging link training failures. **NOTE:** The status code changes while reacting to the link status. Therefore, software may need to read LTSSM multiple times to ensure the value has stabilized. 000000b - DETECT_QUIET 000001b - DETECT_ACTIVE 000010b - POLL_ACTIVE 000011b - POLL_COMPLIANCE 000100b - POLL_CONFIG 000101b - PRE_DETECT_QUIET 000110b - DETECT_WAIT 000111b - CFG_LINKWD_START 001000b - CFG_LINKWD_ACEPT 001001b - CFG_LANENUM_WAIT 001010b - CFG_LANENUM_ACEPT 001011b - CFG_COMPLETE 001100b - CFG_IDLE 001101b - RCVRY_LOCK 001110b - RCVRY_SPEED 001111b - RCVRY_RCVRCFG 010000b - RCVRY_IDLE 010001b - L0 010010b - L0S 010011b - L123_SEND_EIDLE 010100b - L1_IDLE 010101b - L2_IDLE 010110b - L2_WAKE 010111b - DISABLED_ENTRY 011000b - DISABLED_IDLE 011001b - DISABLED 011010b - LPBK_ENTRY 011011b - LPBK_ACTIVE 011100b - LPBK_EXIT 011101b - LPBK_EXIT_TIMEOUT 011110b - HOT_RESET_ENTRY 011111b - HOT_RESET 100000b - RCVRY_EQ0 100001b - RCVRY_EQ1 100010b - RCVRY_EQ2 100011b - RCVRY_EQ3

25.6 Functional Description

The PCI Express protocol relies on a requestor/completer relationship where one device requests that some desired action be performed by some target device and the target device completes the task and responds. Usually the requests and responses occur through a network of links, but to the requester and to the completer, the intermediate components are transparent.

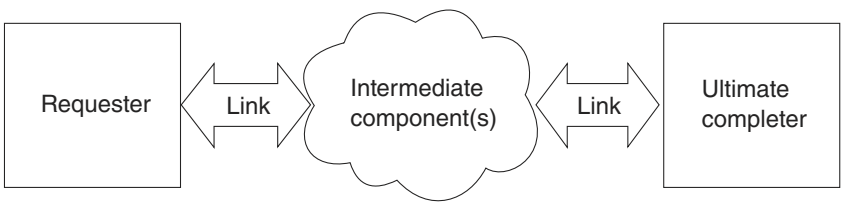


Figure 25-3. Requester/Completer Relationship

Each PCI Express device is divided into two halves—transmit (TX) and receive (RX), and each of these halves is further divided into three layers—transaction, data link, and physical, as shown in the following figure.

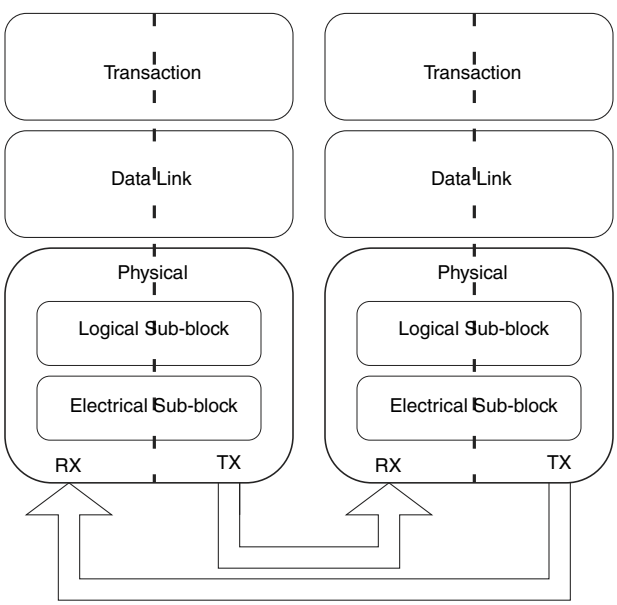


Figure 25-4. PCI Express High-Level Layering

Packets are formed in the transaction layer (TLPs) and data link layer (DLLPs), and each subsequent layer adds the necessary encodings and framing, as shown in the following figure. As packets are received, they are decoded and processed by the same layers but in reverse order, so they may be processed by the layer or by the device application software.

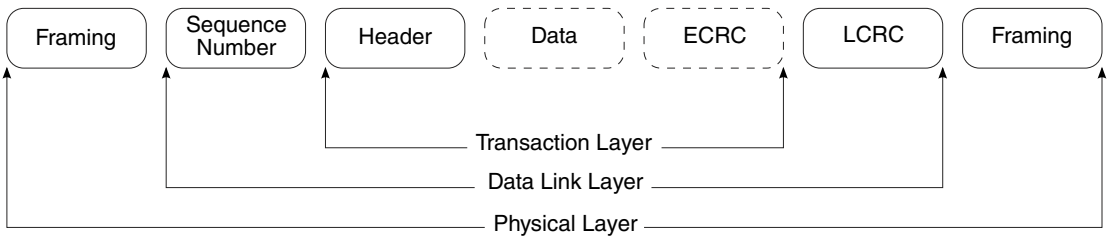


Figure 25-5. PCI Express Packet Flow

25.6.1 Architecture

This section describes implementation details of the PCI Express controller.

25.6.1.1 PCI Express Transactions

The following table contains the list of transactions that the PCI Express controller supports as an initiator and a target.

Table 25-4. PCI Express Transactions

PCI Express Transaction	Supported as an Initiator	Supported as a Target	Definition
Mrd	Yes	Yes	Memory Read Request
MRdLk	No	No	Memory Read Lock. As a target, CplLk with UR status is returned.
MWr	Yes	Yes	Memory Write Request to memory-mapped PCI-Express space
IORd	Yes (RC only)	No	I/O Read request. As a target, Cpl with UR status is returned.
IOWr	Yes (RC only)	No	I/O Write Request. As a target, Cpl with UR status is returned.
CfgRd0	Yes (RC only)	Yes (EP only)	Configuration Read Type 0.
CfgWr0	Yes (RC only)	Yes (EP only)	Configuration Write Type 0.
CfgRd1	Yes (RC only)	No	Configuration Read Type 1. As a target, Cpl with UR status is returned.
CfgWr1	Yes (RC only)	No	Configuration Write Type 1. As a target, Cpl with UR status is returned.
Msg	Yes	Yes	Message Request. Message without data is not forwarded to memory.
MsgD	Yes (RC only)	Yes (EP only)	Message Request with Data payload.
Cpl	Yes	Yes	Completion without Data

Table continues on the next page...

Table 25-4. PCI Express Transactions (continued)

PCI Express Transaction	Supported as an Initiator	Supported as a Target	Definition
CplD	Yes	Yes	Completion with Data
CplLk	No	Yes	Completion for Locked Memory Read without Data. The only time that CplLk is returned with UR status is when the controller receives a MRdLk command.
CplDLk	No	No	Completion for Locked Memory Read with Data

25.6.1.2 Transaction ordering rules

In general, transactions are serviced in the order that they are received.

However, transactions can be reordered as they are sent due to a stalled condition such as a full internal buffer. The following table describes the transaction ordering rules for this device.

Table 25-5. PCI Express controller TLP transaction ordering rules

Row pass column?		Posted request	Non-posted request		Completion	
		Memory write or message request (col 2)	Read request (Col 3)	I/O or Configuration write request (col 4)	Read completion (Col 5)	I/O or configuration write completion (col 6)
Posted request	Memory write or message request (row A)	a) No[1] b) No[1]	Yes	Yes	a) Yes[2] b) N/A[3]	a) Yes[2] b) N/A[3]
Non-posted request	Read request (row B)	No	No	No	a) No[4] b) Yes[5]	a) No[4] b)Yes[5]
	I/O or configuration write request (Row C)	No	No	No	a) No[4] b) Yes[5]	a) No[4] b) Yes[5]
Completion	Read completion (row D)	a) No[6] b) Yes[7]	Yes	Yes	a) No[8] b) No[8]	No
	I/O or configuration write completion (row E)	a) No[6] b) Yes[7]	Yes	Yes	No	No

1. Regardless of the setting of the relaxed ordering (RO) bit, a posted request cannot bypass another posted request.
2. Regardless of the setting of the relaxed ordering bit, a posted request can always bypass a completion.
3. N/A indicates that the original rules at these entries defined by the *PCI Express Base Specification* do not apply to RC or EP.
4. A non-posted request cannot bypass a completion if the relaxed ordering bit is cleared (that is, RO = 0).

5. A non-posted request can bypass a completion if the relaxed ordering bit is set (that is, RO = 1).
6. A read completion, I/O write completion, or configuration write completion cannot bypass a posted request if the relaxed ordering bit is cleared (that is, RO = 0).
7. A read completion, I/O write completion, or configuration write completion can bypass a posted request if the relaxed ordering bit is set (that is, RO = 1).
8. Regardless of the setting of the relaxed ordering bit, a read completion cannot bypass another read completion.

In general, the following points summarize the ordering rules for sending the next outstanding request:

- A posted request can bypass all other transactions except another posted request.
- A non-posted request cannot bypass posted or other non-posted requests, but it can bypass a completion if the relaxed ordering (RO) bit is set.(See Table 25-5).
- A completion can bypass posted requests if the relaxed ordering (RO) bit is set and can bypass non-posted transactions. However, a completion cannot bypass other completions.

25.6.1.3 Internal Address Translation Unit

The core uses the iATU to replace the TLP address and TLP header fields in the current TLP request header. This section presents the following topics:

- iATU Overview
- Programming the iATU
- Outbound iATU Operation
- Inbound iATU Operation

25.6.1.3.1 iATU Overview

Address translation is used for mapping different address ranges to different memory spaces supported by your application. A typical example maps the internal platform memory space to PCI memory space. The iATU supports type translation. Without address translation, your application address is passed to/from the PCIe TLPs directly through the internal platform . You can program the iATU to implement your own address translation scheme without the need for additional external hardware.

Inbound Features:

- Match mode operation for MEM TLPs. No translation for completions. Selectable BAR Match mode operation for MEM TLPs.

NOTE

Address match mode can only be used in RC mode; BAR match mode must be used in EP mode.

    - TLPs destined for configuration registers in an upstream port are not translated.
    - TLPs that are not error-free (ECRC, malformed and so on) are not translated.
- Programmable TLP header field matching.
    - TYPE/AT
- Function number
- 2 address regions programmable for location and size.
- Programmable enable/disable per region.
- Automatic FMT field translation between three DWORDs and four DWORDs for 64-bit addresses.
- ECAM Configuration Shift mode to allow a 256 MB CFG1 space to be located anywhere in the 64-bit address space.
- Supports regions from 4 kB to 4 GB in size.

Outbound Features:

- Address Match mode operation for MEM and I/O, CFG, and MSG TLPs. No translation for completions.
- Supports type translation through TLP type header field replacement for MEM or I/O types to MSG/CFG types.
    - Includes posted to non-posted translation (for example, MWr to CfgWr0)
    - No translation from completions
- Programmable TLP header field replacement.
    - TYPE/TD/TC/AT/ATTR/MSG-Code
- 2 address regions programmable for location and size.
- Programmable enable/disable per region.
- Automatic FMT field translation between three DWORDs and four DWORDs for 64-bit addresses.
- Configuration Shift mode. Optimizes the memory footprint of CFG accesses destined for the internal platform interface in a multifunction device.
- Response code which defines the completion status to return for accesses matching a region.
- Supports regions from 4 kB to 4 GB in size.

NOTE

The default behavior of the ATU when there is no address match in the outbound direction or no TLP attribute match in the inbound direction, is to pass the transaction through.

25.6.1.3.2 Programming the iATU

You can access the iATU registers through the DBI interface or through PCIe CFG accesses. The following registers are used for programming the iATU.

Table 25-6. iATU Register Map

Byte Offset	Description
0x900	iATU Index Register
0x904	iATU Region Control 1 Register
0x908	iATU Region Control 2 Register
0x90C	iATU Region Lower Base Address Register
0x910	iATU Region Upper Base Address Register
0x914	iATU Region Limit Address Register
0x918	iATU Region Lower Target Address Register
0x91C	iATU Region Upper Target Address Register
0x920	iATU Region Control 3 Register

The iATU registers are programmed through an indirect addressing scheme (using an index register) to reduce the address footprint in the PCI Express extended configuration space. The index 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.

25.6.1.3.3 Outbound iATU Operation

This section describes the processing of outbound requests by the iATU. This section presents the following topics:

- Overview (Address Match Mode)
- RID BDF Number Replacement
- iATU Outbound MSG Handling
- CFG Handling
- CFG Shift Feature
- FMT Translation
- No Address Match Result
- Writing to a MRdLk Region
- Outbound Programming Example

25.6.1.3.3.1 Overview (Address Match Mode)

The address field of each request MEM and I/O TLP is checked to see if it falls into any of the enabled address regions defined by the “Start” and “End” addresses as defined in Figure 25-6.

NOTE

When the “Region Enable” bit of the Region Control 2 register is “0”, then that region is not used for address matching.

When 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 header fields are replaced with the corresponding fields in the “iATU Control 1 Register” (IATU_REGION_CTRL_1_OFF_OUTBOUND_0). When your application internal platform address field matches more than one of the 2 address regions, then the first (lowest of the numbers from 0 to 1) enabled region to be matched is used. For details on what happens when there is no address match. This operational mode (called “Address Match Mode“) is always used for outbound translation.

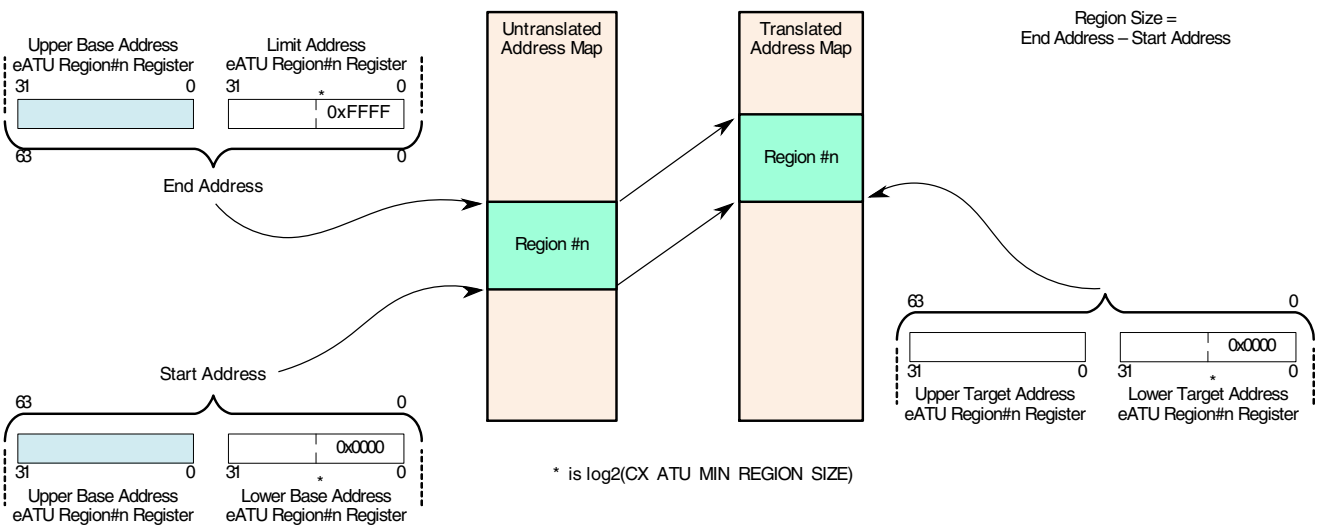


Figure 25-6. iATU Address Region Mapping: Outbound and Inbound (Address Match Mode), 64-bit Address

When the PCIe core is operating with 32-bit addresses, then the operation is defined as in figure below.

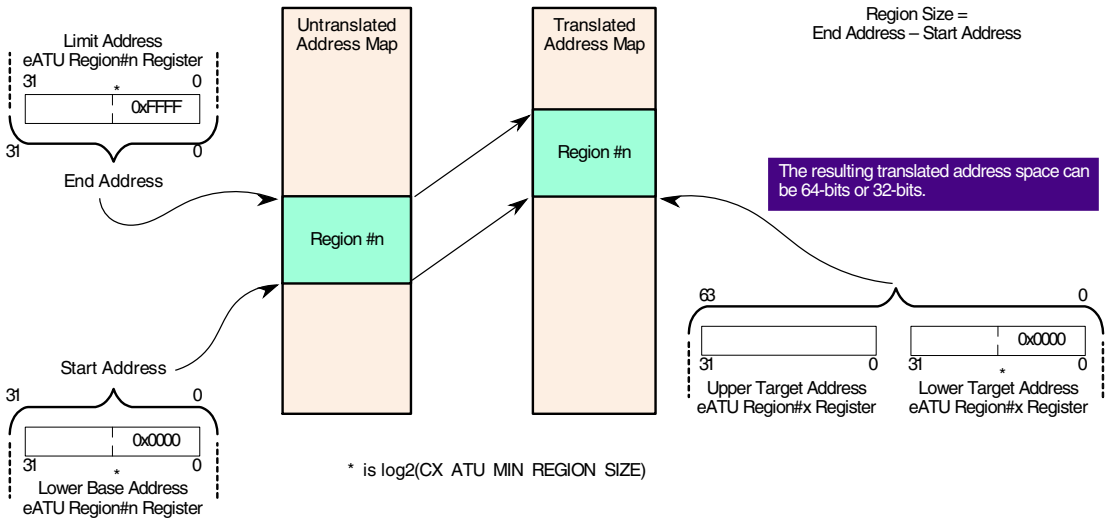


Figure 25-7. iATU Address Region Mapping: Outbound and Inbound (Address Match Mode), 32-bit Address

The upper 32 bits of the Target Address register always forms the upper 32 bits of the translated address because:

- The maximum region size is 4 GB.
- A region must not cross a 4 GB boundary.

In the figure, CX_ATU_MIN_REGION_SIZE specifies the minimum size of an address translation region and is 4 KB.

25.6.1.3.3.2 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 field of the TLP is taken from the 3-bit “Function Number” field of the “iATU Control 1 Register” (IATU_REGION_CTRL_1_OFF_OUTBOUND_0). The value in this field must be 0x0 unless multifunction operation is enabled (the number of PFs is greater than 1).

NOTE

The requester ID uses the 8:5:3 bit PCI Bus.Device.Function (BDF) format.

25.6.1.3.3.3 iATU Outbound MSG Handling

NOTE

The iATU only supports generating outbound messages with data (MsgD); messages without data are not supported. The MWr transactions to be translated must have a non-zero effective length.

The iATU supports TYPE translation/conversion of MEM and I/O TLPs to MsgD TLPs. This supports applications that are unable to directly generate MsgD TLPs. When there is a successful address match on an outbound MEM TLP, and the translated TLP type field is MSG (that is, the type field of the “iATU Control 1 Register” IATU_REGION_CTRL_1_OFF_OUTBOUND_0 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” (IATU_REGION_CTRL_2_OFF_OUTBOUND_0).

25.6.1.3.3.4 CFG Handling

The iATU supports translation of I/O and MEM TLPs to CFG TLPs. The 16-bit Bus.Device.Function (BDF) is derived from bits [31:16] of the “iATU Lower Target Address Register” (IATU_LWR_TARGET_ADDR_OFF_OUTBOUND_0). You can shift/remap the BDF within the PCIe address space using the CFG Shift Feature. Address translation of CFG TLPs is also possible with the iATU.

25.6.1.3.3.5 CFG Shift Feature

This expander feature can be enabled by setting the “CFG Shift” bit of the “iATU Control 2 Register” (IATU_REGION_CTRL_2_OFF_OUTBOUND_0). This shifts/maps the BDF, which is bits [27:12] of the target address, up to bits [31:16] of the translated address. This allows all outgoing I/O and 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 Section 7.2.2 of the PCI Express Base 3.0 Specification, revision 1.0. ECAM supports the mapping (MEM to CFG translation) from memory address space to PCIe configuration space address as defined in Table 25-7. ECAM maps bits [27:12] of the untranslated MEM TLP to become the BDF of the resulting CFG TLP.

Table 25-7. ECAM Mapping

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

25.6.1.3.3.6 FMT Translation

The iATU automatically sets the TLP format field for three DWORDs when it detects all zeroes in the upper 32 bits of the translated address. Otherwise, it sets it to four DWORDs when it detects a 64-bit address (that is, when there is a “1” in the upper 32 bits of the translated address). When the original address and the translated address are of different format, the iATU ensures that the TLP header size matches the translated address format.

25.6.1.3.3.7 No Address Match Result

When there is no address match then the address is untranslated but the TLP header information comes from the relevant fields on the internal platform interface.

25.6.1.3.3.8 Writing to a MRdLk Region

When there is a successful address match for an outbound write and the type header field in the “iATU Control 1 Register” is “ 00001” indicating a locked MEM transfer, then the core sets the type field to “0000” (MEM).

25.6.1.3.3.9 Outbound Programming Example

Define outbound region 1 as a 64 kB I/O region from 0x8000_0000_D000_0000 to 0x8000_0000_D000_FFFF, to be mapped to 0x00010000 in the PCIe I/O space.

1. Setup the Index 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 through the Region Control 1 Register.

Write 0x00000002 to Address {0x700 +0x204} to define the type of the region to be I/O.

5. Enable the region.

   Write 0x80000000 to Address {0x700 +0x208} to enable the region.

25.6.1.3.4 Inbound iATU Operation

This section discusses how the iATU processes inbound requests. The topics are:

- Overview
- MEM Match Modes
- CFG Handling (Upstream Port)
- FMT Translation
- Inbound Programming Example

25.6.1.3.4.1 Overview

NOTE

- 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 can also be optionally used to further refine the matching process.
- Another difference is that for MEM TLPs, you can select between address matching (as used in outbound operation) or BAR matching. Normally, an Endpoint uses BAR match mode, and a Root Complex uses address mode as a Root Complex normally does not implement BARs.

The following translation rules and limitations apply:

- When there is no match, then the address is untranslated. In addition
- TLPs destined for the configuration registers in an upstream port are not translated.
- TLPs that are not error-free (ECRC, malformed and so on) are not translated.
- Address translation of all TLP types (MEM, CFG, and MSG) except completion is supported in Address Match mode. In BAR Match mode, only translation of MEM is supported.

The setting of the “Match Mode” field in the “iATU Control 2 Register” (IATU_REGION_CTRL_2_OFF_ OUTBOUND_0) determines how iATU inbound matching is done for each TLP type.

Table 25-8. Determination of Match Mode By TLP Type and “Match Mode” Field of iATU Control 2 Register

	Match Mode = 0	Match Mode = 1
MEM	Address Match Mode	BAR Match Mode
CFG0	Routing ID Match Mode	Accept Mode
MSG/MSGD	Address Match Mode	Vendor ID Match Mode

25.6.1.3.4.2 MEM Match Modes

Inbound address translation for MEM TLPs operates in one of two matching modes as determined by the “Inbound Match Mode” field in the “iATU Control 2 Register”.

25.6.1.3.4.2.1 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 address regions as shown in Figure 25-6 .

NOTE

When the “Region Enable” bit of the “Region Control 2 Register” is”0”, then that region is not used for address matching.

When an address match is found then the TLP address field is modified as follows:

Address = Address - Base Address + Target Address

When the TLP address field matches more than one of the 2 address regions, then the first (lowest of the numbers from 0 to 1) enabled region to be matched is used.

25.6.1.3.4.2.2 BAR Match Mode

Looking for an address match is a two-step process.

1. The standard internal PCI Express BAR Matching Mechanism checks if the address field of any MEM request TLP falls into any address region defined by the enabled BAR addresses and masks.

2. When a matched BAR is found, then the iATU compares the BAR ID to the “BAR Number” field in the “iATU Control 2 Register” for all enabled regions.Figure 25-8 and Figure 25-9 provide more details on inbound translation in BAR Match Mode.

NOTE

BAR Match Mode can only be used for MEM transactions

When the PCIe core is operating with 64-bit BARs, the operation is defined as in figure below.

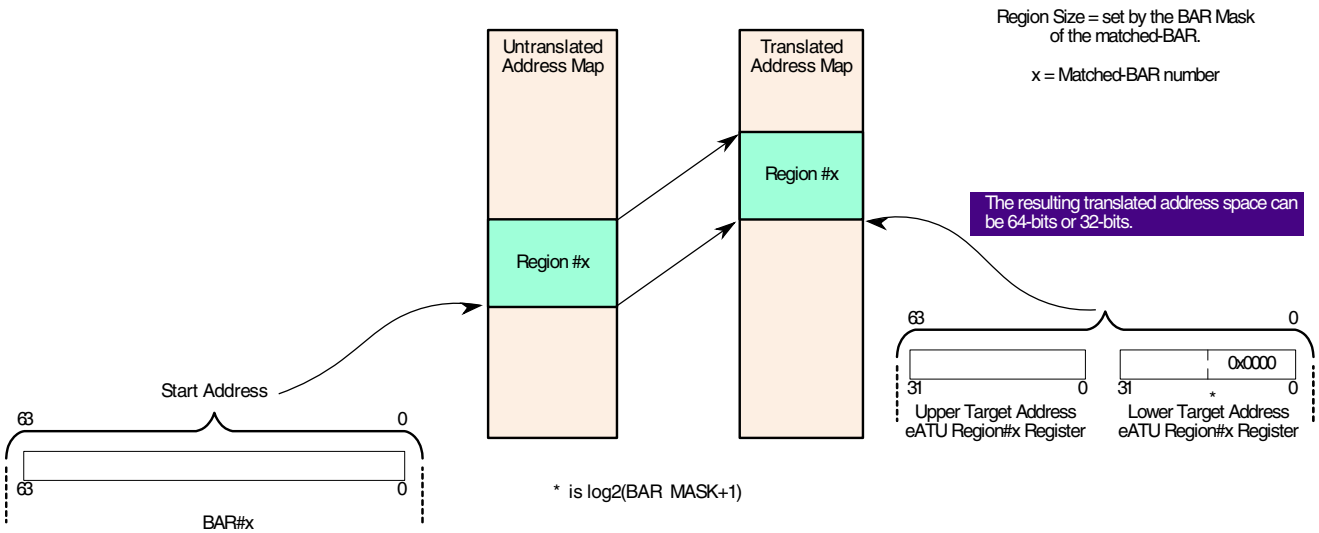


Figure 25-8. iATU Address Region Mapping: Inbound (BAR Match Mode), 64-bit BAR

In address match mode, when the address range does not match one of its BAR ranges in an upstream port, then the device rejects the request with unsupported request (UR) completion status and no translation occurs.

When the PCIe core is operating with 32-bit BARs, the operation is defined as in figure below.

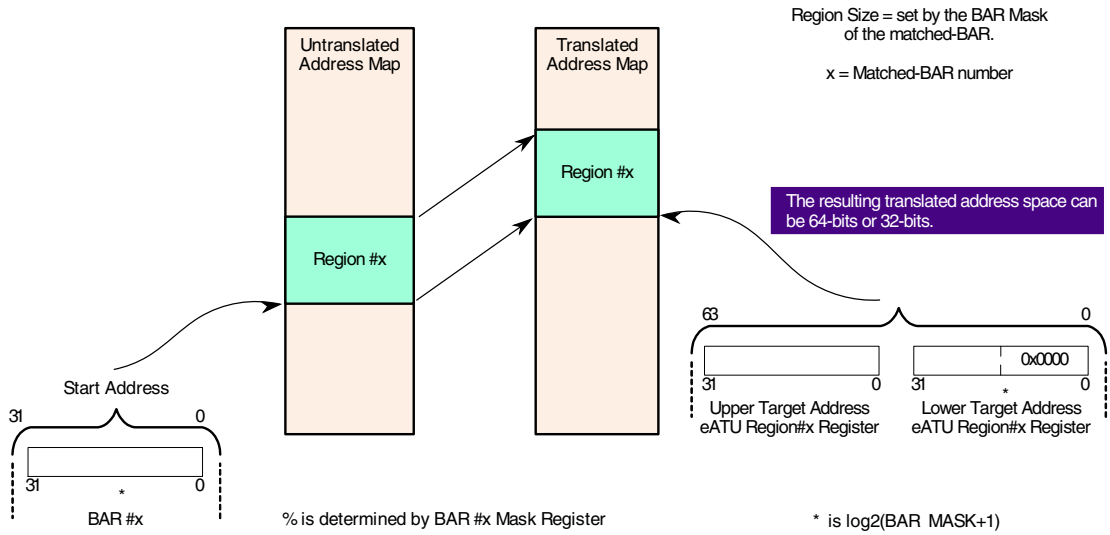


Figure 25-9. iATU Address Region Mapping: Inbound (BAR Match Mode), 32-bit BAR

25.6.1.3.4.3 CFG Handling (Upstream Port)

The PEX controller normally routes CFG TLPs to the configuration registers without translating them. The iATU only translates CFG0 TLPs that the PEX controller has routed to the internal platform. Inbound address translation for CFG0 TLPs operates in one of two matching modes as determined by the “Inbound CFG0 Match Mode” field in the “iATU Control 2 Register”.

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 8-11 of TLP header) as an address. This corresponds to the upper 16 bits of the address in MEM and I/O transactions.

Accept Mode: The PEX controller always accepts CFG0 TLPs even when the CFG bus number does not match the current bus number of the device. This mode follows that behavior. The routing ID of received CFG0 TLPs are ignored when determining a match.

CFG1 Transactions: For CfgRd1/CfgWr1 transactions, the base and limit addresses could enclose the entire 32-bit 4 GB memory space with the routing ID forming the upper 16 bits. The target address maps these CFG transactions to anywhere in application address space.

25.6.1.3.4.4 FMT Translation

The iATU automatically sets the TLP format field for three DWORDs when it detects all zeroes in the upper 32 bits of the translated address. Otherwise it sets it to four DWORDs when it detects a 64-bit address (when there is a “1” in the upper 32 bits of the translated address). When the original address and the translated address are of a different format then the iATU ensures that the TLP header size matches the translated address format.

25.6.1.3.4.5 Inbound Programming Example

*25.6.1.3.4.5.1 **BAR match mode***

Define inbound region 2 as MEM region matching BAR4 (BAR match mode) mapping to 0x8000_0000_2000_0000 in your application memory space

1. Setup the Index Register.

   Write 0x80000002 to Address { 0x700 + 0x200 } to set inbound region 2 as 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 through 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 BAR4.

*25.6.1.3.4.5.2 **Address match mode***

Define inbound Region 0 as: MEM region matching TLPs with addresses in the range 0x00010000 to 0x0005ffff mapped to 0x1000_0000_2000_0000 - 0x1000_0000_2004_ffff in your application memory space

1. Setup the Index 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 through 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.

NOTE

- EP port: Defined MEM regions must be inside an enabled BAR range.
- RC port: Defined MEM 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.

25.6.1.4 Memory Space Addressing

A PCI Express memory transaction can address a 32- or 64-bit memory space. As an initiator, the controller is capable of sending 32- or 64-bit memory packets. Any transaction from the internal platform that (after passing through the translation mechanism) has a translated address greater than 4G is sent as a 64-bit memory packet. Otherwise, a 32-bit memory packet is sent. As a target device, the controller is capable of decoding 32- or 64-bit memory packets. This is done through two 32-bit inbound windows and two 64-bit inbound windows. All inbound addresses are translated to 40-bit internal platform addresses.

Table 25-9. PCI Express Memory Transactions

Transaction	Type[4:0]	FMT[1]	FMT[0]
MRd (32-bit address)	00000	0	0
MRd (64-bit address)	00000	0	1
MWr (32-bit address)	00000	1	0
MWr (64-bit address)	00000	1	1

25.6.1.5 I/O Space Addressing

As an initiator, the controller can send I/O transactions in RC mode only. This can be done by programming one of the outbound translation window's attribute to send I/O transactions. The controller does not support I/O transactions as a target.

Table 25-10. PCI Express I/O Transactions

Transaction	Type	FMT[1]
IORd	00010	0
IOWr	00010	1

The PCI Express Base Specification restricts I/O space to 4 GB (32-bit I/O address).

25.6.1.6 Configuration Space Addressing

The controller can generate outbound configuration transactions in RC mode only. The controller does not support inbound configuration transactions in RC mode.

The controller accepts inbound configuration transactions in EP mode only. The controller does not support generating outbound configuration transactions in EP mode.

Table 25-11. PCI Express Configuration Transactions

Transaction	Type	FMT[1]
CfgRd0	00100	0
CfgWr0	00100	1
CfgRd1	00101	0
CfgWr1	00101	1

Note that all configuration transactions sent on PCI Express require a response, regardless whether they are read or a write configuration transactions.

25.6.1.7 Messages

This section describes outbound message generation and inbound message reception in RC and EP modes.

25.6.1.7.1 Outbound Message Generation

Software can generate the following outbound message transactions:

Outbound Message	Generated by
PME_Turn_Off	See PCI Express PM turnoff message support
Set_Slot_Power_Limit	Outbound iATU translation to MsgD with the appropriate encoding.
Set_Slot_Power_Limit	Writing to the Slot Power Limit Scale and Slot Power Limit Value fields of the Slot Capabilities Register in the PCI Express Capabilities Structure.

25.6.1.7.2 Inbound Messages

The following table provides a complete list of supported inbound messages in RC mode.

Table 25-12. PCI Express RC Inbound Message Handling

Name	Code[7:0]	Routing[2:0]	Action
INTx Interrupt Signaling			
Assert_INTA	0010 0000	100	Assert INTA virtual wire. Sent to GIC.
Assert_INTB	0010 0001	100	Assert INTB virtual wire. Sent to GIC.
Assert_INTC	0010 0010	100	Assert INTC virtual wire. Sent to GIC.
Assert_INTD	0010 0011	100	Assert INTD virtual wire. Sent to GIC.
Deassert_INTA	0010 0100	100	De-assert INTA virtual wire. Sent to GIC.
Deassert_INTB	0010 0101	100	De-assert INTB virtual wire. Sent to GIC.
Deassert_INTC	0010 0110	100	De-assert INTC virtual wire. Sent to GIC.
Deassert_INTD	0010 0111	100	De-assert INTD virtual wire. Sent to GIC.
Power Management			
PM_PME	0001 1000	000	Generate interrupt to GIC
PME_TO_Ack	0001 1011	101	Generate interrupt to GIC
Error Signaling			
ERR_COR	0011 0000	000	Generate interrupt to GIC
ERR_NONFATAL	0011 0001	000	Generate interrupt to GIC
ERR_FATAL	0011 0011	000	Generate interrupt to GIC

The following table provides a complete list of supported inbound messages in EP mode.

Table 25-13. PCI Express EP Inbound Message Handling

Name	Code[7:0]	Routing[2:0]	Action
Power Management			
PME_Turn_Off	0001 1001	011	
Slot Power Limit Support			
Set_Slot_Power_Limit	0101 0000	100	Update power value in PCI Express device capability register in configuration space.

25.6.1.8 Error Handling

The PCI Express specification classifies errors as correctable and uncorrectable. Correctable errors result in degraded performance, but uncorrectable errors generally result in functional failures. As shown in the figure below, uncorrectable errors can further be classified as fatal or non-fatal.

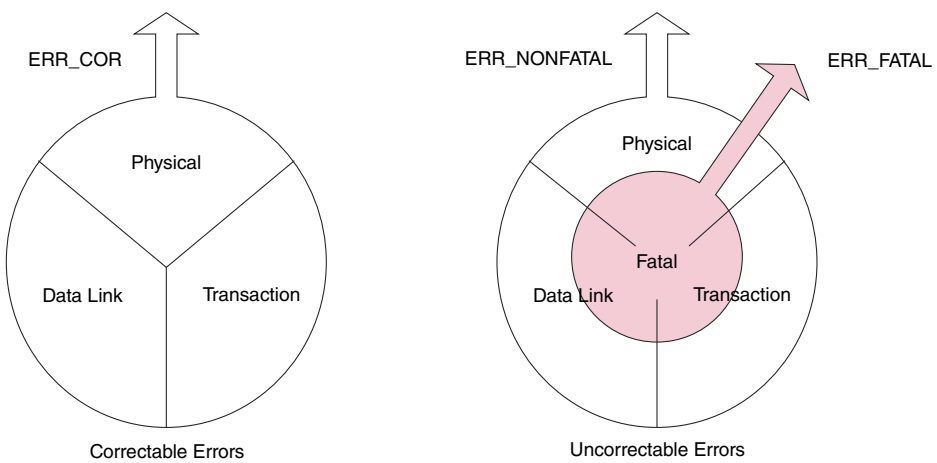


Figure 25-10. PCI Express Error Classification

25.6.1.8.1 PCI Express Error Logging and Signaling

The figure below shows the PCI Express-defined sequence of operations related to signaling and logging of errors detected by a device. Note that the PCI Express controller on this device supports the advanced error handling capabilities shown within the dotted lines.

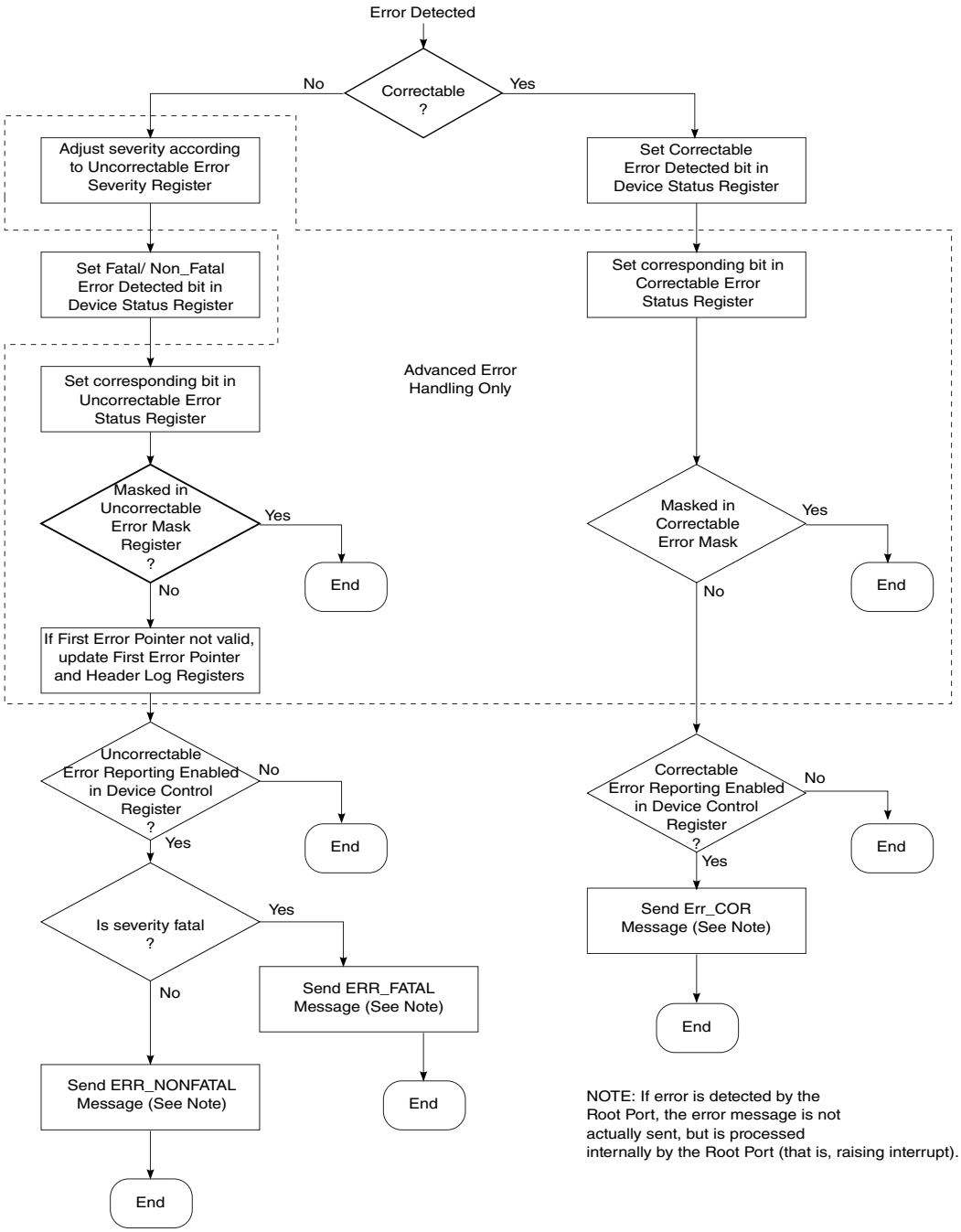


Figure 25-11. PCI Express Device Error Signaling Flowchart

25.6.2 Interrupts

Both message signaled interrupts (MSI) and legacy INTx are supported.

- Legacy INTx interrupts are handled by virtual-wire message transactions. See Inbound Messages.

25.6.3 Initial Credit Advertisement

To prevent overflowing of the receiver's buffers and for ordering compliance purposes, the transmitter cannot send transactions unless it has enough flow control (FC) credits to send. Each device maintains an FC credit pool. The FC information is conveyed between the two link partners by DLLPs during link training (initial credit advertisement). The transaction layer performs the FC accounting functions. One FC unit is four DWs (16-bytes) of data.

Table 25-14. Initial credit advertisement

Credit Type	Initial Credit Advertisement
PH (Memory Write, Message Write)	8
PD (Memory Write, Message Write)	128
NPH (Memory Read, IO Read, Cfg Read, Cfg Write)	12
NPD (IO Write, Cfg Write)	2
CPLH (Memory Read Completion, IO R/W Completion, Cfg R/W Completion)	Infinite
CPLD (Memory Read Completion, IO Read Completion, Cfg Read Completion)	Infinite

25.6.4 Power Management

All device power states are supported with the exception of D3cold. Only L0s ASPM mode is supported if enabled by configuring the Link Control register's bits 1-0 in configuration space. Note that there is no power saving in the controller when the device is put into a non-D0 state. The only power saving is the I/O drivers when the controller is put into a non-L0 link state.

Table 25-15. Power Management State Supported

Component D-State	Permissible Interconnect State	Action
D0	L0, L0s	In full operation.
D1	L0, L0s, L1	All outbound traffic is stalled. All inbound traffic is thrown away. The only exceptions are PME messages and configuration transactions.
D2	L0, L0s, L1	All outbound traffic is stalled. All inbound traffic is thrown away. The only exceptions are PME messages and configuration transactions.

Table continues on the next page...

Table 25-15. Power Management State Supported (continued)

Component D-State	Permissible Interconnect State	Action
D3hot	L0, L0s, L1, L2/L3 Ready	All outbound traffic is stalled. All inbound traffic is thrown away. The only exceptions are PME messages and configuration transactions. Note that if a transition of D3hot to D0 occurs, a reset is performed to the controller's configuration space. In addition, link training restarts.
D3cold	L3	Completely off.

25.6.4.1 L2/L3 Ready Link State

The L2/L3 Ready link state is entered after the EP device is put into a D3hot state followed by a PME_Turn_Off/PME_TO_Ack message handshake protocol. Exiting this state requires a POR reset or a WAKE_B signal from the EP device. The PCI Express controller (in EP mode) does not support the generation of beacon; therefore, as an alternative, the device can use one of the GPIO signals as an enable to an external tristate buffer to generate a WAKE_B signal, as shown in the figure below.

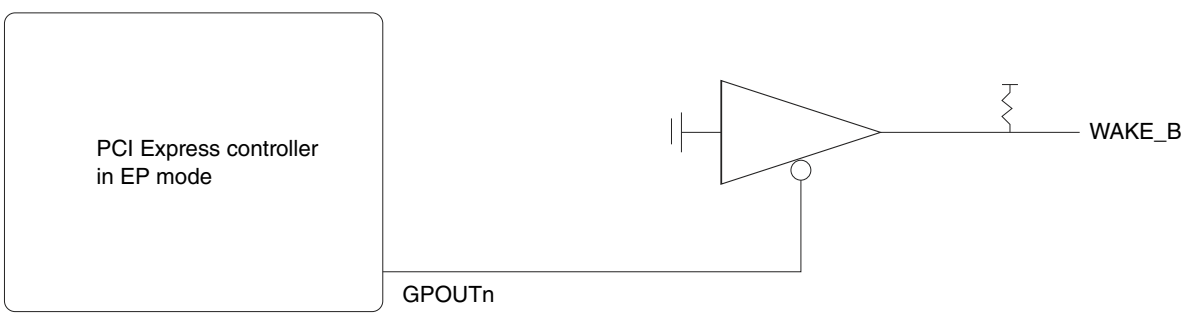


Figure 25-12. WAKE_B Generation Example

In RC mode, the WAKE_B signal from the EP device can be connected to one of the external interrupt inputs to service the WAKE_B request.

25.6.5 Hot Reset

When a hot reset condition occurs, the controller (in both RC and EP mode) initiates a clean-up of all outstanding transactions and returns to an idle state. All configuration register bits that are non-sticky are reset. Link training takes place subsequently. The device is permitted to generate a hot reset condition on the bus when it is configured as an RC device by setting the "Secondary Bus Reset" bit in the Bridge Control Register in

the configuration space. As an EP device, it is not permitted to generate a hot reset condition; it can only detected a hot reset condition and initiates the clean-up procedure appropriately.

25.7 Initialization/Application Information

25.7.1 Gen 3 Link Equalization

Link equalization allows components to adjust the transmitter and receiver setup of each lane to improve the signal quality and meet the requirements for operating at Gen 3 data rates. However, link equalization cannot be performd in loopback mode. Before attempting link training in loopback mode link equalization must be disabled by setting the "Equalization Disable" bit in the Gen3 Control Register at PCI Express extended configuration space offset 890h.

25.7.2 Configuring the chip for inbound CCSR accesses

The chip must be configured using the following procedure prior to receiving any inbound memory transactions targeting CCSR space:

1. Program COHERENCY_CONTROL_3_OFF (offset 8E8h) = 0000_0000h
2. Program COHERENCY_CONTROL_2_OFF (offset 8E4h) = 0000_0000h
3. Program COHERENCY_CONTROL_1_OFF (offset 8E0h) = 1000_0000h

This procedure establishes a boundary between CCSR and memory at 1000_0000h in the system memory map.

NOTE

Any inbound memory transactions targeting CCSR space must set NO_SNOOP = 1 in the TLP.

25.7.3 Poisoned TLP handling

In some cases, the handling of inbound Poisoned TLPs needs special attention. Inbound Poisoned TLP requests are dropped and the Poisoned TLP flag in the Uncorrectable Error Status Register (bit 12) is set, provided it is not masked. However, for an inbound Poisoned TLP response to an outbound request, the response is forwarded to the application as a completion and the Poisoned TLP flag in the Uncorrectable Error Status

Register (bit 12) is set (again, provided it is not masked). This allows the requesting transaction to terminate without stalling, although the error is flagged. In this case, the error handler should cause the response to be discarded.

Chapter 26
Quad Serial Peripheral Interface (QuadSPI)

26.1 The QuadSPI module as implemented on the chip

This section provides details about how the QuadSPI module is implemented on the chip.

26.1.1 LS1012A QuadSPI signals

The following table lists the SoC signal names and their corresponding QuadSPI module signal names used in this chapter:

Table 26-1. LS1012A QuadSPI signals

LS1012A signal name	QuadSPI module signal
QSPI_A_SCK	SCKFA
Not used	SCKFB
QSPI_A_CS0	PCSFA1
Not used	PCSFA2
Not used	PCSFB1
Not used	PCSFB2
QSPI_A_DATA[0:3]	IOFA
Not used	IOFB
Not used	DQSFA
Not used	DQSFB

NOTE

References of all the signals that are not used in the chip should be ignored throughout the chapter.

26.1.2 QuadSPI_MCR[SCLKCFG] mapping

The following table shows the QuadSPI serial clock configuration in the chip.

Table 26-2. QuadSPI_MCR[SCLKCFG] field description

Field	Description
31-24	The QuadSPI_MCR[SCLKCFG] bit is reserved.
SCLKCFG	The QuadSPI clock division is performed by the SCFG_QSPI_CFG register. See QSPI configuration register (QSPI_CFG) for different QuadSPI clock division settings.

26.1.3 Number of supported flash device interface

The chip only supports Flash Device A in single die package (1x4-bit (quad) mode). Parallel mode is not supported in this chip. So, all the sections on Flash Device B and parallel mode should be ignored.

26.1.4 QuadSPI boot initialization sequence

QSPI_A_CS0 is used for boot initialization. QuadSPI is only accessed in 1-bit mode and at low speed for RCW/PBI fetch. See Table 26-4 for the RCW or PBI frequency of QuadSPI during this phase.

PBI can reconfigure the QuadSPI to either higher frequency or 2-/4-bit mode as required for the QuadSPI boot to occur faster.

26.1.5 LS1012A QuadSPI module special consideration

QuadSPI occupies 128 MB of address space from 0x4000_0000 staring address as mentioned in system address space and the QuadSPI AMBA Base parameter also refers to the 0x4000_0000 address.

The QuadSPI module implements the following parameter settings in the chip:

Table 26-3. LS1012A QuadSPI parameter settings

QuadSPI parameters	LS1012A parameter value
Multimaster mode support	Not supported, any references to that should be ignored. **NOTE:** Multimaster mode is not supported so QuadSPI_BUF0CR[MSTRID] field is tied to zero and QuadSPI_BUF0CR[HP_EN] is not supported.
Stop mode support	Yes. Refers to LPM20 low power mode of the chip.

QuadSPI is used as a pre-boot initialization device. The QuadSPI module supports one serial flash ports. See QSPI configuration register (QSPI_CFG) for QuadSPI configuration details.

Table 26-4. QuadSPI clock divider options

		RCW Phase		PBI Phase/Boot Phase	
SCFG_QSPI_CFG[CLK_SEL]	Divider value	SYSCLK (MHz)	Interface clock (MHz)	Platform clock (MHz)	Interface clock (MHz)
0	256	125	1.953	1000 MHz	3.9
1 (Default)	64				15.6
2	32				31.2
3	24				41.7
4	20				50
5	16				62.5
6	12				83.3

26.1.5.1 Supported read modes

The table below provides an overview of this chip's QuadSPI read modes.

Table 26-5. QuadSPI read modes

Read modes		QuadSPI_MCR[DDR_EN]	QuadSPI_MCR[DQS_EN]	Data learning support	For more information
SDR mode	Internal sampling (N/1, I/1)	0	0	No	Internal sampling

26.1.5.2 QuadSPI register reset values

All chip-specific reset values for the QuadSPI registers are shown in the following tables. All other register's reset values are shown in the module memory map.

Table 26-6. QuadSPI module configuration register reset values

Register	Reset Value
Module Configuration Register (QuadSPI_MCR)	000F_4000h

Table 26-7. QuadSPI Buffer Configuration Register Reset Values

Register	Reset Value
Buffer0 Configuration Register (QuadSPI_BUF0CR)	0000_0000h
Buffer1 Configuration Register (QuadSPI_BUF1CR)	0000_0000h
Buffer2 Configuration Register (QuadSPI_BUF2CR)	0000_0000h
Buffer3 Configuration Register (QuadSPI_BUF3CR)	8000_0000h

Table 26-8. Serial Flash A1 Top Address Reset Values

Register	Reset Value
QuadSPI_SFA1AD	47FF_FC00h

Table 26-9. Serial Flash A2 Top Address Reset Values

Register	Reset Value
QuadSPI_SFA2AD	4FFF_FC00h

Table 26-10. Serial Flash B1 Top Address Reset Values

Register	Reset Value
QuadSPI_SFB1AD	57FF_FC00h

Table 26-11. Serial Flash B2 Top Address Reset Values

Register	Reset Value
QuadSPI_SFB2AD	5FFF_FC00h

26.2 Introduction

The Quad Serial Peripheral Interface (QuadSPI) block acts as an interface to a single serial flash device, with up to four bidirectional data lines.

26.2.1 Features

The QuadSPI supports the following features:

- Flexible sequence engine to support various flash vendor devices. As there is no specific standard, the module supports various kind of flashes from different vendors. Refer Serial Flash Devices for example sequences.
- Single, dual, quad modes of operation.
- AHB master to read RX Buffer data via AMBA AHB bus (64-bit width interface) or IP registers space (32-bit access).
    - AHB master can be a DMA with configurable inner loop size.
- Multimaster accesses with priority
    - Flexible and configurable buffer for each master
- Multiple interrupt conditions (see Table 26-19)
- All AHB accesses to flash devices are directly memory mapped to the chip system memory.
- Programmable sequence engine to cater to future command/protocol changes and able to support all existing vendor commands and operations. Software needs to select the corresponding sequence according to the connected flash device.
    - Supports 3-byte and 4-byte addressing.

26.2.2 Block Diagram

The following figure is a block diagram of the Quad Serial Peripheral Interface (QuadSPI) module.

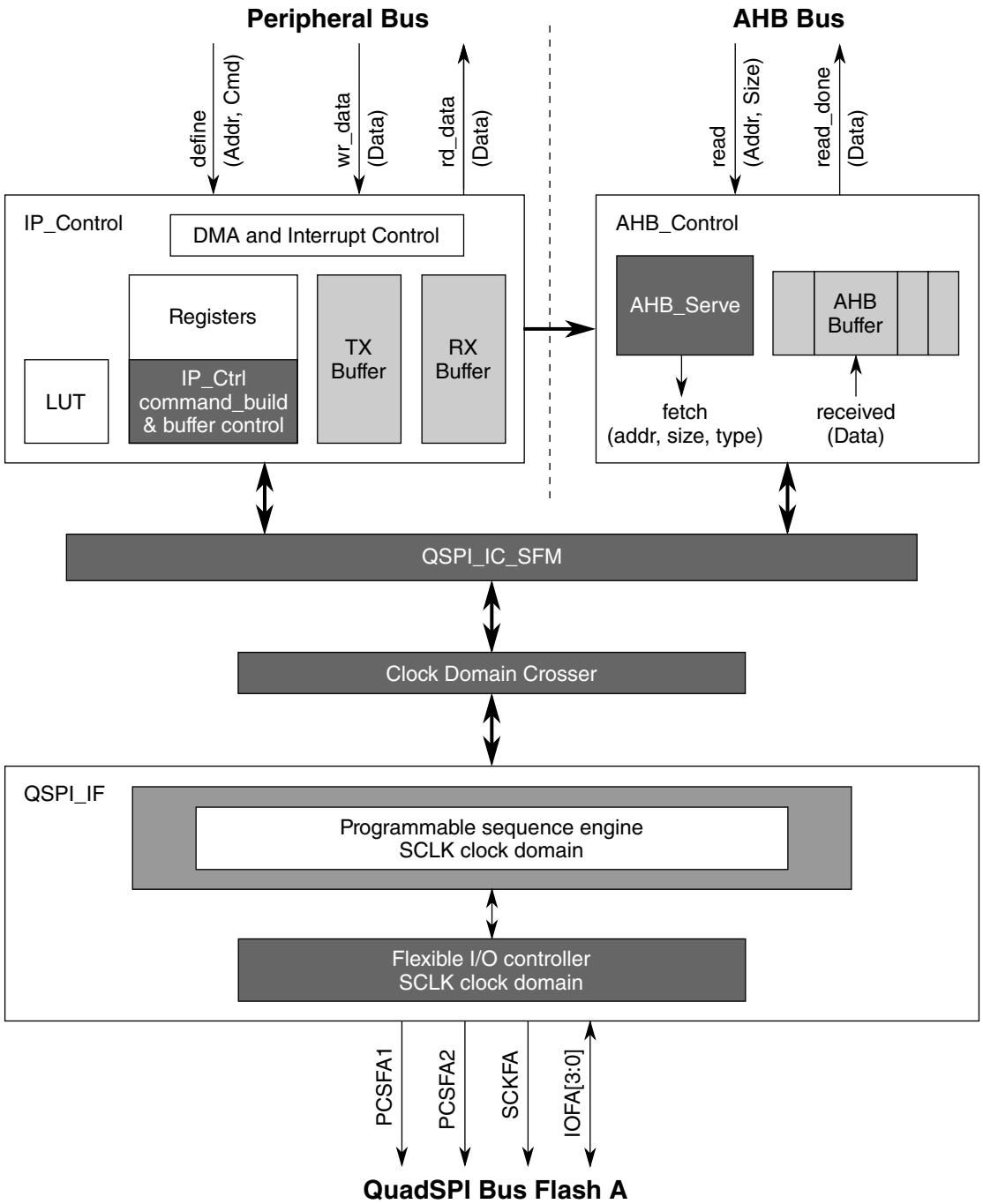


Figure 26-1. QuadSPI Block Diagram

26.2.3 QuadSPI Modes of Operation

For power management through IPS interface access and correct config register programming sequences, QuadSPI supports three modes: normal, module disable and stop mode.

- Normal Mode can be used for write or read accesses to an external serial flash device.
  - Serial Flash Write: Data can be programmed into the flash via the IP interface only. Refer to Flash Programming for further details.
  - Serial Flash Read: Read the contents of the serial flash device. Two separate read channels are available via RX Buffer and AHB Buffer, see Flash Read.
- Stop Mode: The mode is used for power management. When a request is made to enter Stop Mode, the QuadSPI block acknowledges the request and completes the SFM Command in progress, then the system clocks to the QuadSPI block may be shut off
- Module Disable Mode: The mode is used for power management. The clock to the non-memory mapped logic in the QuadSPI can be stopped while in Module Disable Mode. The module enters the mode by setting QSPI_MCR[MDIS] or when a request is asserted by an external controller while QSPI_MCR[DOZE] is set.

26.2.4 Acronyms and Abbreviations

The following table contains acronyms and abbreviations used in this document.

Table 26-12. Acronyms and Abbreviations

Terms	Description
AHB	Advanced High-performance Bus, version of AMBA
AMBA	Advanced Microcontroller Bus Architecture
BE	Big Endian Byte Ordering
CS	Chip Select
DMA	Direct Memory Access
MB	Megabyte. Each MB is 1024 * 1024 bytes
IFM	Individual Flash Mode
LSB	Least Significant Bit
MSB	Most Significant Bit
PCS	Peripheral Chip Select
QSPI, QuadSPI	Quad Serial Peripheral Interface
SCK	Serial Communications Clock
w1c	Write 1 to clear, writing a 1 to this field resets the flag
I/O	Input output. In this document, I/O lines are also referred as **pads**.

26.2.5 Glossary for QuadSPI module

Table 26-13. Glossary

Term	Definition
AHB Command	An AHB Command is a SFM Command triggered by a read access to the address range belonging to the memory mapped access defined in Table 26-18. Refer to AHB Commands for details.
Asserted	A signal that is asserted is in its active state. An active low signal changes from logic level one to logic level zero when asserted, and an active high signal changes from logic level zero to logic level one.
Clear	To clear a bit or bits means to establish logic level zero on the bit or bits.
Clock Phase	Determines when the data should be sampled relative to the active edge of SCK
Clock Polarity	Determines the idle state of the SCK signal.
Drain	To remove entries from a FIFO by software or hardware.
Endianness	Byte Ordering scheme.
Field	Two or more register bits grouped together.
Fill	To add entries to a FIFO by software or hardware.
Host	Refers to another functional block in the device containing the QuadSPI module
Instruction Code	8 bits defining the type of command to be executed.
IP Command	An IP Command is a SFM Command triggered by writing into the QSPI_IPCR[SEQID] field.
Logic level one	The voltage that corresponds to Boolean true (1) state.
Logic level zero	The voltage that corresponds to Boolean false (0) state.
Negated	A signal that is negated is in its inactive state. An active low signal changes from logic level 0 to logic level 1 when negated, and an active high signal changes from logic level 1 to logic level 0.
QSPI_AMBA_BASE	The first address of the serial flash device as presented to the QuadSPI controller. This may be the base of the serial flash in the system address map or it may be a remapping, for instance to 0x0, done by the system. Refer to the system address map.
QSPI_ARDB_BASE	First address of QuadSPI Rx Buffer on system memory map.
Set	To set a bit or bits means to establish logic level one on the bit or bits.
RX Buffer PUSH Event	Addition of valid entries into the RX Buffer. In the default case each Buffer PUSH Event adds 2 entries to the RX Buffer since the interface to the serial clock domain is 64 bits in width. Depending on the number of bytes read from the serial flash device it is possible for the very last Buffer PUSH Event that only one entry is added. The QSPI_RBSR[RDBFL] field is incremented by the number of entries added to the RX Buffer.
RX Buffer POP Event	Removal of valid entries from the RX Buffer. Each Buffer POP Event removes (QSPI_RBCT[WMRK] + 1) valid entries from the buffer. The QSPI_RBSR[RDBFL] field is decremented by the same number and the QSPI_RBSR[RDCTR] field is incremented accordingly.
Individual Flash Mode	Access to a single, individual serial flash device. Refer to Serial Flash Access Schemes for details.
SFM Command	Serial Flash Memory Command. A SFM command consists of an instruction code and all other parameters (for example, size or mode bytes) needed for that specific instruction code. Triggering a command either initiates a transaction on the external serial flash or results in an error. Refer to Table 26-25 for details on errors.
Single/Dual/Quad Instructions	Depending on the serial flash device connected to the QuadSPI module there will be instructions using a different number of data lines. • Single pad: Single line I/O with one data out and one data in line to/from the serial flash device.

Table continues on the next page...

Table 26-13. Glossary (continued)

Term	Definition
	• Dual pad: Dual line I/O with two bidirectional I/O lines, driven alternatively by the serial flash device or the QuadSPI module • Quad pad: Quad line I/O with 4 bidirectional I/O lines, driven alternatively by the serial flash device or the QuadSPI
Transaction	A transaction consists of all flags, data and signals in either direction to execute a command for an attached serial flash device. It is a combination of chip select, sclk, instruction code, address, mode- and/or dummy bytes, transmit and/or receive data.
LUT	Look-up table.

26.3 External Signal Description

This section provides the external signal information of the QuadSPI module.

The following table lists the external signals belonging to the module in conjunction with the different modes of operation.

Table 26-14. Signal Properties

Signal Name	Function	Direction	Description
PCSFA1	Peripheral Chip Select Flash A1	O	This signal is the chip select for the serial flash device A1. A1 represents the first of the two flash devices that share IOFA.
PCSFA2	Peripheral Chip Select Flash A2	O	This signal is the chip select for the serial flash device A2. A2 represents the the second of the two flash devices that share IOFA.
SCKFA	Serial Clock Flash A	O	This signal is the serial clock output to the serial flash device A.
IOFA[3:0]	Serial I/O Flash A	I/O	These signals are the data I/O lines to/from the serial flash device A. Refer to Driving External Signals for details about the signal drive and timing behavior. Note that the signal pins of the serial flash device may change their function according to the SFM Command executed, leaving them as control inputs when Single and Dual Instructions are executed. The module supports driving these inputs to dedicated values. In single I/O mode, QuadSPI drives data on IOFA[0] and expects data on IOFA[1].

26.3.1 Driving External Signals

The different phases of the serial flash access scheme are shown in the following figure.

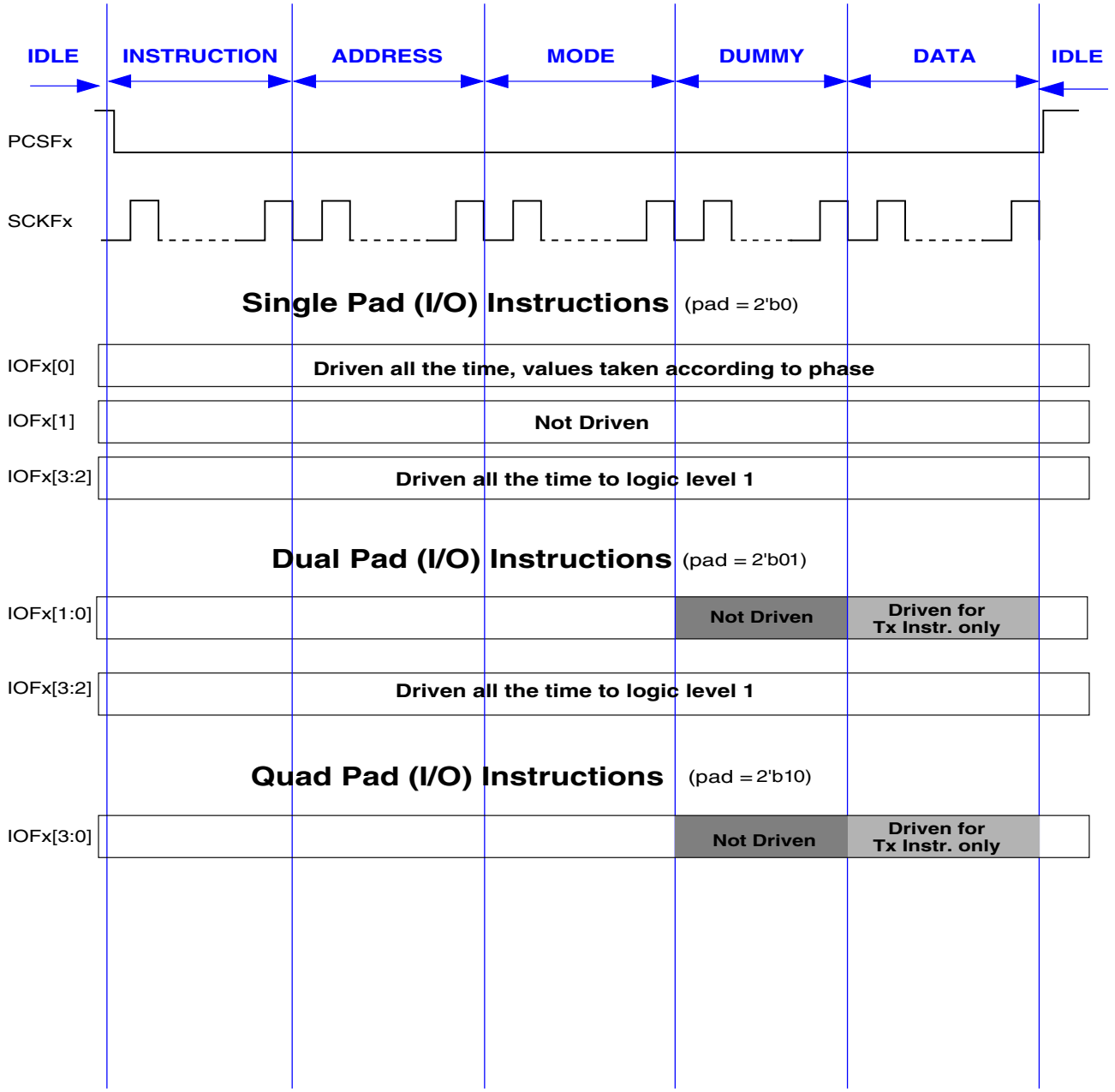


Figure 26-2. Serial Flash Access Scheme

The different phases and the I/O driving characteristics of the QuadSPI module are characterized in the following way:

- IDLE: Serial flash device not selected. No interaction with the serial flash device. All IOFx signals driven.
- INSTRUCTION: Serial flash device selected. The instruction is sent to the serial flash device. All IOFx signals are driven.

- ADDRESS: Serial Flash Address is sent to the device. All IOFx signals are driven. Note that this phase is not applicable for all SFM Commands.
- MODE: Mode bytes are sent to the serial flash device. All IOFx signals are driven. Note that this phase is not applicable for all SFM Commands.
- DUMMY: Dummy clocks are provided to the serial flash device. Refer to the Figure 26-2 for the IOFx signals driven. The actual data lines required for the SFM Command executed are not driven for data read commands. Note that this phase is not applicable for all SFM Commands.
- DATA: Serial flash data are sent to or received from the serial flash device. Refer to the preceding figure for the IOFx signals driven. The actual data lines required for the SFM Command executed are not driven for data read commands. Note that this phase is not applicable for all SFM Commands.

The PCSFx and SCKFx signals are driven permanently throughout all the phases.

In Individual Flash Mode this applies to the selected flash device.

26.4 Memory Map and Register Definition

This section provides the memory map and register definitions of the QuadSPI module.

26.4.1 Register Write Access

This section describes the write access restriction terms that apply to all registers, which can be one of the following:

- **Register Write Access Restriction**

  For each register bit and register field, the write access conditions are specified in the detailed register description. A description of the write access conditions is given in the following table. If, for a specific register bit or field, none of the given write access conditions is fulfilled, any write attempt to this register bit or field is ignored without any notification. The values of the bits or fields are not changed.

  The condition term [A or B] indicates that the register or field can be written to if at least one of the conditions is fulfilled.

Table 26-15. Register Write Access Restrictions

Condition	Description
Anytime	No write access restriction.
Disabled Mode	Write access only if *QSPI_MCR[MDIS] = 1*.
Normal Mode	Write access only if the module is in *Normal Mode*.

- **Register Write Access Requirements**

  All registers can be accessed with 8-bit, 16-bit, and 32-bit wide operations. For some of the registers, at least a 16/32-bit wide write access is required to ensure correct operation. This write access requirement is stated in the detailed register description for each register affected.

26.4.2 Peripheral Bus Register Descriptions

This section provides the memory map and register definitions of the QuadSPI module.

NOTE

Access to location and 0x190 does not give transfer error.

QuadSPI memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
155_0000	Module Configuration Register (QuadSPI_MCR)	32	R/W	See section	26.4.2.1/ 1440
155_0008	IP Configuration Register (QuadSPI_IPCR)	32	R/W	0000_0000h	26.4.2.2/ 1443
155_000C	Flash Configuration Register (QuadSPI_FLSHCR)	32	R/W	0000_0303h	26.4.2.3/ 1443
155_0010	Buffer0 Configuration Register (QuadSPI_BUF0CR)	32	R/W	See section	26.4.2.4/ 1444
155_0014	Buffer1 Configuration Register (QuadSPI_BUF1CR)	32	R/W	See section	26.4.2.5/ 1445
155_0018	Buffer2 Configuration Register (QuadSPI_BUF2CR)	32	R/W	See section	26.4.2.6/ 1446
155_001C	Buffer3 Configuration Register (QuadSPI_BUF3CR)	32	R/W	See section	26.4.2.7/ 1447

Table continues on the next page...

QuadSPI memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
155_0020	Buffer Generic Configuration Register (QuadSPI_BFGENCR)	32	R/W	0000_0000h	26.4.2.8/ 1448
155_0030	Buffer0 Top Index Register (QuadSPI_BUF0IND)	32	R/W	0000_0000h	26.4.2.9/ 1449
155_0034	Buffer1 Top Index Register (QuadSPI_BUF1IND)	32	R/W	0000_0000h	26.4.2.10/ 1450
155_0038	Buffer2 Top Index Register (QuadSPI_BUF2IND)	32	R/W	0000_0000h	26.4.2.11/ 1450
155_0100	Serial Flash Address Register (QuadSPI_SFAR)	32	R/W	0000_0000h	26.4.2.12/ 1451
155_0108	Sampling Register (QuadSPI_SMPR)	32	R/W	0000_0000h	26.4.2.13/ 1451
155_010C	RX Buffer Status Register (QuadSPI_RBSR)	32	R	0000_0000h	26.4.2.14/ 1453
155_0110	RX Buffer Control Register (QuadSPI_RBCT)	32	R/W	0000_0000h	26.4.2.15/ 1454
155_0150	TX Buffer Status Register (QuadSPI_TBSR)	32	R	0000_0000h	26.4.2.16/ 1455
155_0154	TX Buffer Data Register (QuadSPI_TBDR)	32	R/W	0000_0000h	26.4.2.17/ 1455
155_015C	Status Register (QuadSPI_SR)	32	R	0000_3800h	26.4.2.18/ 1457
155_0160	Flag Register (QuadSPI_FR)	32	w1c	0800_0000h	26.4.2.19/ 1459
155_0164	Interrupt and DMA Request Select and Enable Register (QuadSPI_RSER)	32	R/W	0000_0000h	26.4.2.20/ 1462
155_0168	Sequence Suspend Status Register (QuadSPI_SPNDST)	32	R	0000_0000h	26.4.2.21/ 1466
155_016C	Sequence Pointer Clear Register (QuadSPI_SPTRCLR)	32	R/W	0000_0000h	26.4.2.22/ 1468
155_0180	Serial Flash A1 Top Address (QuadSPI_SFA1AD)	32	R/W	See section	26.4.2.23/ 1469
155_0184	Serial Flash A2 Top Address (QuadSPI_SFA2AD)	32	R/W	See section	26.4.2.24/ 1469
155_0200	RX Buffer Data Register (QuadSPI_RBDR0)	32	R	0000_0000h	26.4.2.25/ 1470
155_0204	RX Buffer Data Register (QuadSPI_RBDR1)	32	R	0000_0000h	26.4.2.25/ 1470
155_0208	RX Buffer Data Register (QuadSPI_RBDR2)	32	R	0000_0000h	26.4.2.25/ 1470
155_020C	RX Buffer Data Register (QuadSPI_RBDR3)	32	R	0000_0000h	26.4.2.25/ 1470
155_0210	RX Buffer Data Register (QuadSPI_RBDR4)	32	R	0000_0000h	26.4.2.25/ 1470

Table continues on the next page...

QuadSPI memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
155_0214	RX Buffer Data Register (QuadSPI_RBDR5)	32	R	0000_0000h	26.4.2.25/ 1470
155_0218	RX Buffer Data Register (QuadSPI_RBDR6)	32	R	0000_0000h	26.4.2.25/ 1470
155_021C	RX Buffer Data Register (QuadSPI_RBDR7)	32	R	0000_0000h	26.4.2.25/ 1470
155_0220	RX Buffer Data Register (QuadSPI_RBDR8)	32	R	0000_0000h	26.4.2.25/ 1470
155_0224	RX Buffer Data Register (QuadSPI_RBDR9)	32	R	0000_0000h	26.4.2.25/ 1470
155_0228	RX Buffer Data Register (QuadSPI_RBDR10)	32	R	0000_0000h	26.4.2.25/ 1470
155_022C	RX Buffer Data Register (QuadSPI_RBDR11)	32	R	0000_0000h	26.4.2.25/ 1470
155_0230	RX Buffer Data Register (QuadSPI_RBDR12)	32	R	0000_0000h	26.4.2.25/ 1470
155_0234	RX Buffer Data Register (QuadSPI_RBDR13)	32	R	0000_0000h	26.4.2.25/ 1470
155_0238	RX Buffer Data Register (QuadSPI_RBDR14)	32	R	0000_0000h	26.4.2.25/ 1470
155_023C	RX Buffer Data Register (QuadSPI_RBDR15)	32	R	0000_0000h	26.4.2.25/ 1470
155_0240	RX Buffer Data Register (QuadSPI_RBDR16)	32	R	0000_0000h	26.4.2.25/ 1470
155_0244	RX Buffer Data Register (QuadSPI_RBDR17)	32	R	0000_0000h	26.4.2.25/ 1470
155_0248	RX Buffer Data Register (QuadSPI_RBDR18)	32	R	0000_0000h	26.4.2.25/ 1470
155_024C	RX Buffer Data Register (QuadSPI_RBDR19)	32	R	0000_0000h	26.4.2.25/ 1470
155_0250	RX Buffer Data Register (QuadSPI_RBDR20)	32	R	0000_0000h	26.4.2.25/ 1470
155_0254	RX Buffer Data Register (QuadSPI_RBDR21)	32	R	0000_0000h	26.4.2.25/ 1470
155_0258	RX Buffer Data Register (QuadSPI_RBDR22)	32	R	0000_0000h	26.4.2.25/ 1470
155_025C	RX Buffer Data Register (QuadSPI_RBDR23)	32	R	0000_0000h	26.4.2.25/ 1470
155_0260	RX Buffer Data Register (QuadSPI_RBDR24)	32	R	0000_0000h	26.4.2.25/ 1470
155_0264	RX Buffer Data Register (QuadSPI_RBDR25)	32	R	0000_0000h	26.4.2.25/ 1470
155_0268	RX Buffer Data Register (QuadSPI_RBDR26)	32	R	0000_0000h	26.4.2.25/ 1470

Table continues on the next page...

QuadSPI memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
155_026C	RX Buffer Data Register (QuadSPI_RBDR27)	32	R	0000_0000h	26.4.2.25/ 1470
155_0270	RX Buffer Data Register (QuadSPI_RBDR28)	32	R	0000_0000h	26.4.2.25/ 1470
155_0274	RX Buffer Data Register (QuadSPI_RBDR29)	32	R	0000_0000h	26.4.2.25/ 1470
155_0278	RX Buffer Data Register (QuadSPI_RBDR30)	32	R	0000_0000h	26.4.2.25/ 1470
155_027C	RX Buffer Data Register (QuadSPI_RBDR31)	32	R	0000_0000h	26.4.2.25/ 1470
155_0300	LUT Key Register (QuadSPI_LUTKEY)	32	R/W	5AF0_5AF0h	26.4.2.26/ 1470
155_0304	LUT Lock Configuration Register (QuadSPI_LCKCR)	32	R/W	0000_0002h	26.4.2.27/ 1471
155_0310	Look-up Table register (QuadSPI_LUT0)	32	R/W	See section	26.4.2.28/ 1472
155_0314	Look-up Table register (QuadSPI_LUT1)	32	R/W	See section	26.4.2.28/ 1472
155_0318	Look-up Table register (QuadSPI_LUT2)	32	R/W	See section	26.4.2.28/ 1472
155_031C	Look-up Table register (QuadSPI_LUT3)	32	R/W	See section	26.4.2.28/ 1472
155_0320	Look-up Table register (QuadSPI_LUT4)	32	R/W	See section	26.4.2.28/ 1472
155_0324	Look-up Table register (QuadSPI_LUT5)	32	R/W	See section	26.4.2.28/ 1472
155_0328	Look-up Table register (QuadSPI_LUT6)	32	R/W	See section	26.4.2.28/ 1472
155_032C	Look-up Table register (QuadSPI_LUT7)	32	R/W	See section	26.4.2.28/ 1472
155_0330	Look-up Table register (QuadSPI_LUT8)	32	R/W	See section	26.4.2.28/ 1472
155_0334	Look-up Table register (QuadSPI_LUT9)	32	R/W	See section	26.4.2.28/ 1472
155_0338	Look-up Table register (QuadSPI_LUT10)	32	R/W	See section	26.4.2.28/ 1472
155_033C	Look-up Table register (QuadSPI_LUT11)	32	R/W	See section	26.4.2.28/ 1472
155_0340	Look-up Table register (QuadSPI_LUT12)	32	R/W	See section	26.4.2.28/ 1472
155_0344	Look-up Table register (QuadSPI_LUT13)	32	R/W	See section	26.4.2.28/ 1472
155_0348	Look-up Table register (QuadSPI_LUT14)	32	R/W	See section	26.4.2.28/ 1472

Table continues on the next page...

QuadSPI memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
155_034C	Look-up Table register (QuadSPI_LUT15)	32	R/W	See section	26.4.2.28/ 1472
155_0350	Look-up Table register (QuadSPI_LUT16)	32	R/W	See section	26.4.2.28/ 1472
155_0354	Look-up Table register (QuadSPI_LUT17)	32	R/W	See section	26.4.2.28/ 1472
155_0358	Look-up Table register (QuadSPI_LUT18)	32	R/W	See section	26.4.2.28/ 1472
155_035C	Look-up Table register (QuadSPI_LUT19)	32	R/W	See section	26.4.2.28/ 1472
155_0360	Look-up Table register (QuadSPI_LUT20)	32	R/W	See section	26.4.2.28/ 1472
155_0364	Look-up Table register (QuadSPI_LUT21)	32	R/W	See section	26.4.2.28/ 1472
155_0368	Look-up Table register (QuadSPI_LUT22)	32	R/W	See section	26.4.2.28/ 1472
155_036C	Look-up Table register (QuadSPI_LUT23)	32	R/W	See section	26.4.2.28/ 1472
155_0370	Look-up Table register (QuadSPI_LUT24)	32	R/W	See section	26.4.2.28/ 1472
155_0374	Look-up Table register (QuadSPI_LUT25)	32	R/W	See section	26.4.2.28/ 1472
155_0378	Look-up Table register (QuadSPI_LUT26)	32	R/W	See section	26.4.2.28/ 1472
155_037C	Look-up Table register (QuadSPI_LUT27)	32	R/W	See section	26.4.2.28/ 1472
155_0380	Look-up Table register (QuadSPI_LUT28)	32	R/W	See section	26.4.2.28/ 1472
155_0384	Look-up Table register (QuadSPI_LUT29)	32	R/W	See section	26.4.2.28/ 1472
155_0388	Look-up Table register (QuadSPI_LUT30)	32	R/W	See section	26.4.2.28/ 1472
155_038C	Look-up Table register (QuadSPI_LUT31)	32	R/W	See section	26.4.2.28/ 1472
155_0390	Look-up Table register (QuadSPI_LUT32)	32	R/W	See section	26.4.2.28/ 1472
155_0394	Look-up Table register (QuadSPI_LUT33)	32	R/W	See section	26.4.2.28/ 1472
155_0398	Look-up Table register (QuadSPI_LUT34)	32	R/W	See section	26.4.2.28/ 1472
155_039C	Look-up Table register (QuadSPI_LUT35)	32	R/W	See section	26.4.2.28/ 1472
155_03A0	Look-up Table register (QuadSPI_LUT36)	32	R/W	See section	26.4.2.28/ 1472

Table continues on the next page...

QuadSPI memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
155_03A4	Look-up Table register (QuadSPI_LUT37)	32	R/W	See section	26.4.2.28/ 1472
155_03A8	Look-up Table register (QuadSPI_LUT38)	32	R/W	See section	26.4.2.28/ 1472
155_03AC	Look-up Table register (QuadSPI_LUT39)	32	R/W	See section	26.4.2.28/ 1472
155_03B0	Look-up Table register (QuadSPI_LUT40)	32	R/W	See section	26.4.2.28/ 1472
155_03B4	Look-up Table register (QuadSPI_LUT41)	32	R/W	See section	26.4.2.28/ 1472
155_03B8	Look-up Table register (QuadSPI_LUT42)	32	R/W	See section	26.4.2.28/ 1472
155_03BC	Look-up Table register (QuadSPI_LUT43)	32	R/W	See section	26.4.2.28/ 1472
155_03C0	Look-up Table register (QuadSPI_LUT44)	32	R/W	See section	26.4.2.28/ 1472
155_03C4	Look-up Table register (QuadSPI_LUT45)	32	R/W	See section	26.4.2.28/ 1472
155_03C8	Look-up Table register (QuadSPI_LUT46)	32	R/W	See section	26.4.2.28/ 1472
155_03CC	Look-up Table register (QuadSPI_LUT47)	32	R/W	See section	26.4.2.28/ 1472
155_03D0	Look-up Table register (QuadSPI_LUT48)	32	R/W	See section	26.4.2.28/ 1472
155_03D4	Look-up Table register (QuadSPI_LUT49)	32	R/W	See section	26.4.2.28/ 1472
155_03D8	Look-up Table register (QuadSPI_LUT50)	32	R/W	See section	26.4.2.28/ 1472
155_03DC	Look-up Table register (QuadSPI_LUT51)	32	R/W	See section	26.4.2.28/ 1472
155_03E0	Look-up Table register (QuadSPI_LUT52)	32	R/W	See section	26.4.2.28/ 1472
155_03E4	Look-up Table register (QuadSPI_LUT53)	32	R/W	See section	26.4.2.28/ 1472
155_03E8	Look-up Table register (QuadSPI_LUT54)	32	R/W	See section	26.4.2.28/ 1472
155_03EC	Look-up Table register (QuadSPI_LUT55)	32	R/W	See section	26.4.2.28/ 1472
155_03F0	Look-up Table register (QuadSPI_LUT56)	32	R/W	See section	26.4.2.28/ 1472
155_03F4	Look-up Table register (QuadSPI_LUT57)	32	R/W	See section	26.4.2.28/ 1472
155_03F8	Look-up Table register (QuadSPI_LUT58)	32	R/W	See section	26.4.2.28/ 1472

Table continues on the next page...

QuadSPI memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
155_03FC	Look-up Table register (QuadSPI_LUT59)	32	R/W	See section	26.4.2.28/ 1472
155_0400	Look-up Table register (QuadSPI_LUT60)	32	R/W	See section	26.4.2.28/ 1472
155_0404	Look-up Table register (QuadSPI_LUT61)	32	R/W	See section	26.4.2.28/ 1472
155_0408	Look-up Table register (QuadSPI_LUT62)	32	R/W	See section	26.4.2.28/ 1472
155_040C	Look-up Table register (QuadSPI_LUT63)	32	R/W	See section	26.4.2.28/ 1472

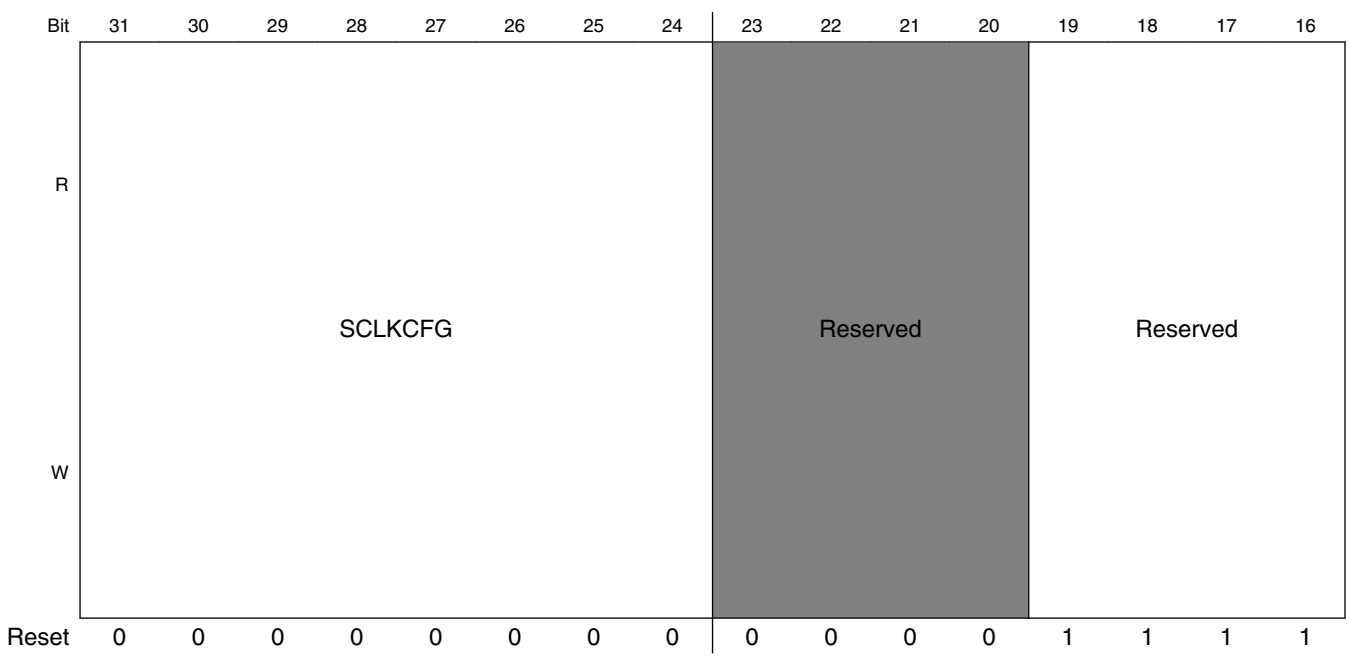
26.4.2.1 Module Configuration Register (QuadSPI_MCR)

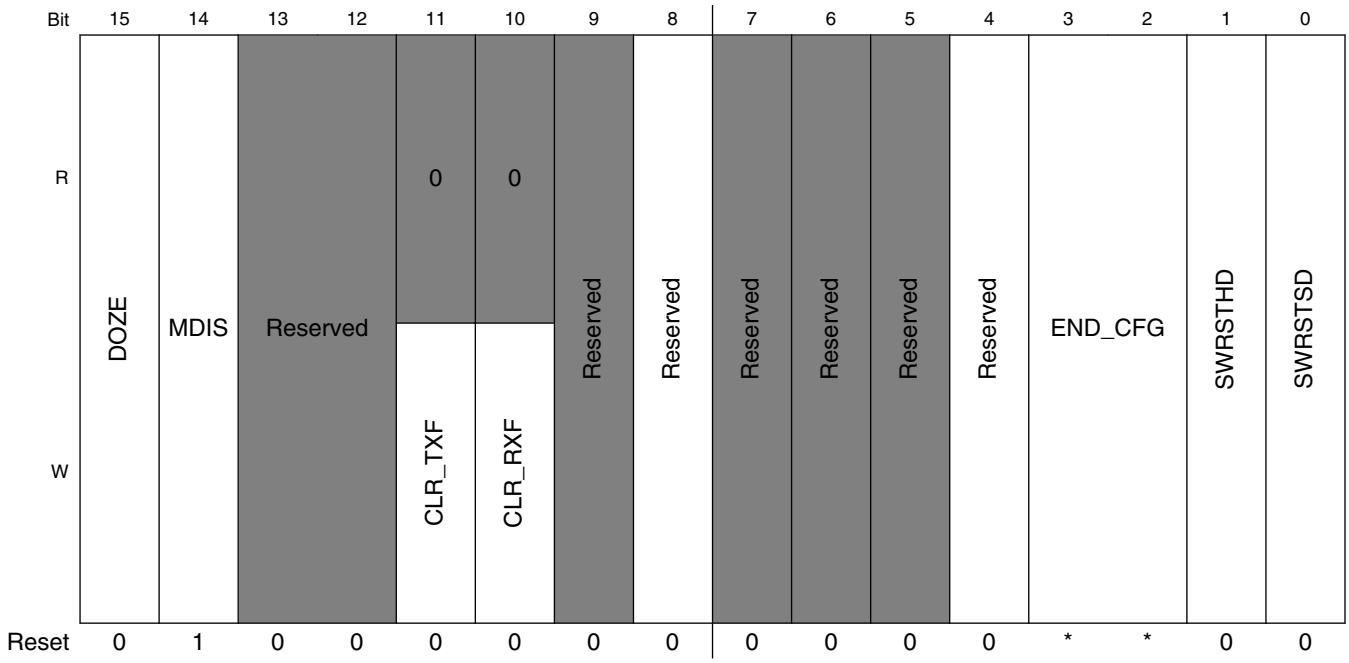
The QuadSPI_MCR holds configuration data associated with QuadSPI operation.

Write:

- *SCLKCFG: Disabled Mode*
- *All other fields: Anytime*

Address: 155_0000h base + 0h offset = 155_0000h





* Notes:
- END_CFG field: See the module configuration for the device specific reset value.

QuadSPI_MCR field descriptions

Field	Description
31–24 SCLKCFG	Serial Clock Configuration. This field configuration is chip specific. For details, refer to chip-specific QuadSPI information. It may be used for dividing clocks.
23–20 Reserved	This field is reserved.
19–16 Reserved	This field is reserved. This field is reserved and should always be set to 0xF.
15 DOZE	Doze Enable. The DOZE bit provides support for externally controlled Doze Mode power-saving mechanism. 0 A doze request will be ignored by the QuadSPI module 1 A doze request will be processed by the QuadSPI module
14 MDIS	Module Disable. The MDIS bit allows the clock to the non-memory mapped logic in the QuadSPI to be stopped, putting the QuadSPI in a software controlled power-saving state. 0 Enable QuadSPI clocks. 1 Allow external logic to disable QuadSPI clocks.
13–12 Reserved	This field is reserved.
11 CLR_TXF	Clear TX FIFO/Buffer. Invalidates the TX Buffer content. This is a self-clearing field. 0 No action. 1 Read and write pointers of the TX Buffer are reset to 0. QSPI_TBSR[TRCTR] is reset to 0.

Table continues on the next page...

QuadSPI_MCR field descriptions (continued)

Field	Description
10 CLR_RXF	Clear RX FIFO. Invalidates the RX Buffer. This is a self-clearing field. 0 No action. 1 Read and write pointers of the RX Buffer are reset to 0. QSPI_RBSR[RDBFL] is reset to 0.
9 Reserved	This field is reserved.
8 Reserved	This field is reserved.
7 Reserved	This field is reserved.
6 Reserved	This field is reserved.
5 Reserved	This field is reserved.
4 Reserved	This field is reserved.
3–2 END_CFG	Defines the endianness of the QuadSPI module. For more details refer to Byte Ordering Endianess
1 SWRSTHD	Software reset for AHB domain 0 No action 1 AHB domain flops are reset. Does not reset configuration registers. It is advisable to reset both the serial flash domain and AHB domain at the same time. Resetting only one domain might lead to side effects. **NOTE:** The software resets need the clock to be running to propagate to the design. The MCR[MDIS] should therefore be set to 0 when the software reset bits are asserted. Also, before they can be deasserted again (by setting MCR[SWRSTHD] to 0), it is recommended to set the MCR[MDIS] bit to 1. Once the software resets have been deasserted, the normal operation can be started by setting the MCR[MDIS] bit to 0.
0 SWRSTSD	Software reset for serial flash domain 0 No action 1 Serial Flash domain flops are reset. Does not reset configuration registers. It is advisable to reset both the serial flash domain and AHB domain at the same time. Resetting only one domain might lead to side effects. **NOTE:** The software resets need the clock to be running to propagate to the design. The MCR[MDIS] should therefore be set to 0 when the software reset bits are asserted. Also, before they can be deasserted again (by setting MCR[SWRSTSD] to 0), it is recommended to set the MCR[MDIS] bit to 1. Once the software resets have been deasserted, the normal operation can be started by setting the MCR[MDIS] bit to 0.

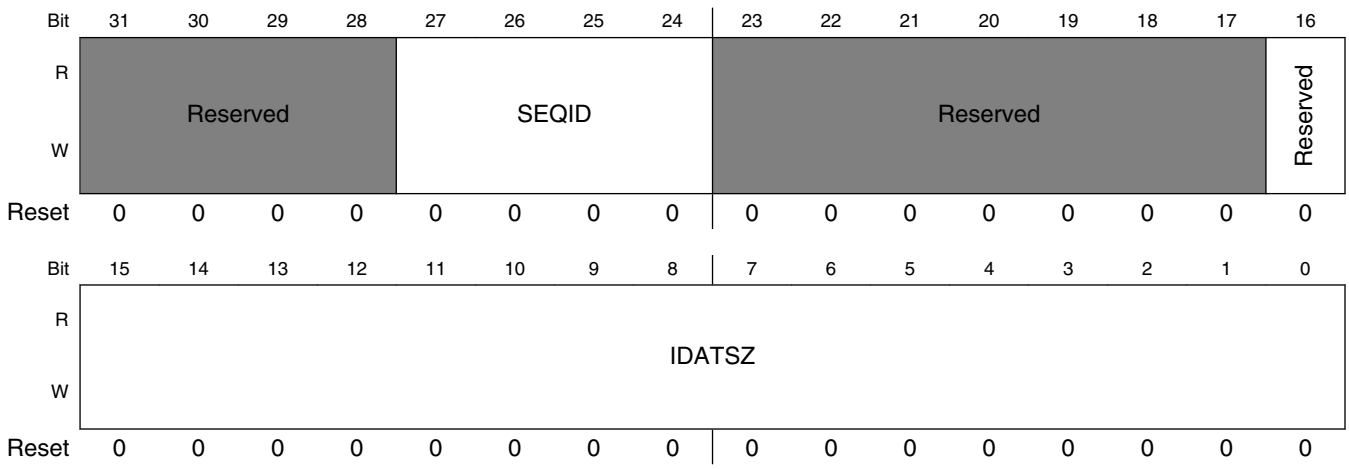
26.4.2.2 IP Configuration Register (QuadSPI_IPCR)

The IP configuration register provides all the configuration required for an IP initiated command. An IP command can be triggered by writing in the SEQID field of this register. If the SEQID field is written successfully, a new command to the external serial flash is started as per the sequence pointed to by the SEQID field. Refer to Normal Mode , for details about the command triggering and command execution.

Write:

- *QSPI_SR[IP_ACC]=0*

Address: 155_0000h base + 8h offset = 155_0008h



QuadSPI_IPCR field descriptions

Field	Description
31–28 Reserved	This field is reserved.
27–24 SEQID	Points to a sequence in the Look-up table. This field defines bits [6:2] of the LUT index. The bits [1:0] are always assumed to be 0. Refer to Look-up Table for more details. A write to this field triggers a transaction on the serial flash interface.
23–17 Reserved	This field is reserved.
16 Reserved	This field is reserved.
IDATSZ	IP data transfer size. Defines the data transfer size in bytes of the IP command.

26.4.2.3 Flash Configuration Register (QuadSPI_FLSHCR)

The Flash configuration register contains the flash device specific timings that must be met by the QuadSPI controller for the device to function correctly.

Write:

- *QSPI_SR[AHB_ACC] = 0*
- *QSPI_SR[IP_ACC] = 0*

Address: 155_0000h base + Ch offset = 155_000Ch

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																				TCSH				Reserved				TCSS			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1

QuadSPI_FLSHCR field descriptions

Field	Description
31–12 -	This field is reserved. Reserved.
11–8 TCSH	Serial flash CS hold time in terms of serial flash clock cycles. Default value '3' should be used in case AHB prefetch size is less than or equal to 32-bytes DDR/16-bytes SDR. **NOTE:** The actual delay between chip select and clock is defined as: • TCSH = 1 SCK clk if N= 0/1 else, N SCK clk if N>1, where N is the setting of TCSH.
7–4 Reserved	This field is reserved. Reserved.
TCSS	Serial flash CS setup time in terms of serial flash clock cycles. **NOTE:** 1. The actual delay between chip select and clock is defined as: • TCSS = 0.5 SCK clk if N= 0/1 else, N+0.5 SCK clk if N>1, where N is the setting of TCSS. 2. Any update to the TCSS register bits is visible on the flash interface only from the second transaction following the update.

26.4.2.4 Buffer0 Configuration Register (QuadSPI_BUF0CR)

This register provides the configuration for any access to buffer0. An access is routed to buffer0 when the master port number of the incoming AHB request matches the MSTRID field of BUF0CR. Any buffer "miss" leads to a serial flash transaction being triggered as per the sequence pointed to the SEQID field. Buffer0 may also be configured as a high priority buffer by setting the HP_EN field of this register.

Write:

- *QSPI_SR[AHB_ACC] = 0*

Address: 155_0000h base + 10h offset = 155_0010h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	HP_ EN	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	ADATSZ								Reserved				MSTRID			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	*	*	*	*

* Notes:
- MSTRID field: See the module configuration for reset value.

QuadSPI_BUF0CR field descriptions

Field	Description
31 HP_EN	High Priority Enable. When set, the master associated with this buffer is assigned a priority higher than the rest of the masters. An access by a high priority master will suspend any ongoing prefetch by another AHB master and will be serviced on high priority. Refer to Flexible AHB Buffers for details.
30–16 Reserved	This field is reserved.
15–8 ADATSZ	AHB data transfer size Defines the data transfer size in 8 bytes of an AHB triggered access to serial flash. For example, a value of 0x2 will set transfer size to 16 bytes. When ADATSZ = 0, the data size mentioned the sequence pointed to by the SEQID field overrides this value. Software should ensure that this transfer size is not greater than the size of this buffer.
7–4 Reserved	This field is reserved.
MSTRID	Master ID. The ID of the AHB master associated with BUFFER0. Any AHB access with this master port number is routed to this buffer. It must be ensured that the master IDs associated with all buffers must be different.

26.4.2.5 Buffer1 Configuration Register (QuadSPI_BUF1CR)

This register provides the configuration for any access to buffer1. An access is routed to buffer1 when the master port number of the incoming AHB request matches the MSTRID field of BUF1CR. Any buffer "miss" leads to the buffer being flushed and a serial flash transaction being triggered as per the sequence pointed to by the SEQID field.

Write:
- *QSPI_SR[AHB_ACC] = 0*

Address: 155_0000h base + 14h offset = 155_0014h

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																ADATSZ								Reserved				MSTRID			
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	*	*	*	*

* Notes:
- MSTRID field: See the module configuration for reset value.

QuadSPI_BUF1CR field descriptions

Field	Description
31–16 Reserved	This field is reserved.
15–8 ADATSZ	AHB data transfer size Defines the data transfer size in 8 bytes of an AHB triggered access to serial flash. For example, a value of 0x2 will set transfer size to 16 bytes. When ADATSZ = 0, the data size mentioned the sequence pointed to by the SEQID field overrides this value. Software should ensure that this transfer size is not greater than the size of this buffer.
7–4 Reserved	This field is reserved.
MSTRID	Master ID. The ID of the AHB master associated with BUFFER1. Any AHB access with this master port number is routed to this buffer. It must be ensured that the master IDs associated with all buffers must be different.

26.4.2.6 Buffer2 Configuration Register (QuadSPI_BUF2CR)

This register provides the configuration for any access to buffer2. An access is routed to buffer2 when the master port number of the incoming AHB request matches the MSTRID field of BUF2CR. Any buffer "miss" leads to the buffer being flushed and a serial flash transaction being triggered as per the sequence pointed to by the SEQID field.

Write:

- *QSPI_SR[AHB_ACC] = 0*

Address: 155_0000h base + 18h offset = 155_0018h

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																ADATSZ								Reserved				MSTRID			
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	*	*	*	*

* Notes:

- MSTRID field: See the module configuration for the device specific reset value.

QuadSPI_BUF2CR field descriptions

Field	Description
31–16 Reserved	This field is reserved. Reserved.
15–8 ADATSZ	AHB data transfer size Defines the data transfer size in 8 bytes of an AHB triggered access to serial flash. For example, a value of 0x2 will set transfer size to 16 bytes. When ADATSZ = 0, the data size mentioned the sequence pointed to by the SEQID field overrides this value. Software should ensure that this transfer size is not greater than the size of this buffer.
7–4 Reserved	This field is reserved. Reserved.

Table continues on the next page...

QuadSPI_BUF2CR field descriptions (continued)

Field	Description
MSTRID	Master ID. The ID of the AHB master associated with BUFFER2. Any AHB access with this master port number is routed to this buffer. It must be ensured that the master IDs associated with all buffers must be different.

26.4.2.7 Buffer3 Configuration Register (QuadSPI_BUF3CR)

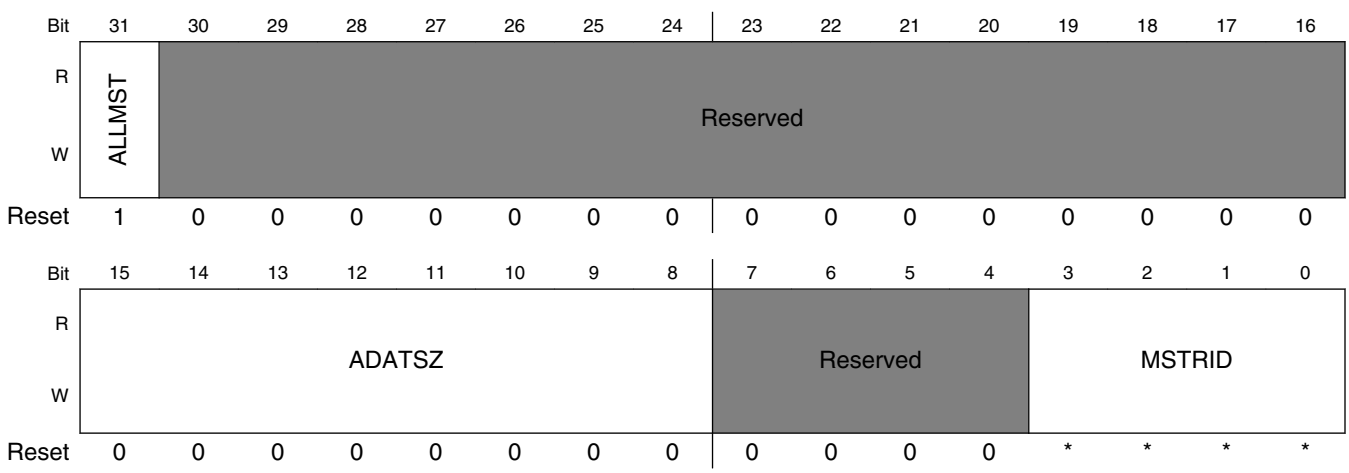
This register provides the configuration for any access to buffer3. An access is routed to buffer3 when the master port number of the incoming AHB request matches the MSTRID field of BUF3CR. Any buffer "miss" leads to the buffer being flushed a serial flash transaction being triggered as per the sequence pointed to by the SEQID field.

In the case that the ALLMST field is not set, any such transaction (where master port number does not match any of the MSTRID fields) will be considered as an illegal programming. This kind of programming is not allowed.

Write:

- *QSPI_SR[AHB_ACC] = 0*

Address: 155_0000h base + 1Ch offset = 155_001Ch



* Notes:

- MSTRID field: See the module configuration for the device specific reset value.

QuadSPI_BUF3CR field descriptions

Field	Description
31 ALLMST	All master enable. When set, buffer3 acts as an all-master buffer. Any AHB access with a master port number not matching with the master ID of buffer0 or buffer1 or buffer2 is routed to buffer3. When set, the MSTRID field of this register is ignored.

Table continues on the next page...

QuadSPI_BUF3CR field descriptions (continued)

Field	Description
30–16 Reserved	This field is reserved. Reserved.
15–8 ADATSZ	AHB data transfer size Defines the data transfer size in 8 Bytes of an AHB triggered access to serial flash. When ADATSZ = 0, the data size mentioned the sequence pointed to by the SEQID field overrides this value. Software should ensure that this transfer size is not greater than the size of this buffer.
7–4 Reserved	This field is reserved.
MSTRID	Master ID. The ID of the AHB master associated with BUFFER3. Any AHB access with this master port number is routed to this buffer. It must be ensured that the master IDs associated with all buffers must be different.

26.4.2.8 Buffer Generic Configuration Register (QuadSPI_BFGENCR)

This register provides the generic configuration to any of the buffer accesses. Any buffer "miss" leads to the buffer being flushed and a serial flash transaction being triggered as per the sequence pointed to by the SEQID field.

Write:

- *QSPI_SR[AHB_ACC] = 0*

Address: 155_0000h base + 20h offset = 155_0020h

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	0	0	0	0	0

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

QuadSPI_BFGENCR field descriptions

Field	Description
31–17 Reserved	This field is reserved.
16 Reserved	This field is reserved.

Table continues on the next page...

QuadSPI_BFGENCR field descriptions (continued)

Field	Description
15–12 SEQID	Points to a sequence in the Look-up-table. The SEQID defines the bits [6:2] of the LUT index. The bits [1:0] are always assumed to be 0. Refer to Look-up Table. **NOTE:** If the sequence pointer differs between the new and previous sequence then the user should reset this. See QSPI_SPTRCLR for more information.
Reserved	This field is reserved.

26.4.2.9 Buffer0 Top Index Register (QuadSPI_BUF0IND)

This register specifies the top index of buffer0, which defines its size. Note that that the 3 LSBs of this register are set to zero. This ensures that the buffer is 64-bit aligned, as each buffer entry is 64 bits long.

The register value should be set to the desired number of bytes. For example, setting BUF0IND[31:3] to 0 gives 0 bytes, 1 gives 8 bytes etc.

The size of buffer0 is the difference between BUF0IND and 0.

Software should ensure that BUF0IND value is not greater than the overall size of the buffer. The hardware does not provide any protection against illegal programming.

Write:

- *QSPI_SR[AHB_ACC] = 0*

Address: 155_0000h base + 30h offset = 155_0030h

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	TPINDX0																													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

QuadSPI_BUF0IND field descriptions

Field	Description
31–3 TPINDX0	Top index of buffer 0.
Reserved	This field is reserved. Reserved.

26.4.2.10 Buffer1 Top Index Register (QuadSPI_BUF1IND)

This register specifies the top index of buffer1, which defines its size. Note that the 3 LSBs of this register are set to zero. This ensures that the buffer is 64-bit aligned as each buffer entry is 64 bits long.

The size of buffer1 is the difference between BUF1IND and BUF0IND. The register value should be entered in bytes. For example, If BUF0IND = 0x100 then setting BUF1IND = 0x130 will set buffer1 size to 0x30 bytes.

It is the responsibility of the software to ensure that BUF1IND value is not greater than the overall size of the buffer. The hardware does not provide any protection against illegal programming.

Write:

- *QSPI_SR[AHB_ACC] = 0*

Address: 155_0000h base + 34h offset = 155_0034h

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	TPINDX1																													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

QuadSPI_BUF1IND field descriptions

Field	Description
31–3 TPINDX1	Top index of buffer 1.
Reserved	This field is reserved.

26.4.2.11 Buffer2 Top Index Register (QuadSPI_BUF2IND)

This register specifies the top index of buffer2, which defines its size. Note that that the 3 LSBs of this register are set to zero. This ensures that the buffer is 64-bit aligned as each buffer entry is 64 bits long.

The size of buffer2 is the difference between the BUF2IND and BUF1IND. The register value should be entered in bytes. For example, if BUF1IND = 0x130 then setting BUF2IND = 0x180 will set buffer2 size to 0x50 bytes.

It is the responsibility of the software to ensure that BUF2IND value is not greater than the overall size of the buffer. The hardware does not provide any protection against illegal programming.

Write:

- *QSPI_SR[AHB_ACC] = 0*

Address: 155_0000h base + 38h offset = 155_0038h

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	TPINDX2																													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

QuadSPI_BUF2IND field descriptions

Field	Description
31–3 TPINDX2	Top index of buffer 2.
Reserved	This field is reserved.

26.4.2.12 Serial Flash Address Register (QuadSPI_SFAR)

The module automatically translates this address on the memory map to the address on the flash itself. When operating in 24-bit mode, only bits 23-0 are sent to the flash. In 32-bit mode, bits 27-0 are used with bits 31-28 driven to 0 Refer to Table 26-16 for the mapping between the access mode and the QSPI_SFAR content and to Normal Mode for details about the command triggering and command execution. The software should ensure that the serial flash address provided in the QSPI_SFAR register lies in the valid flash address range as defined in Table 26-16.

Write:

- *QSPI_SR[IP_ACC] = 0*

Address: 155_0000h base + 100h offset = 155_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	SFADR																															
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

QuadSPI_SFAR field descriptions

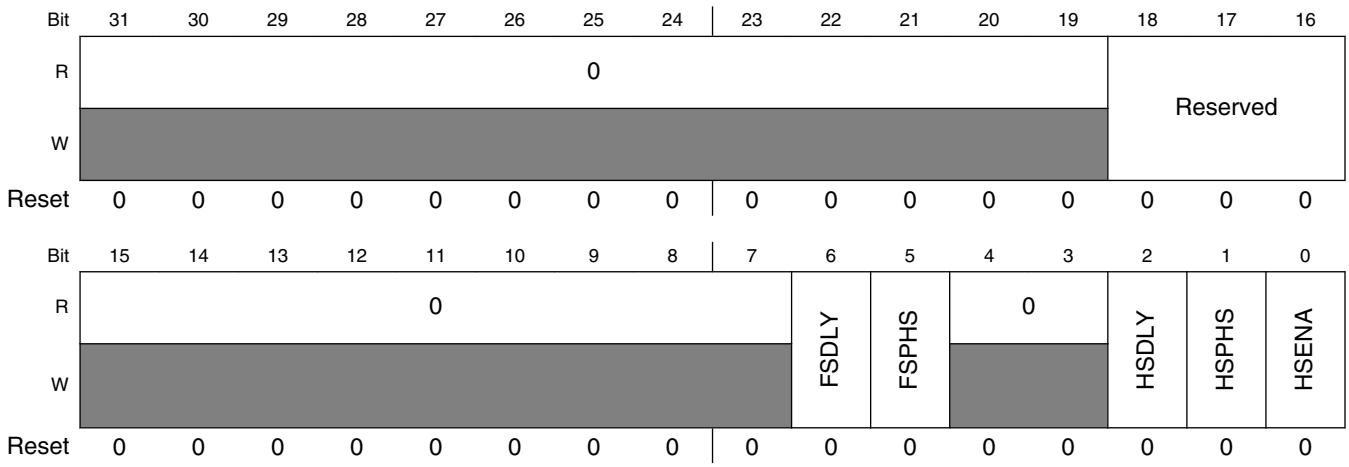
Field	Description
SFADR	Serial Flash Address. The register content is used as byte address for all following IP Commands.

26.4.2.13 Sampling Register (QuadSPI_SMPR)

The Sampling Register allows configuration of how the incoming data from the external serial flash devices are sampled in the QuadSPI module.

Write: Disabled Mode

Address: 155_0000h base + 108h offset = 155_0108h



QuadSPI_SMPR field descriptions

Field	Description
31–19 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
18–16 Reserved	This field is reserved.
15–7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 FSDLY	Full Speed Delay selection for SDR instructions. Select the delay with respect to the reference edge for the sample point valid for full speed commands. 0 One clock cycle delay 1 Two clock cycles delay. **NOTE:** Please refer to Supported read modes for its usage. It is ignored when using non-DQS DDR instructions.
5 FSPHS	Full Speed Phase selection for SDR instructions. Select the edge of the sampling clock valid for full speed commands. 0 Select sampling at non-inverted clock 1 Select sampling at inverted clock. **NOTE:** Please refer to Supported read modes for its usage. FSPHS programming is only supported with internal non-DQS SDR sampling. It must be '0' in other sampling modes.
4–3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 HSDLY	Half Speed Delay selection for SDR instructions. Only relevant when HSENA bit is set. Select the delay with respect to the reference edge for the sample point valid for half speed commands. 0 One clock cycle delay 1 Two clock cycle delay
1 HSPHS	Half Speed Phase selection for SDR instructions.

Table continues on the next page...

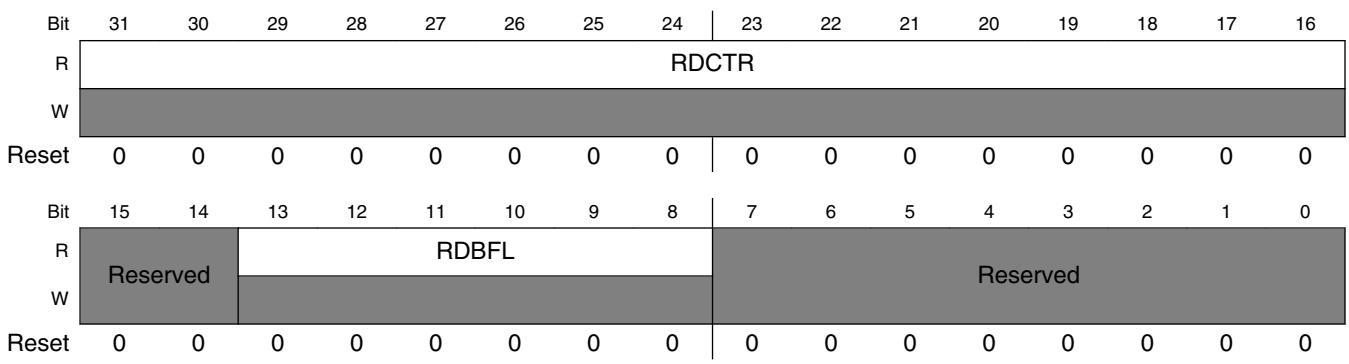
QuadSPI_SMPR field descriptions (continued)

Field	Description
	Only relevant when HSENA bit is set. Select the delay with respect to the reference edge for the sample point valid for half speed commands. 0 Select sampling at non-inverted clock 1 Select sampling at inverted clock
0 HSENA	Half Speed serial flash clock Enable This bit enables the divide by 2 of the clock to the external serial flash device for all commands, only in SDR. Refer to Serial Flash Clock Frequency Limitations for details. 0 Disable divide by 2 of serial flash clock for half speed commands 1 Enable divide by 2 of serial flash clock for half speed commands

26.4.2.14 RX Buffer Status Register (QuadSPI_RBSR)

This register contains information related to the receive data buffer.

Address: 155_0000h base + 10Ch offset = 155_010Ch



QuadSPI_RBSR field descriptions

Field	Description
31–16 RDCTR	Read Counter. Indicates how many entries of 4 bytes have been removed from the RX Buffer. For example, a value of 0x2 would indicate 8 bytes have been removed. It is incremented by the number (QSPI_RBCT[WMRK] + 1) on RX Buffer POP event. The RX Buffer can be popped using DMA or pop flag QSPI_FR[RBDF]. The QSPI_RSER[RBDDE] defines which pop has to be done. For further details, refer to AHB RX Data Buffer (QSPI_ARDB0 to QSPI_ARDB31) and "Data Transfer from the QuadSPI Module Internal Buffers" section in Flash Read.
15–14 Reserved	This field is reserved.
13–8 RDBFL	RX Buffer Fill Level. Indicates how many entries of 4 bytes are still available in the RX Buffer. For example, a value of 0x2 would indicate 8 bytes are available.
Reserved	This field is reserved.

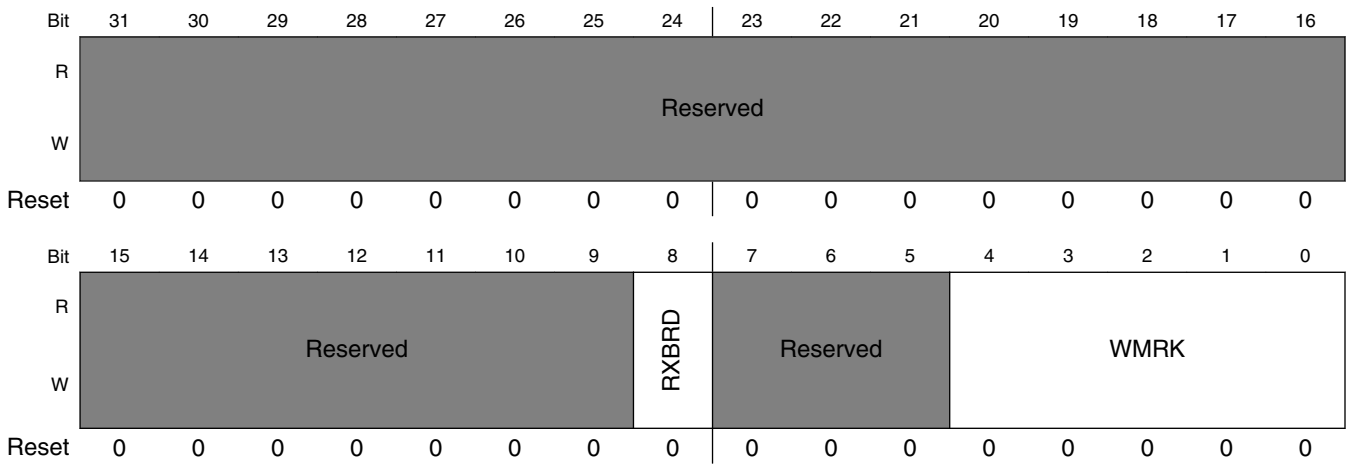
26.4.2.15 RX Buffer Control Register (QuadSPI_RBCT)

This register contains control data related to the receive data buffer.

Write:

- *QSPI_SR[IP_ACC] = 0*

Address: 155_0000h base + 110h offset = 155_0110h



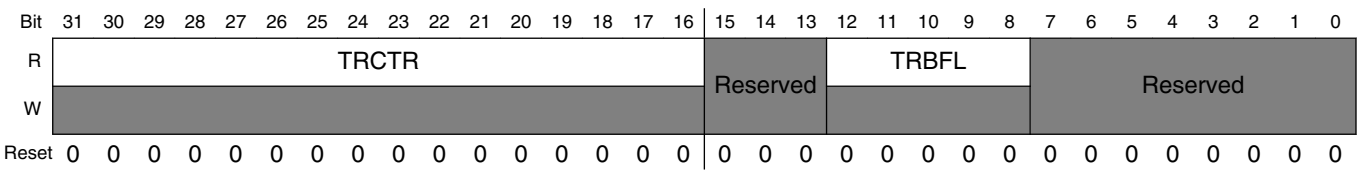
QuadSPI_RBCT field descriptions

Field	Description
31–9 Reserved	This field is reserved.
8 RXBRD	RX Buffer Readout. This field specifies the access scheme for the RX Buffer readout. 0 RX Buffer content is read using the AHB Bus registers QSPI_ARDB0 to QSPI_ARDB31. For details, refer to Exclusive Access to Serial Flash for AHB Commands. 1 RX Buffer content is read using the IP Bus registers QSPI_RBDR0 to QSPI_RBDR31.
7–5 Reserved	This field is reserved.
WMRK	RX Buffer Watermark. This field determines when the readout action of the RX Buffer is triggered. When the number of valid entries in the RX Buffer is equal to or greater than the number given by (WMRK+1) the QSPI_SR[RXWE] flag is asserted.The value should be entered as the number of 4-byte entries minus 1. For example, a value of 0x0 would set the watermark to 4 bytes, 1 to 8 bytes, 2 to 12 bytes, and so on. For details, refer to DMA Usage.

26.4.2.16 TX Buffer Status Register (QuadSPI_TBSR)

This register contains information related to the transmit data buffer.

Address: 155_0000h base + 150h offset = 155_0150h



QuadSPI_TBSR field descriptions

Field	Description
31–16 TRCTR	Transmit Counter. This field indicates how many entries of 4 bytes have been written into the TX Buffer by host accesses. It is reset to 0 when a 1 is written to QSPI_MCR[CLR_TXF]. It is incremented on each write access to the QSPI_TBDR register when another word has been pushed onto the TX Buffer. When it is not cleared the TRCTR field wraps around to 0. Refer to TX Buffer Data Register (QuadSPI_TBDR) for details.
15–13 Reserved	This field is reserved.
12–8 TRBFL	TX Buffer Fill Level. The TRBFL field contains the number of entries of 4 bytes each available in the TX Buffer for the QuadSPI module to transmit to the serial flash device.
Reserved	This field is reserved.

26.4.2.17 TX Buffer Data Register (QuadSPI_TBDR)

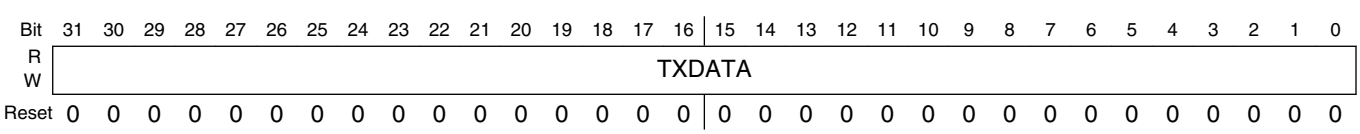
The QSPI_TBDR register provides access to the circular TX Buffer of depth 64 bytes . This buffer provides the data written into it as write data for the page programming commands to the serial flash device. Refer to Table 26-22 for the byte ordering scheme. A write transaction on the flash with data size of less than 32 bits will lead to the removal of one data entry from the TX buffer. The valid bits will be used and the rest of the bits will be discarded.

Write:

- *QSPI_SR[TXFULL] = 0*

32-bit write access required

Address: 155_0000h base + 154h offset = 155_0154h



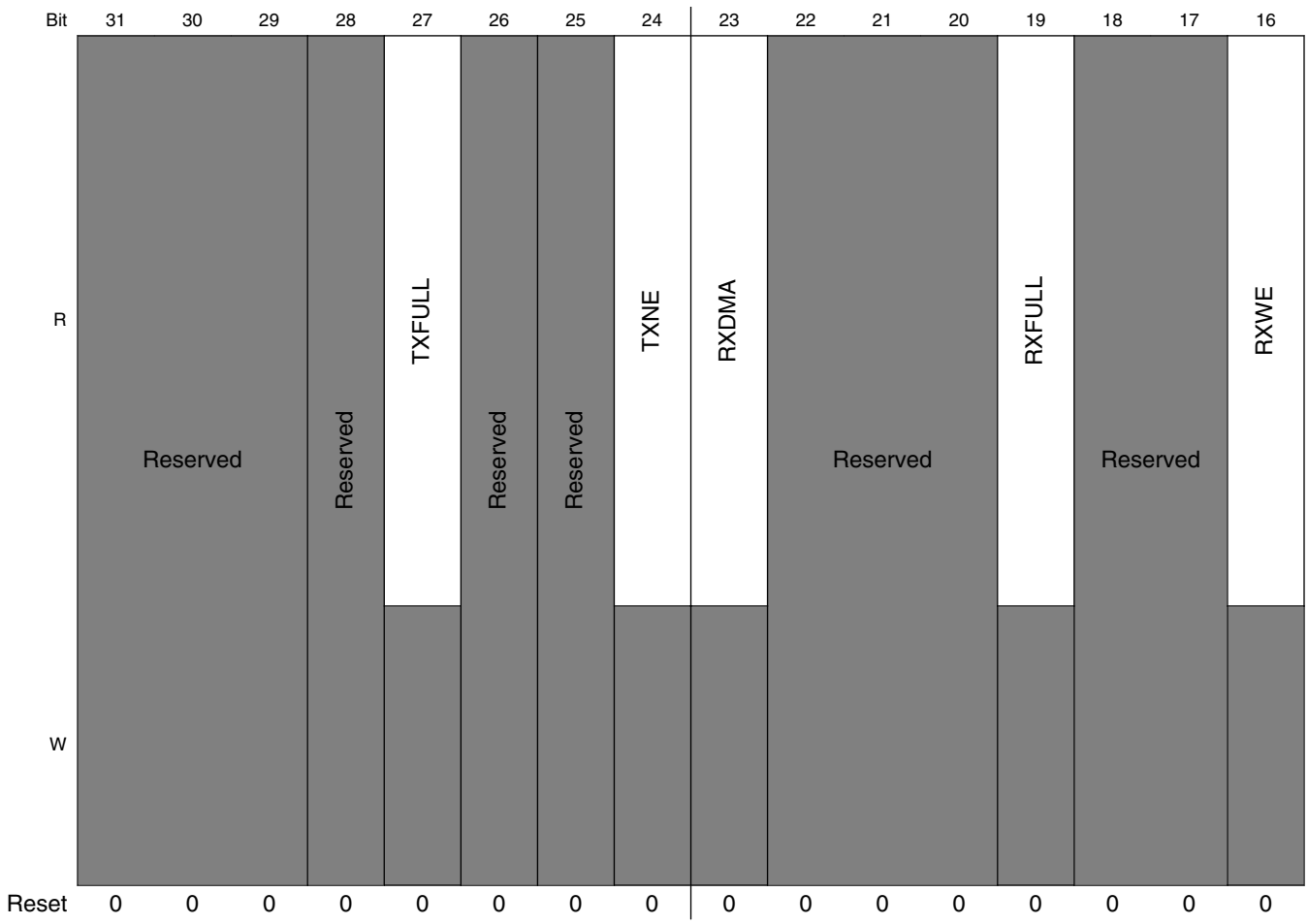
QuadSPI_TBDR field descriptions

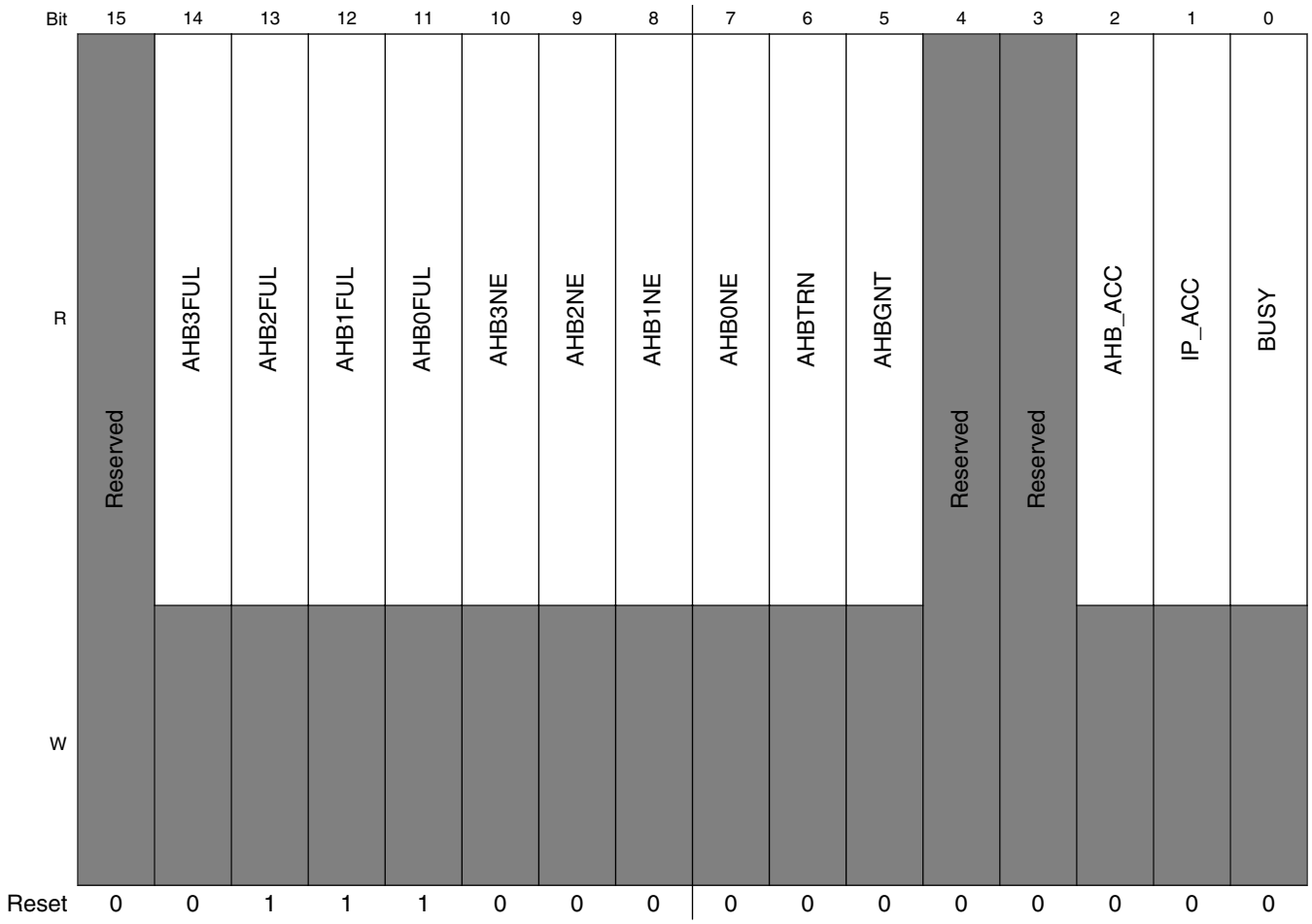
Field	Description
TXDATA	TX Data On write access the data is written into the next available entry of the TX Buffer and the QPSI_TBSR[TRBFL] field is updated accordingly. On a read access, the last data written to the register is returned.

26.4.2.18 Status Register (QuadSPI_SR)

The QSPI_SR register provides all available status information about SFM command execution and arbitration, the RX Buffer, TX Buffer, and the AHB Buffer.

Address: 155_0000h base + 15Ch offset = 155_015Ch





QuadSPI_SR field descriptions

Field	Description
31–29 Reserved	This field is reserved.
28 Reserved	This field is reserved.
27 TXFULL	TX Buffer Full. Asserted when no more data can be stored.
26 Reserved	This field is reserved.
25 Reserved	This field is reserved.
24 TXNE	TX Buffer Not Empty: Asserted when TX Buffer contains data.
23 RXDMA	RX Buffer DMA. Asserted when RX Buffer read out via DMA is active i.e DMA is requested or running.
22–20 Reserved	This field is reserved.
19 RXFULL	RX Buffer Full. Asserted when the RX Buffer is full, i.e. that QSPI_RBSR[RDBFL] field is equal to 32.

Table continues on the next page...

QuadSPI_SR field descriptions (continued)

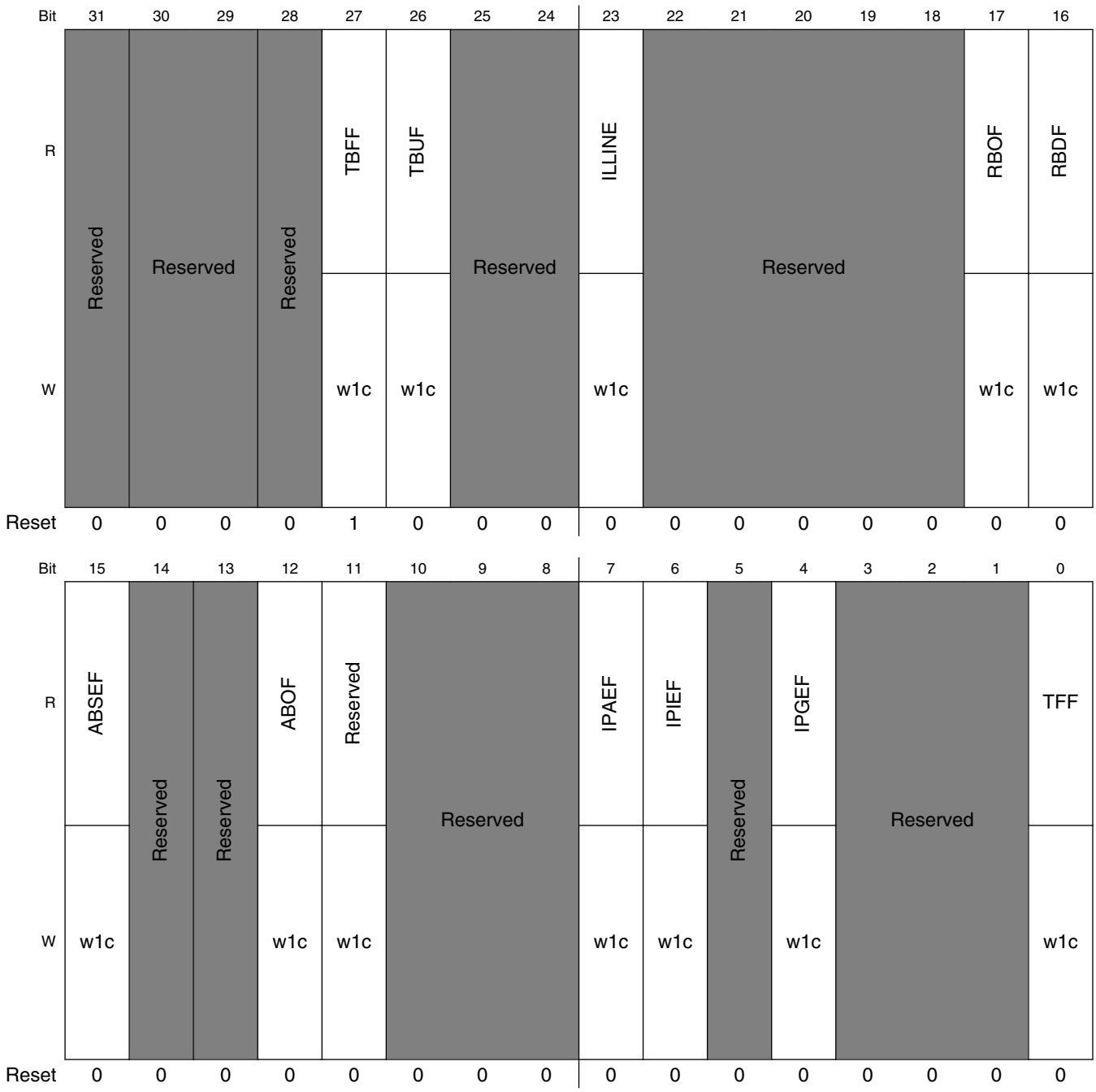
Field	Description
18–17 Reserved	This field is reserved.
16 RXWE	RX Buffer Watermark Exceeded. Asserted when the number of valid entries in the RX Buffer exceeds the number given in the QSPI_RBCT[WMRK] field.
15 Reserved	This field is reserved.
14 AHB3FUL	AHB 3 Buffer Full. Asserted when AHB 3 buffer is full.
13 AHB2FUL	AHB 2 Buffer Full. Asserted when AHB 2 buffer is full.
12 AHB1FUL	AHB 1 Buffer Full. Asserted when AHB 1 buffer is full.
11 AHB0FUL	AHB 0 Buffer Full. Asserted when AHB 0 buffer is full.
10 AHB3NE	AHB 3 Buffer Not Empty. Asserted when AHB 3 buffer contains data.
9 AHB2NE	AHB 2 Buffer Not Empty. Asserted when AHB 2 buffer contains data.
8 AHB1NE	AHB 1 Buffer Not Empty. Asserted when AHB 1 buffer contains data.
7 AHB0NE	AHB 0 Buffer Not Empty. Asserted when AHB 0 buffer contains data.
6 AHBTRN	AHB Access Transaction pending. Asserted when there is a pending request on the AHB interface. Refer to the AMBA specification for details.
5 AHBGNT	AHB Command priority Granted: Asserted when another module has been granted priority of AHB Commands against IP Commands. For details refer to Command Arbitration.
4 Reserved	This field is reserved.
3 RESERVED	This field is reserved.
2 AHB_ACC	AHB Access. Asserted when the transaction currently executed was initiated by AHB bus.
1 IP_ACC	IP Access. Asserted when transaction currently executed was initiated by IP bus.
0 BUSY	Module Busy. Asserted when module is currently busy handling a transaction to an external flash device.

26.4.2.19 Flag Register (QuadSPI_FR)

The QSPI_FR register provides all available flags about SFM command execution and arbitration which may serve as source for the generation of interrupt service requests. Note that the error flags in this register do not relate directly to the execution of the transaction in the serial flash device itself but only to the behavior and conditions visible in the QuadSPI module.

Write: Enabled Mode

Address: 155_0000h base + 160h offset = 155_0160h



QuadSPI_FR field descriptions

Field	Description
31 Reserved	This field is reserved.
30–29 RESERVED	This field is reserved.

Table continues on the next page...

QuadSPI_FR field descriptions (continued)

Field	Description
28 Reserved	This field is reserved.
27 TBFF	TX Buffer Fill Flag. Before writing to the TX buffer, this bit should be cleared. Then this bit has to be read back. If the bit is set, the TX Buffer can take more data. If the bit remains cleared, the TX buffer is full. Refer to Tx Buffer Operation for details.
26 TBUF	TX Buffer Underrun Flag. Set when the module tried to pull data although TX Buffer was empty. The IP Command leading to the TX Buffer underrun is continued (data sent to the serial flash device is undefined). Here valid 'underrun' means that, it should have occurred during the transacting such that few bytes (i.e., less than 4 bytes) are left in FIFO and remaining are filled with “FFFFh". Software should start TX transaction only when 128-bits are written in TX buffer. The application must clear the TX Buffer in response to this event by writing a '1' to the QSPI_MCR[CLR_TXF].
25–24 Reserved	This field is reserved.
23 ILLINE	Illegal Instruction Error Flag. Set when an illegal instruction is encountered by the controller in any of the sequences. As soon as this flag is set, user should assert QSPI_MCR[SWRSTSD] and QSPI_MCR[SWRSTHD] i.e., reset to flash and AHB domain after re-configuring the correct sequence instruction. Refer to Table 26-20 for a list of legal instructions.
22–18 Reserved	This field is reserved.
17 RBOF	RX Buffer Overflow Flag. Set when not all the data read from the serial flash device could be pushed into the RX Buffer. The IP Command leading to this condition is continued until the number of bytes according to the QSPI_IPCR[IDATSZ] field has been read from the serial flash device. The content of the RX Buffer is not changed.
16 RBDF	RX Buffer Drain Flag. Will be set if the QuadSPI_SR[RXWE] status bit is asserted. Writing 1 into this bit triggers one of the following actions: • If the RX Buffer has up to QuadSPI_RBCT[WMRK] valid entries then the flag is cleared. • If the RX Buffer has more than QuadSPI_RBCT[WMRK] valid entries and the QuadSPI_RSER[RBDDE] bit is not set (flag driven mode) a RX Buffer POP event is triggered. The flag remains set if the RX Buffer contains more than QuadSPI_RBCT[WMRK] valid entries after the RX Buffer POP event is finished. The flag is cleared if the RX Buffer contains less than or equal to QuadSPI_RBCT[WMRK] valid entries after the RX Buffer POP event is finished. Refer to "Receive Buffer Drain Interrupt or DMA Request" section in Normal Mode Interrupt and DMA Requests, for details.
15 ABSEF	AHB Sequence Error Flag. Set when the execution of an AHB Command is started with a WRITE Command in the sequence pointed to by the QSPI_BUFxCR register. (QSPI_BUFxCR implies any one of QSPI_BUF0CR/QSPI_BUF1CR/QSPI_BUF2CR/QSPI_BUF3CR.) Communication with the serial flash device is terminated before the execution of WRITE command by the QuadSPI module. The AHB bus request which triggered this command is answered with an ERROR response.
14 Reserved	Reserved This field is reserved.
13 Reserved	Reserved

Table continues on the next page...

QuadSPI_FR field descriptions (continued)

Field	Description
	This field is reserved.
12 ABOF	AHB Buffer Overflow Flag. Set when the size of the AHB access exceeds the size of the AHB buffer. This condition can occur only if the QSPI_BUFxCR[ADATSZ] field is programmed incorrectly. The AHB Command leading to this condition is continued until the number of entries according to the QSPI_BUFxCR[ADATSZ] field has been read from the serial flash device. The content of the AHB Buffer is not changed.
11 Reserved	This field is reserved.
10–8 Reserved	This field is reserved.
7 IPAEF	IP Command Trigger during AHB Access Error Flag. Set when the following condition occurs: • A write access occurs to the QSPI_IPCR[SEQID] field and the QSPI_SR[AHB_ACC] bit is set. Any command leading to the assertion of the IPAEF flag is ignored.
6 IPIEF	IP Command Trigger could not be executed Error Flag. Set when the QSPI_SR[IP_ACC] bit is set (i.e. an IP triggered command is currently executing) and any of the following conditions occurs: • Write access to the QSPI_IPCR register. Any command leading to the assertion of the IPIEF flag is ignored • Write access to the QSPI_SFAR register. • Write access to the QSPI_RBCT register.
5 Reserved	This field is reserved.
4 IPGEF	IP Command Trigger during AHB Grant Error Flag. Set when the following condition occurs: • A write access occurs to the QSPI_IPCR[SEQID] field and the QSPI_SR[AHBGNT] bit is set. Any command leading to the assertion of the IPGEF flag is ignored.
3–1 Reserved	This field is reserved.
0 TFF	IP Command Transaction Finished Flag. Set when the QuadSPI module has finished a running IP Command. If an error occurred the related error flags are valid, at the latest, in the same clock cycle when the TFF flag is asserted.

26.4.2.20 Interrupt and DMA Request Select and Enable Register (QuadSPI_RSER)

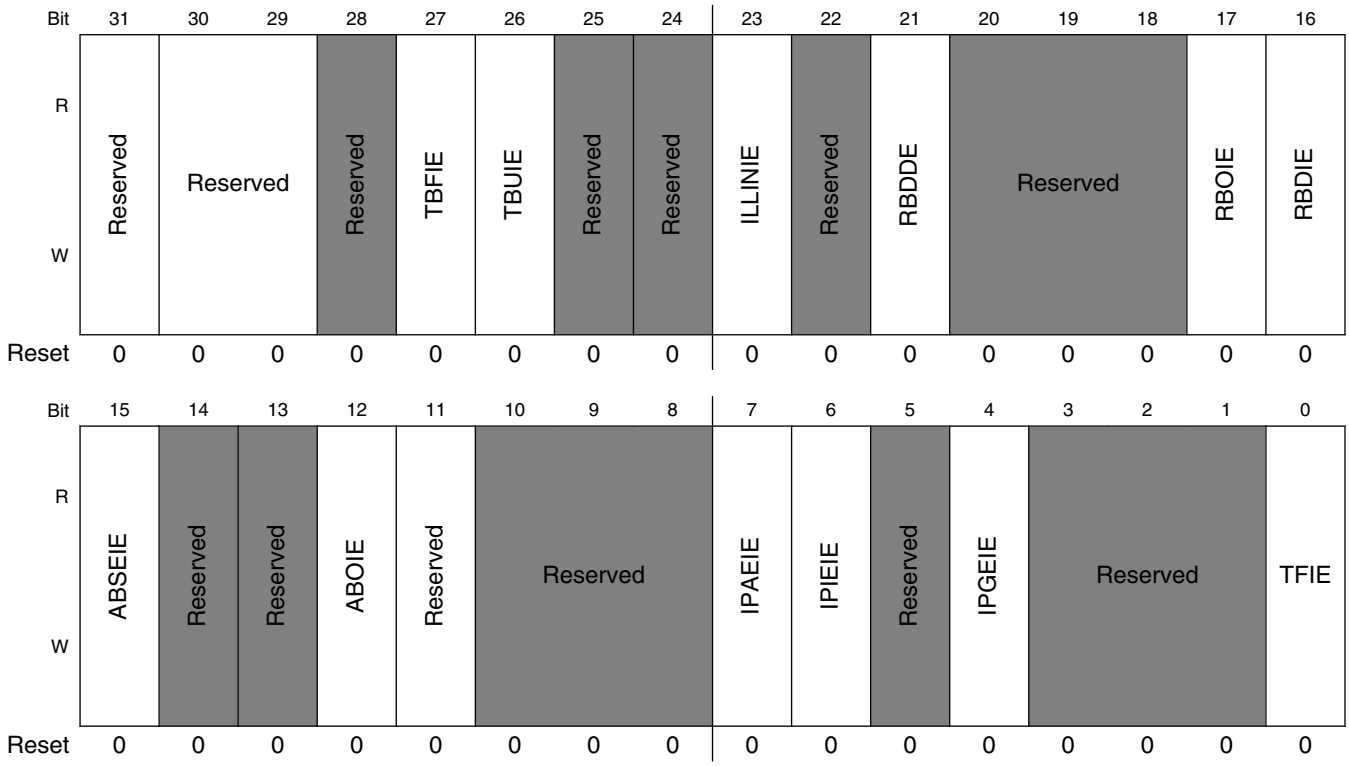
The QuadSPI_RSER register provides enables and selectors for the interrupts in the QuadSPI module.

NOTE

Each flag of the QuadSPI_FR register enabled as source for an interrupt prevents the QuadSPI module from entering Stop Mode or Module Disable Mode when this flag is set.

Write: Anytime

Address: 155_0000h base + 164h offset = 155_0164h



QuadSPI_RSER field descriptions

Field	Description
31 Reserved	Reserved This field is reserved.
30–29 Reserved	This field is reserved.
28 Reserved	This field is reserved.
27 TBFIE	TX Buffer Fill Interrupt Enable 0 No TBFF interrupt will be generated 1 TBFF interrupt will be generated
26 TBUIE	TX Buffer Underrun Interrupt Enable 0 No TBUF interrupt will be generated 1 TBUF interrupt will be generated
25 Reserved	This field is reserved.
24 Reserved	This field is reserved.
23 ILLINIE	Illegal Instruction Error Interrupt Enable. Triggered by ILLINE flag in QSPI_FR 0 No ILLINE interrupt will be generated 1 ILLINE interrupt will be generated

Table continues on the next page...

QuadSPI_RSER field descriptions (continued)

Field	Description
22 Reserved	This field is reserved.
21 RBDDE	RX Buffer Drain DMA Enable: Enables generation of DMA requests for RX Buffer Drain. When this bit is set DMA requests are generated as long as the QSPI_SR[RXWE] status bit is set. 0 No DMA request will be generated 1 DMA request will be generated
20–18 Reserved	This field is reserved.
17 RBOIE	RX Buffer Overflow Interrupt Enable 0 No RBOF interrupt will be generated 1 RBOF interrupt will be generated
16 RBDIE	RX Buffer Drain Interrupt Enable: Enables generation of IRQ requests for RX Buffer Drain. When this bit is set the interrupt is asserted as long as the QuadSPI_SR[RBDF] flag is set. 0 No RBDF interrupt will be generated 1 RBDF Interrupt will be generated
15 ABSEIE	AHB Sequence Error Interrupt Enable: Triggered by ABSEF flags of QSPI_FR 0 No ABSEF interrupt will be generated 1 ABSEF interrupt will be generated
14 Reserved	Reserved This field is reserved.
13 Reserved	Reserved This field is reserved.
12 ABOIE	AHB Buffer Overflow Interrupt Enable 0 No ABOF interrupt will be generated 1 ABOF interrupt will be generated
11 Reserved	This field is reserved.
10–8 Reserved	This field is reserved.
7 IPAEIE	IP Command Trigger during AHB Access Error Interrupt Enable 0 No IPAEF interrupt will be generated 1 IPAEF interrupt will be generated
6 IPIEIE	IP Command Trigger during IP Access Error Interrupt Enable 0 No IPIEF interrupt will be generated 1 IPIEF interrupt will be generated
5 Reserved	This field is reserved.
4 IPGEIE	IP Command Trigger during AHB Grant Error Interrupt Enable

Table continues on the next page...

QuadSPI_RSER field descriptions (continued)

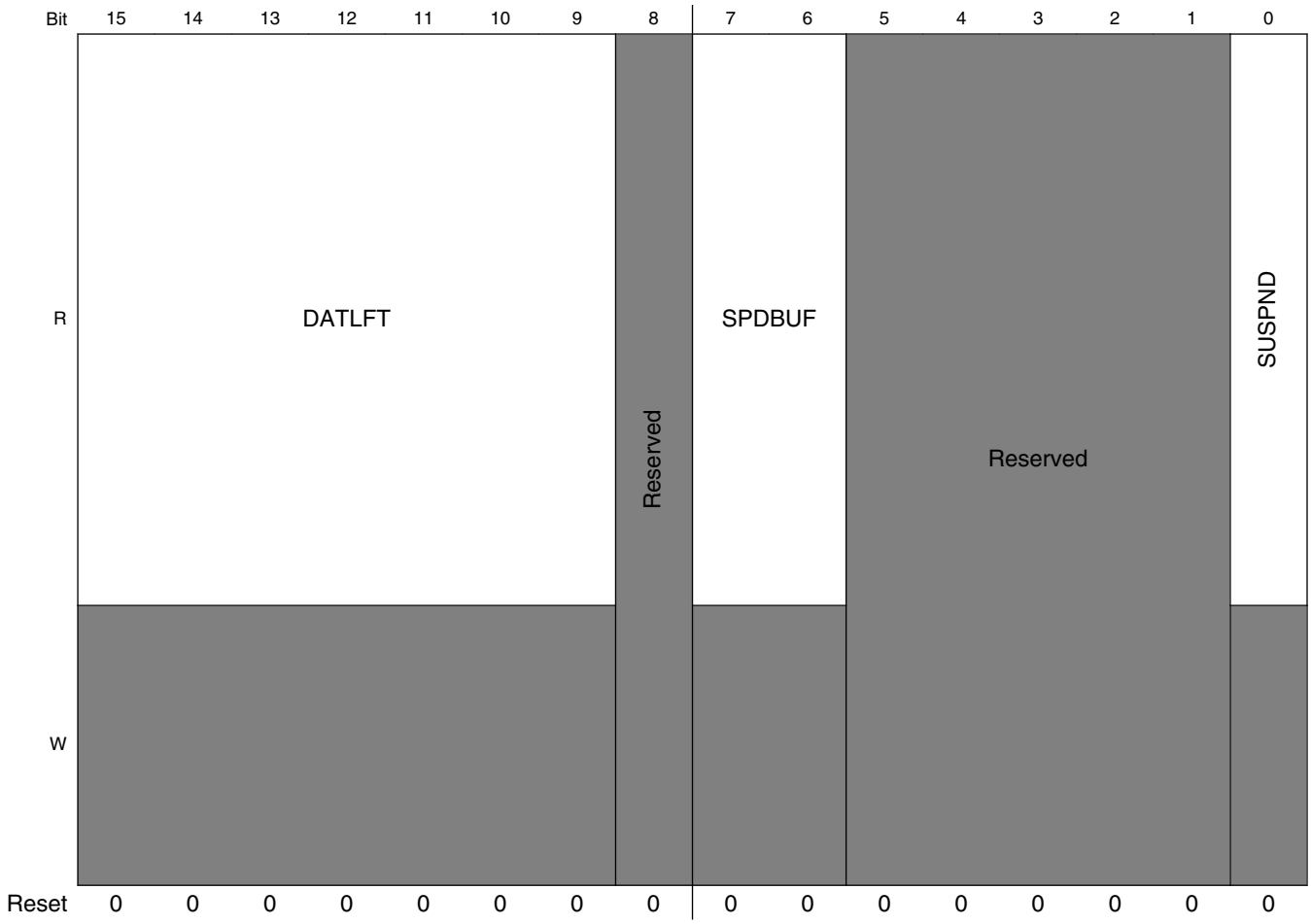
Field	Description
	0 No IPGEF interrupt will be generated 1 IPGEF interrupt will be generated
3–1 Reserved	This field is reserved. Reserved.
0 TFIE	Transaction Finished Interrupt Enable 0 No TFF interrupt will be generated 1 TFF interrupt will be generated

26.4.2.21 Sequence Suspend Status Register (QuadSPI_SPNDST)

The sequence suspend status register provides information specific to any suspended sequence. An AHB sequence may be suspended when a high priority AHB master makes an access before the AHB sequence completes the data transfer requested.

Address: 155_0000h base + 168h offset = 155_0168h

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



QuadSPI_SPNDST field descriptions

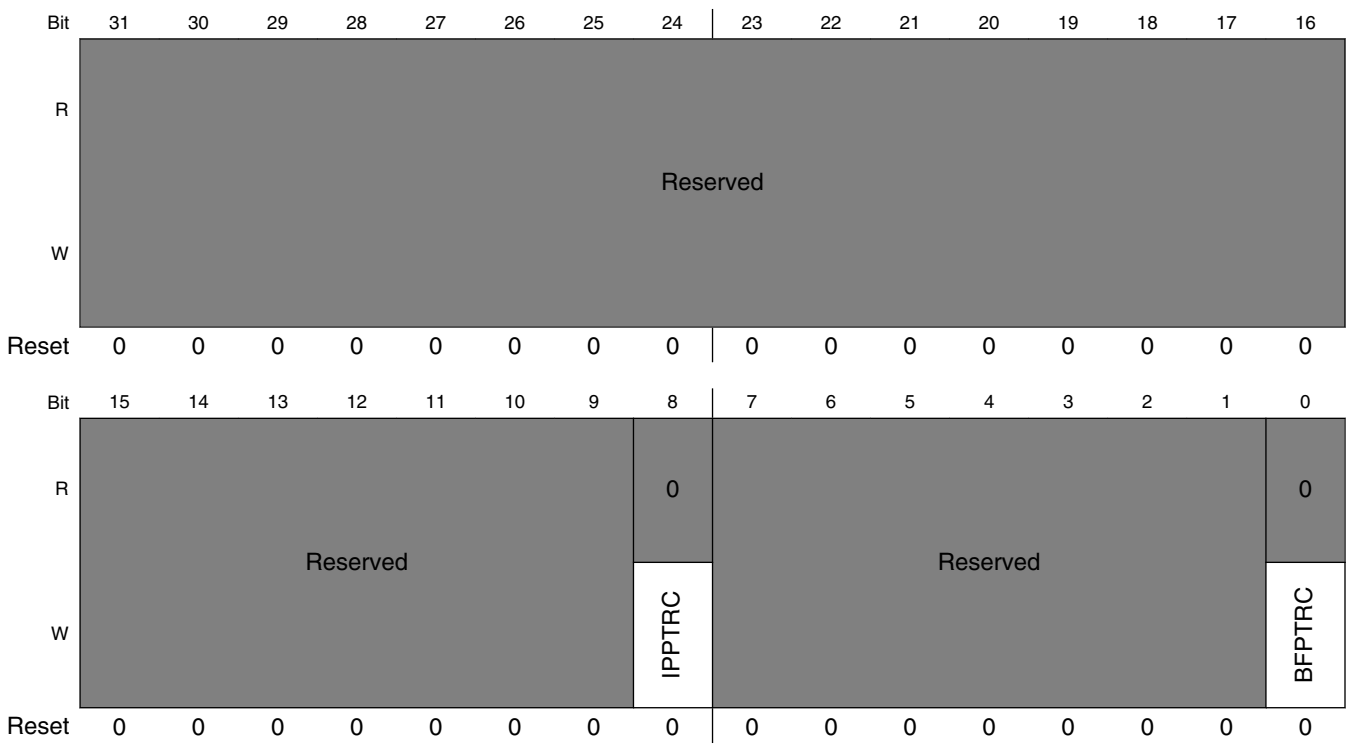
Field	Description
31–16 Reserved	This field is reserved.
15–9 DATLFT	Data left: Provides information about the amount of data left to be read in the suspended sequence. Valid only when SUSPND is set to 1'b1. Value in terms of 64 bits or 8 bytes
8 Reserved	This field is reserved.
7–6 SPDBUF	Suspended Buffer: Provides the suspended buffer number. Valid only when SUSPND is set to 1'b1
5–1 Reserved	This field is reserved.
0 SUSPND	When set, it signifies that a sequence is in suspended state

26.4.2.22 Sequence Pointer Clear Register (QuadSPI_SPTRCLR)

The sequence pointer clear register provides bits to reset the IP and Buffer sequence pointers. The sequence pointer contains the index of which instruction within the LUT entry is to be executed next. For example, if the LUT entry ends on a JMP_ON_CS value of 2, the index will be stored as 2.

The software should reset the sequence pointers whenever the sequence ID is changed by updating the SEQID field in QSPI_IPCR or QSPI_BFGENCR.

Address: 155_0000h base + 16Ch offset = 155_016Ch



QuadSPI_SPTRCLR field descriptions

Field	Description
31–9 Reserved	This field is reserved.
8 IPPTRC	IP Pointer Clear: 1: Clears the sequence pointer for IP accesses as defined in QuadSPI_IPCR This is a self-clearing field. This bit should be programmed two times to clear the sequence pointers.
7–1 Reserved	This field is reserved. Reserved.
0 BFPTRC	Buffer Pointer Clear: 1: Clears the sequence pointer for AHB accesses as defined in QuadSPI_BFGENCR. This is a self-clearing field. This bit should be programmed two times to clear the sequence pointers.

26.4.2.23 Serial Flash A1 Top Address (QuadSPI_SFA1AD)

The QSPI_SFA1AD register provides the address mapping for the serial flash A1.The difference between QSPI_SFA1AD[TPADA1] and QSPI_AMBA_BASE defines the size of the memory map for serial flash A1.

Write:

- *QSPI_SR[IP_ACC] = 0*
- *QSPI_SR[AHB_ACC] = 0*

Address: 155_0000h base + 180h offset = 155_0180h

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	TPADA1																						Reserved									
Reset	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	0	0	0	0	0	0	0	0	0	0

* Notes:

- TPADA1 field: See the module configuration for the device specific reset values.

QuadSPI_SFA1AD field descriptions

Field	Description
31–10 TPADA1	Top address for Serial Flash A1. In effect, TPADxx is the first location of the next memory.
Reserved	This field is reserved.

26.4.2.24 Serial Flash A2 Top Address (QuadSPI_SFA2AD)

The QSPI_SFA2AD register provides the address mapping for the serial flash A2.The difference between QSPI_SFA2AD[TPADA2] and QSPI_SFA1AD[TPADA1] defines the size of the memory map for serial flash A2.

Write:

- *QSPI_SR[IP_ACC] = 0*
- *QSPI_SR[AHB_ACC] = 0*

Address: 155_0000h base + 184h offset = 155_0184h

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	TPADA2																						Reserved									
Reset	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	0	0	0	0	0	0	0	0	0	0

* Notes:

- TPADA2 field: See the module configuration for the device specific reset values.

QuadSPI_SFA2AD field descriptions

Field	Description
31–10 TPADA2	Top address for Serial Flash A2. In effect, TPxxAD is the first location of the next memory.
Reserved	This field is reserved.

26.4.2.25 RX Buffer Data Register (QuadSPI_RBDR*n*)

The QuadSPI_RBDR registers provide access to the individual entries in the RX Buffer. Refer to Table 26-22 for the byte ordering scheme.

QuadSPI_RBDR0 corresponds to the actual position of the read pointer within the RX Buffer. The number of valid entries available depends from the number of RX Buffer entries implemented and from the number of valid buffer entries available in the RX Buffer.

Example 1, RX Buffer filled completely with 32 words: In this case the address range for valid read access extends from QuadSPI_RBDR0 to QuadSPI_RBDR31.

Example 2, RX Buffer filled with 5 valid words: RX Buffer fill level QuadSPI_RBSR[RDBFL] is 5. In this case an access to QuadSPI_RBDR4 provides the last valid entry.

Any access beyond the range of valid RX Buffer entries provides undefined results.

Address: 155_0000h base + 200h 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	RXDATA																															
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

QuadSPI_RBDR*n* field descriptions

Field	Description
RXDATA	RX Data. The RXDATA field contains the data associated with the related RX Buffer entry. Data format and byte ordering is given in Byte Ordering of Serial Flash Read Data .

26.4.2.26 LUT Key Register (QuadSPI_LUTKEY)

The LUT Key register contains the key to lock and unlock the Look-up-table. Refer to Look-up Table for details.

Write: Anytime

Address: 155_0000h base + 300h offset = 155_0300h

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	KEY																															
Reset	0	1	0	1	1	0	1	0	1	1	1	1	0	0	0	0	0	1	0	1	1	0	1	0	1	1	1	1	0	0	0	0

QuadSPI_LUTKEY field descriptions

Field	Description
KEY	The key to lock or unlock the LUT. The KEY is 0x5AF05AF0. The read value is always 0x5AF05AF0

26.4.2.27 LUT Lock Configuration Register (QuadSPI_LCKCR)

The LUT lock configuration register is used along with QSPI_LUTKEY register to lock or unlock the LUT. This register has to be written immediately after QSPI_LUTKEY register for the lock or unlock operation to be successful. Refer to Look-up Table for details. Setting both the LOCK and UNLOCK bits as "00" or "11" is not allowed.

Write: Just after writing the LUT Key Register

(QSPI_LUTKEY)

Address: 155_0000h base + 304h offset = 155_0304h

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														UNLOCK	LOCK
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

QuadSPI_LCKCR field descriptions

Field	Description
31–2 Reserved	This field is reserved.
1 UNLOCK	Unlocks the LUT when the following two conditions are met: 1. This register is written just after the LUT Key Register (QuadSPI_LUTKEY) 2. The LUT key register was written with 0x5AF05AF0 key

Table continues on the next page...

QuadSPI_LCKCR field descriptions (continued)

Field	Description
0 LOCK	Locks the LUT when the following condition is met: 1. This register is written just after the LUT Key Register (QuadSPI_LUTKEY) 2. The LUT key register was written with 0x5AF05AF0 key

26.4.2.28 Look-up Table register (QuadSPI_LUT*n*)

A sequence of instruction-operand pairs may be pre-populated in the LUT according to the device connected on board. Each instruction-operand pair is of 16 bits (2 bytes) each. Every sequence pre-programmed by Program Sequence Engine in the LUT is referred to by its index. The LUT registers are a look-up-table for sequences of instructions. The programmable sequence engine executes the instructions in these sequences to generate a valid serial flash transaction. There are a total of 64 LUT registers. These 64 registers are divided into groups of 4 registers that make a valid sequence. Therefore, QSPI_LUT[0], QSPI_LUT[4], QSPI_LUT[8] ..... QSPI_LUT[60] are the starting registers of a valid sequence. Each of these sets of 4 registers can have a maximum of 8 instructions. Reset value of the register shown below is only applicable to LUT2 to LUT63. A maximum of 16 sequences can be defined at one time. Look-up Table describes the LUT registers in detail.

NOTE

The reset values for LUT0 and LUT1 are 0818_0403h and 2400_1C08h, respectively.

Write: Once the LUT is unlocked

Address: 155_0000h base + 310h offset + (4d × i), where i=0d to 63d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	INSTR1						PAD1		OPRND1							
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	INSTR0						PAD0		OPRND0							
Reset	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

* Notes:
- The reset values for LUT0 and LUT1 are 0818_0403h and 2400_1C08h respectively.

QuadSPI_LUT*n* field descriptions

Field	Description
31–26 INSTR1	Instruction 1

Table continues on the next page...

QuadSPI_LUT*n* field descriptions (continued)

Field	Description
25–24 PAD1	Pad information for INSTR1. 00 1 Pad 01 2 Pads 10 4 Pads 11 NA
23–16 OPRND1	Operand for INSTR1.
15–10 INSTR0	Instruction 0
9–8 PAD0	Pad information for INSTR0. 00 1 Pad 01 2 Pads 10 4 Pads 11 NA
OPRND0	Operand for INSTR0.

26.4.3 Serial Flash Address Assignment

The serial flash address assignment may be modified by writing into Serial Flash A1 Top Address (QuadSPI_SFA1AD) and Serial Flash A2 Top Address (QuadSPI_SFA2AD) for device A . The following table shows how different access modes are related to the address specified for the next SFM Command. Note that this address assignment is valid for both IP and AHB commands.

Table 26-16. Serial Flash Address Assignment

Parameter	Function	Access Mode
QSPI_AMBA_BASE ((31:10) - 22 bits)	QuadSPI AHB base address	
TOP_ADDR_MEMA1(TPADA1)	Top address for the external flash A1 (the first of the two independent flashes sharing the IOFA)	Any access to the address space between TOP_ADDR_MEMA1 and QSPI_AMBA_BASE will be routed to **Serial Flash A1**
TOP_ADDR_MEMA2(TPADA2)	Top address for the external flash A2 (the second of the two independent flashes sharing the IOFA).	Any access to the address space between TOP_ADDR_MEMA2 and TOP_ADDR_MEMA1 will be routed to **Serial Flash A2**

26.5 Flash memory mapped AMBA bus

QSPI_AMBA_BASE defines the address to be used as start address of the serial flash device as defined by the system memory map. Note that this may be a remapping of the physical address of the serial flash in the system. Refer to the system address map. Note that this may be a remapping of the physical address of the serial flash in the system. Refer to the system address map.

Table 26-17. QuadSPI AMBA Bus Memory Map

Address	Register Name
Memory Mapped Serial Flash Data - Individual Flash Mode on Flash A	
QSPI_AMBA_BASE to (TOP_ADDR_MEMA2 - 0x01)	Memory Mapped Serial Flash Data - Individual Flash Mode on Flash A Refer to Memory Mapped Serial Flash Data - Individual Flash Mode on Flash A for details and to Table 26-22 and Table 26-23 for information about the byte ordering.
AHB RX Data Buffer (QSPI_ARDB0 to QSPI_ARDB31)	
QSPI_ARDB_BASE to… (32 * 4 Byte) QSPI_ARDB_BASE + 0x0000_01FF	AHB RX Data Buffer (QSPI_ARDB0 to QSPI_ARDB31) Refer to Table 26-22 for information about the byte ordering.

Note

Any read access to non-implemented addresses will provide undefined results.

In 'Individual Flash Modes', the 3/4 address bytes (as programmed in the instruction/operand in the sequence) available for the flash address is determined by SFADR[8:31] or SFADR[0:31] as given in the table above.

26.5.1 AHB Bus Access Considerations

It has to be noted that all logic in the QuadSPI module implementing the AHB Bus access is designed to read the content of an external serial flash device. Therefore, the following restrictions apply to the QuadSPI module with respect to accesses to the AHB bus.

- At present, the QuadSPI does not support AHB writes so any write access is answered with the ERROR condition according to the AMBA AHB Specification. No write occurs.

- Any AHB Command resulting in the assertion of the QSPI_FR[ABSEF] flag is answered with the ERROR condition according to the AMBA_AHB specification. The resulting AHB Command is ignored.
- AHB Bus access types fully supported are NONSEQ and BUSY.
- AHB access type SEQ is treated in the same way like NONSEQ. Refer to the AMBA AHB Specification for further details.

26.5.2 Memory Mapped Serial Flash Data - Individual Flash Mode on Flash A

Starting with address QSPI_AMBA_BASE the content of the first external serial flash devices is mapped into the address space of the device containing the QuadSPI module. Serial flash address byte address 0x0 corresponds to bus address QSPI_AMBA_BASE with increasing order. Refer to the following table for the address mapping. The byte ordering for 32 bit access is given in Table 26-22 and for 64 bit read access the byte ordering is given in Table 26-23.

Table 26-18. Memory Mapped Individual Flash Mode - Flash A Address Scheme

Memory Mapped Address 32 Bit Access	Memory Mapped Address 64 Bit Access	Serial Flash Byte Address	Flash Device
QSPI_AMBA_BASE + 0x00	QSPI_AMBA_BASE + 0x00	0x00_0000 to 0x00_0003	A1
QSPI_AMBA_BASE + 0x04		0x00_0004 to 0x00_0007	
…	…	…	
TOP_ADDR_MEMA1 - 0x08	TOP_ADDR_MEMA1 - 0x08	(TOP_ADDR_MEMA1- 0x08) to (TOP_ADDR_MEMA1 - 0x04 -0x01)	
TOP_ADDR_MEMA1 - 0x04		(TOP_ADDR_MEMA1 - 0x04) to (TOP_ADDR_MEMA1 - 0x01)	
TOP_ADDR_MEMA1 + 0x00	TOP_ADDR_MEMA1 + 0x00_0000	0x00_0000 to 0x00_0003	A2
TOP_ADDR_MEMA1 + 0x04		0x00_0004 to 0x00_0007	
…..	…	…	
TOP_ADDR_MEMA2 - 0x08	TOP_ADDR_MEMA2 - 0x08	(TOP_ADDR_MEMA2 - 0x08) to (TOP_ADDR_MEMA2 - 0x04 - 0x01)	
TOP_ADDR_MEMA2 - 0x04		(TOP_ADDR_MEMA2 - 0x04) to (TOP_ADDR_MEMA2 - 0x01)	

The available address range depends from the size of the external serial flash device. Any access beyond the size of the external serial flash provides undefined results.

For details concerning the read process refer to Flash Read.

26.5.3 AHB RX Data Buffer (QSPI_ARDB0 to QSPI_ARDB31)

NOTE

See the System Memory map in this document for the base address of the QSPI AHB RX Data Buffer.

memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	AHB RX Data Buffer register (ARDB0)	32	R/W	0000_0000h	26.5.3.1/ 1476
4	AHB RX Data Buffer register (ARDB1)	32	R/W	0000_0000h	26.5.3.1/ 1476
8	AHB RX Data Buffer register (ARDB2)	32	R/W	0000_0000h	26.5.3.1/ 1476
C	AHB RX Data Buffer register (ARDB3)	32	R/W	0000_0000h	26.5.3.1/ 1476
10	AHB RX Data Buffer register (ARDB4)	32	R/W	0000_0000h	26.5.3.1/ 1476
14	AHB RX Data Buffer register (ARDB5)	32	R/W	0000_0000h	26.5.3.1/ 1476
18	AHB RX Data Buffer register (ARDB6)	32	R/W	0000_0000h	26.5.3.1/ 1476
1C	AHB RX Data Buffer register (ARDB7)	32	R/W	0000_0000h	26.5.3.1/ 1476
20	AHB RX Data Buffer register (ARDB8)	32	R/W	0000_0000h	26.5.3.1/ 1476
24	AHB RX Data Buffer register (ARDB9)	32	R/W	0000_0000h	26.5.3.1/ 1476
28	AHB RX Data Buffer register (ARDB10)	32	R/W	0000_0000h	26.5.3.1/ 1476
2C	AHB RX Data Buffer register (ARDB11)	32	R/W	0000_0000h	26.5.3.1/ 1476
30	AHB RX Data Buffer register (ARDB12)	32	R/W	0000_0000h	26.5.3.1/ 1476
34	AHB RX Data Buffer register (ARDB13)	32	R/W	0000_0000h	26.5.3.1/ 1476
38	AHB RX Data Buffer register (ARDB14)	32	R/W	0000_0000h	26.5.3.1/ 1476
3C	AHB RX Data Buffer register (ARDB15)	32	R/W	0000_0000h	26.5.3.1/ 1476
40	AHB RX Data Buffer register (ARDB16)	32	R/W	0000_0000h	26.5.3.1/ 1476

Table continues on the next page...

memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
44	AHB RX Data Buffer register (ARDB17)	32	R/W	0000_0000h	26.5.3.1/ 1476
48	AHB RX Data Buffer register (ARDB18)	32	R/W	0000_0000h	26.5.3.1/ 1476
4C	AHB RX Data Buffer register (ARDB19)	32	R/W	0000_0000h	26.5.3.1/ 1476
50	AHB RX Data Buffer register (ARDB20)	32	R/W	0000_0000h	26.5.3.1/ 1476
54	AHB RX Data Buffer register (ARDB21)	32	R/W	0000_0000h	26.5.3.1/ 1476
58	AHB RX Data Buffer register (ARDB22)	32	R/W	0000_0000h	26.5.3.1/ 1476
5C	AHB RX Data Buffer register (ARDB23)	32	R/W	0000_0000h	26.5.3.1/ 1476
60	AHB RX Data Buffer register (ARDB24)	32	R/W	0000_0000h	26.5.3.1/ 1476
64	AHB RX Data Buffer register (ARDB25)	32	R/W	0000_0000h	26.5.3.1/ 1476
68	AHB RX Data Buffer register (ARDB26)	32	R/W	0000_0000h	26.5.3.1/ 1476
6C	AHB RX Data Buffer register (ARDB27)	32	R/W	0000_0000h	26.5.3.1/ 1476
70	AHB RX Data Buffer register (ARDB28)	32	R/W	0000_0000h	26.5.3.1/ 1476
74	AHB RX Data Buffer register (ARDB29)	32	R/W	0000_0000h	26.5.3.1/ 1476
78	AHB RX Data Buffer register (ARDB30)	32	R/W	0000_0000h	26.5.3.1/ 1476
7C	AHB RX Data Buffer register (ARDB31)	32	R/W	0000_0000h	26.5.3.1/ 1476

26.5.3.1 AHB RX Data Buffer register (ARDB*n*)

The AHB RX Data Buffer register 0 to 31 can be used to read the buffer content of the RX Buffer from successive addresses. QSPI_ARDB0 corresponds to the RX Buffer register entry corresponding to the current value of the read pointer with increasing order.

The increment of the read pointer depends from the access scheme (DMA or flag-driven). Refer to "Data Transfer from the QuadSPI Module Internal Buffers" section in Flash Read section, RX Buffer, data read via register interface and AHB read, for the description of successive accesses to the RX Buffer content. Refer also to Byte Ordering of Serial Flash Read Data for the byte ordering scheme.

Valid address range accessible in the QSPI_ARDBn range depends from the number of RX Buffer entries implemented and from the number of valid buffer entries available in the RX Buffer.

- Example 1, RX Buffer filled completely with 32 words: In this case the address range for valid read access extends from QSPI_ARDB0 to QSPI_ARDB31.
- Example 2, RX Buffer filled with 5 valid words, RX Buffer fill level QSPI_RBSR[RDBFL] is 5. In this case an access to QSPI_ARDB4 provides the last valid entry.

Address: 0h base + 0h 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 W	ARXD																															
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

ARDB*n* field descriptions

Field	Description
ARXD	ARDB provided RX Buffer Data. Byte order (endianness) is identical to the RX Buffer Data Registers.

26.6 Interrupt Signals

The interrupt request lines of the QuadSPI module are mapped to the internal flags according to the following table.

Table 26-19. Assignment of Interrupt Request Lines

IRQ/DMA line	QSPI_FR Flag	Interrupt Description
ipi_int_tfff	TBFF	TX Buffer Fill
ipi_int_tcf	TFF	Peripheral Command Transaction Finished
ipi_int_rfdf	RBDF	RX Buffer Drain
ipi_int_overrun		Buffer Overflow/Underrun Error Logical OR from:
	RBOF	RX Buffer Overrun
	TBUF	TX Buffer Underrun
	ABOF	AHB Buffer Overflow
ipi_int_cerr		Serial Flash Command Error Logical OR from:
	IPAEF	Peripheral access while AHB busy Error
	IPIEF	Peripheral Command could not be triggered Error
	IPGEF	Peripheral access while AHB Grant Error

Table continues on the next page...

Table 26-19. Assignment of Interrupt Request Lines (continued)

IRQ/DMA line	QSPI_FR Flag	Interrupt Description
ipi_int_ored	TBFF, TFF, ILLINE, RBDF, RBOF, TBUF, ABSEF, ABOF, IPAEF, IPIEF, IPGEF,	Logical OR from all the QSPI_FR flags mentioned

26.7 Functional Description

This section provides the functional information of the QuadSPI module.

26.7.1 Serial Flash Access Schemes

Note

In the Individual Flash Mode, all supported commands are available.

26.7.2 Normal Mode

This mode is used to allow communication with an external serial flash device. Compared to the standard SPI protocol, this communication method uses up to 4 bidirectional data lines operating at high data rates. The communication to the external serial flash device consists of an instruction code and optional address, mode, dummy and data transfers. The flexible programmable core engine described below is immune to a wide variety of command/protocol differences in the serial flash devices provided by various flash vendors.

26.7.2.1 Programmable Sequence Engine

The core of the QuadSPI module is a programmable sequence engine that works on "instruction-operand" pairs. The core controller executes each programmed instruction sequentially. The complete list of instructions and the corresponding operands is given in the following table.

Table 26-20. Instruction set

Instruction	Instruction encoding	Pins	Operand	Action on Serial Flash(es)
CMD	6'd1	N=2'd{0,1,2} 2'd0 - One pad 2'd1 - Two pads 2'd2 - Four pads	8 bit command value	Provide the serial flash with operand on the number of pads specified
ADDR	6'd2		Number of address bits to be sent (for example, 8'd24 => 24 address bits required)	Provide the serial flash with address cycles according to the operand on the number of pads specified. The actual address to be provided will be derived from the incoming address in case of AHB initiated transactions and the value of SFAR in case of IPS initiated transactions .
DUMMY	6'd3		Number of dummy clock cycles (should be <= 64 cycles)	Provide the serial flash with dummy cycles as per the operand. The PAD information defines the number of pads in input mode. (for example, one pad implies that pad 1 is not driven, rest all are driven)
MODE	6'd4		8 bit mode value	Provide the serial flash with 8 bit operand on the number of pads specified
MODE2	6'd5	N=2'd{0,1}	2 bit mode value	Provide the serial flash with 2 bit operand on the number of pads[1] specified
MODE4	6'd6	N=2'd{0,1,2}	4 bit mode value	Provide the serial flash with 4 bit operand on the number of pads[2] specified
READ	6'd7	N=2'd{0,1,2} 2'd0 - One pad 2'd1 - Two pads 2'd2 - Four pads	Read data size in bytes (for AHB transactions, the user's application should ensure that data size is a multiple of 8 bytes)	Read data from flash on the number of pads specified. The data size may be overwritten by writing to the ADATSZ field of the QSPI_BUFxCR registers for AHB initiated transactions and IDATSZ field of IP Configuration Register (QuadSPI_IPCR) for IP initiated transactions.
WRITE	6'd8		Write data size in bytes	Write data on number of pads sepcified. The data size may be overwritten by writing to the IDATSZ field of IP Configuration Register (QuadSPI_IPCR) register
JMP_ON_CS	6'd9	NA	Instruction number	Every time the chip select (CS) is deasserted, jump to the instruction pointed to by the operand. This instruction allows the programmer to specify the behavior of the controller when a new read transaction is initiated following a CS deassertion.
STOP	8'd0	NA	NA	Stop execution; deassert CS

1. For a one pad instruction, MODE2 will take 2 serial flash clock cycles on the flash interface.
2. For a one pad instruction, MODE4 will take 4 serial flash clock cycles on the flash interface. For a 4 pad instruction, MODE4 will take 1 serial flash clock cycle on the flash interface.

The programmable sequence engine allows the user to configure the QuadSPI module according to the serial flash connected on board. The flexible structure is easily adaptable to new command/protocol changes from different vendors.

26.7.2.2 Flexible AHB buffers

In order to reduce the latency of the reads for AHB masters, the data read from the serial flash is buffered in flexible AHB buffers. There are four such flexible buffers. The size of each of these buffers is configurable with the minimum size being 0 Bytes and maximum size being the size of the complete buffer instantiated.The size of buffer 0 is defined as being from 0 to QSPI_BUF0IND. The Size of buffer 1 is from QSPI_BUF0IND to QSPI_BUF1IND, buffer2 is from QSPI_BUF1IND to QSPI_BUF2IND and buffer 3 is from QSPI_BUF2IND to the size of the complete buffer, which is given in the chip-specific QuadSPI information.

Each flexible AHB buffer is associated with the following

1. An AHB master. Optionally, buffer3 may be configured as an "all master" buffer by setting the QSPI_BUF3CR[ALLMST] bit. When buffer3 is configured in such a way, any access from a master not associated with any other buffer is routed to buffer3.
2. A datasize field representing the amount of data to be fetched from the flash on every "missed" access.

The master port number of every incoming request is checked and the data is returned/ fetched into the corresponding associated buffer. Every "missed" access to the buffer causes the controller to clear the buffer and fetch QSPI_BUFxCR[ADATSZ] amount of data from the serial flash.As such, there is no benefit in configuring a buffer size of greater than ADATSZ, as the locations greater than ADATSZ will never be used. For any AHB access, the sequence pointed to by the QSPI_BFGENCR[SEQID] field is used for the flash transaction initiated. The data is returned to the master as soon as the requested amount is read from the serial flash. The controller however, continues to prefetch the rest of the data in anticipation of a next consecutive request. Figure 26-3 shows the flexible AHB buffers.

The QSPI_BFGENCR[SEQID] field points to an index of the LUT. Refer to Look-up Table for details.

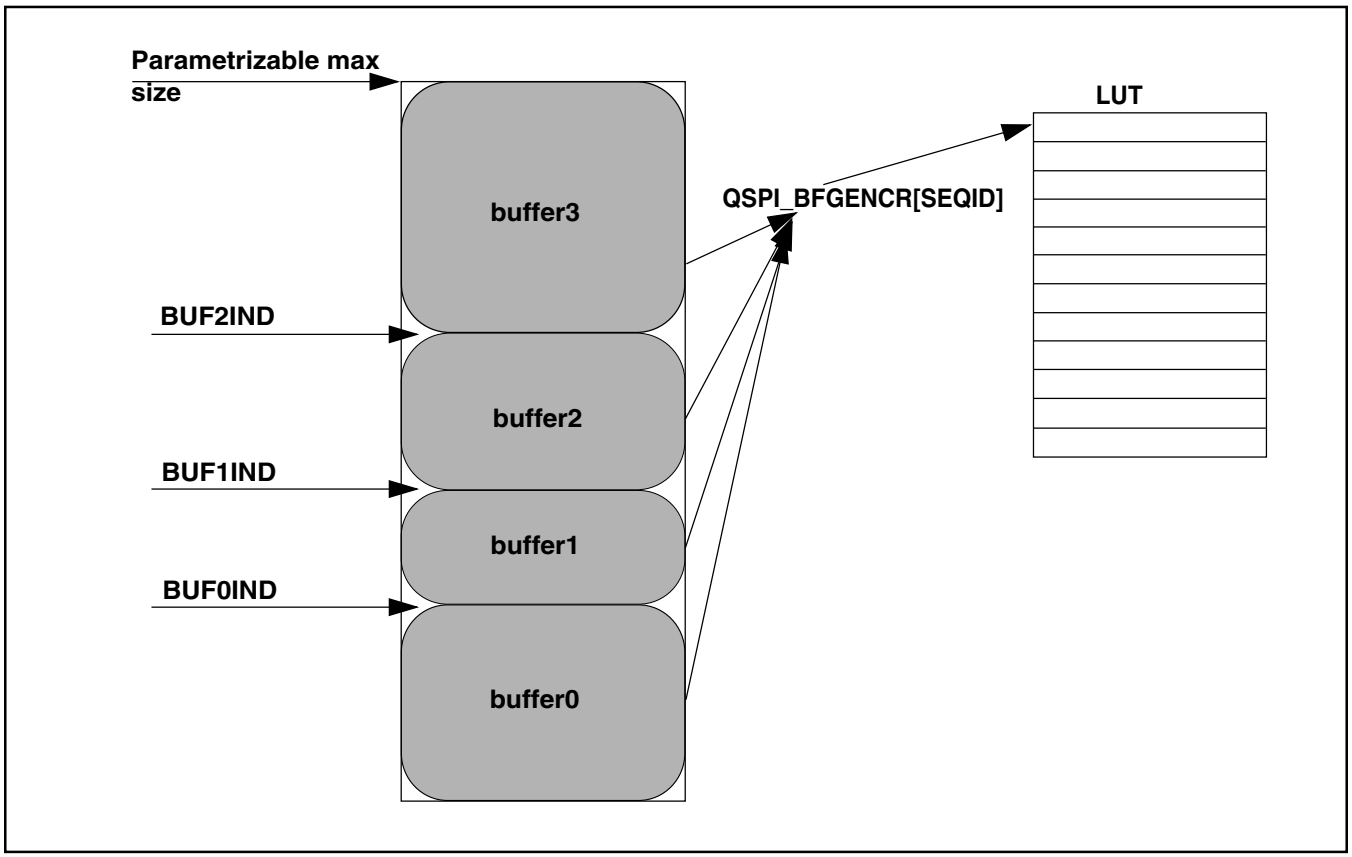


Figure 26-3. Flexible AHB Buffers

Buffer0 may optionally be configured to be associated with a high priority master by setting the QSPU_BUF0CR[HP_EN] bit. An access by a high priority master suspends any ongoing prefetch to any of the other buffers. The ongoing prefetch is suspended and the high priority master is serviced first. Once the high priority masters access completes, the suspended transaction is resumed (before any other AHB access is entertained). The status of the suspended buffer can be read from Sequence Suspend Status Register (QuadSPI_SPNDST).

26.7.2.3 Suspend-Abort Mechanism

Any low priority AHB access can be suspended by a high priority AHB master request. The ongoing transaction is suspended at 64 bit boundary. The suspended transaction is restarted after the high priority master is served and the high priority transaction including data prefetch is completed. While a transaction is in suspended state, it may be aborted if a transaction by the same suspended master is made to a location which is different from the location of the suspended transaction.

Any ongoing transaction is aborted if a request from the same master arrives for a location other than the location at which the transaction is going on. The abort can happen at any point of time.

26.7.2.4 Look-up Table

The Look-up-table or LUT consists of a number of pre-programmed sequences. Each sequence is basically a sequence of instruction-operand pairs which when executed sequentially generates a valid serial flash transaction. Each sequence can have a maximum of 8 instruction-operand pairs. The LUT can hold a maximum of 16 sequences. The figure below shows the basic structure of the sequence in the LUT.

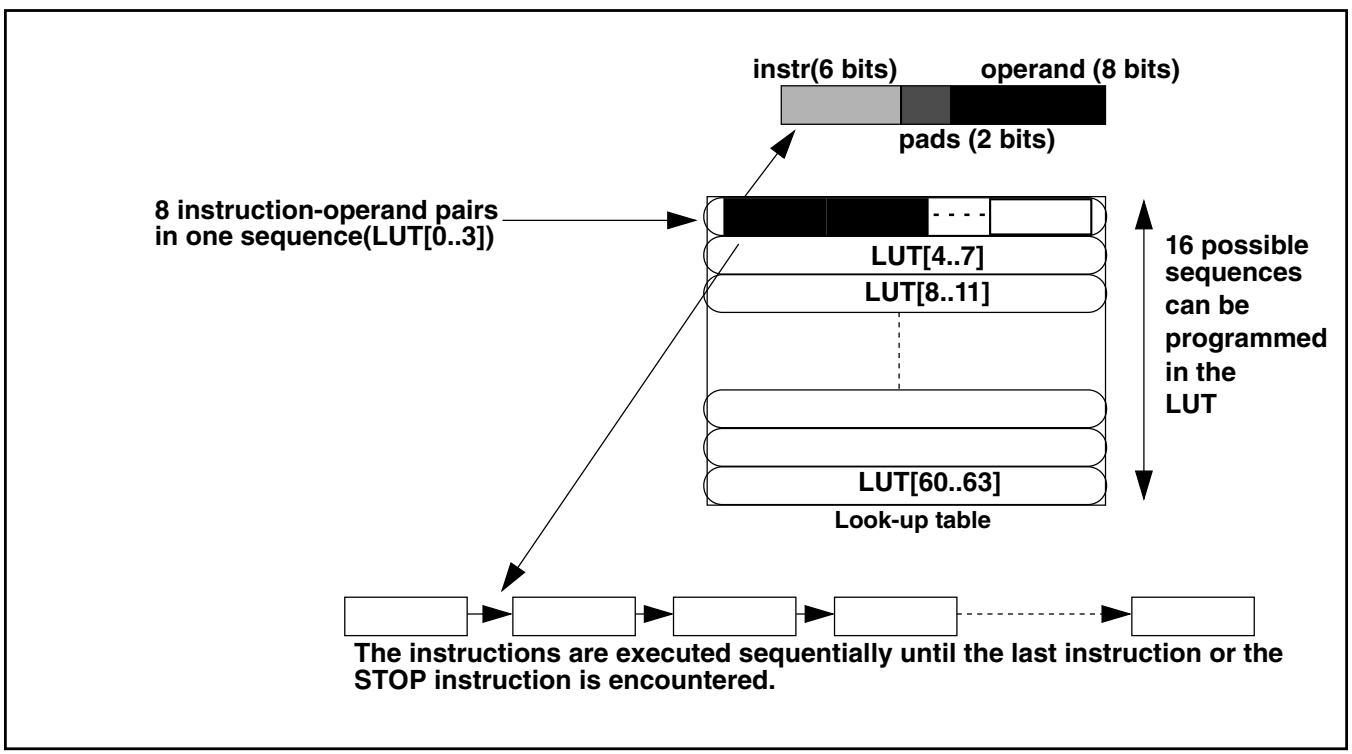


Figure 26-4. LUT and sequence structure

At reset, the index 0 of the look-up-table (LUT[0..3]) is programmed with a basic read sequence as given in Table 26-21. After reset the complete LUT may be reprogrammed according to the device connected on board.In order to protect its contents during a code runover the LUT may be locked, after which a write to the LUT will not be successful until it has been unlocked again.The key for locking or unlocking the LUT is **0x5AF05AF0**. The process for locking and un-locking the LUT is as follows:

Locking the LUT

1. Write the key **(0x5AF05AF0)** in to the LUT Key Register (QuadSPI_LUTKEY).

2. Write 0b01 to the LUT Lock Configuration Register (QuadSPI_LCKCR). Note that this IPS transaction should immediately follow the above IPS transaction (no other IPS transaction can be issued in between). A successful write into this register locks the LUT.

Unlocking the LUT

1. Write the key **(0x5AF05AF0)** into the LUT Key Register (QuadSPI_LUTKEY)
2. Write 0b10 to the LUT Lock Configuration Register (QuadSPI_LCKCR). Note that this IPS transaction should immediately follow the above IPS transaction (no other IPS transaction can be issued in between). A successful write into this register unlocks the LUT.

The lock status of the LUT can be read from QSPI_LCKCR[UNLOCK] and QSPI_LCKCR[LOCK] bit.

Some example sequences are defined in Example Sequences. The reset sequence at LUT index 0 is given in the following table.

Table 26-21. Reset sequence

Instruction	Pad	Operand	Comment
CMD	0x00	0x03	Read Data byte command on one pad
ADDR	0x00	0x18	24 Addr bits to be sent on one pad
READ	0x00	0x08	Read 64 bits
JMP_ON_CS	0x00	0x00	Jump to instruction 0 (CMD)

26.7.2.5 Issuing Serial Flash Memory (SFM) Commands

Each access to the external device follows the same sequence:

1. The user must pre-populate the LUT with the serial flash command sequences that are required for the flash device being used.
2. The QuadSPI module starts executing the instructions in the sequence one by one. The transaction starts and the status bit QSPI_SR[BUSY] is set.
3. Communication with the external serial flash device is started and the transaction is executed.

4. When the transaction is finished (all transmit- and receive operations with the external serial flash device are finished) the status bit QSPI_SR[BUSY] is reset. In case of an IP Command the QSPI_FR[TFF] flag is asserted.

Further details are given in below in Flash Programming and Flash Read.

You can trigger the processing of SFM commands in the QuadSPI module in one of the following ways:

- **Using IP commands**

  For IP Commands the required components need to be written into the following registers:

  - Write the serial flash address to be used by the instruction into QSPI_SFAR, refer to Serial Flash Address Register (QSPI_SFAR). For IP Commands not related to specific addresses, the base address of the related flash need to be programmed.For example, for an instruction which does not require an address (i.e. write enable instruction) the SFAR should be programmed with the base address of the memory the command is to be sent to.
  - Write the sequence ID and data size details in the IP Configuration Register (QSPI_IPCR).
  - Note that the write into the QSPI_IPCR[SEQID] field must be the last step of the sequence. It is possible to combine all fields of the QSPI_IPCR into one single write. Refer to IP Configuration Register (QSPI_IPCR) for details.

  Note that there are some conditions where no IP Command is executed after writing the QSPI_IPCR[SEQID] field and the write operation itself is ignored. They are described in Command Arbitration.

- **Using AHB commands**

  Any AHB memory mapped access is routed to one of the buffers depending on the master port number of the request. If the access is a "miss", a new serial flash transaction is started. The transaction is based on the sequence pointed to by the BFGENCR[SEQID] field as described in Flexible AHB buffers.

  An AHB access is termed memory mapped when the access is to the memory mapped serial flashes, as described in Memory Mapped Serial Flash Data - Individual Flash Mode on Flash A.

  Again the possible error conditions are described in Command Arbitration .

26.7.2.6 Flash Programming

In all cases the memory sector to be written needs to be erased first. The programming sequence itself is then initiated in the following way:

1. Check QuadSPI_SR[BUSY] is de-asserted or '0'. Check that the TX buffer is empty. If it is required to discard the data present in the TX buffer (QSPI_SR[TXNE]) then the TX buffer must be cleared by writing '1' into the QSPI_MCR[CLR_TXF] bit.
2. Program the address related to the command in the QSPI_SFAR register.
3. Provide initial data for the program command into the circular buffer via register TX Buffer Data Register (QSPI_TBDR) . At least one word of data must be written into the TX Buffer up to a maximum of 16.
4. Program the QSPI_IPCR register to trigger the command. The QSPI_IPCR[SEQID] should point to an index of the LUT which has the flash program sequence pre-programmed.The IDATSZ field should be set to denote the size of the write.
5. Depending on the amount of data required, step 3 must be repeated until all the required data have been written into the QSPI_TBDR register.The QSPI_SR[TXFULL] can be used to check if the buffer is ready to receive more data. At any time, the QSPI_TBSR[TRCTR] field can be read to check how many words have been written actually into the TX Buffer.

Upon writing the QSPI_IPCR[SEQID] field (refer to step 4) the QuadSPI module will start to execute the programmed sequence. It is the responsibility of the software to ensure that a correct sequence is programmed into the LUT in accordance with the flash memory connected to the module. The data is fetched from the TX Buffer. It consists of **16** entries of 32-bits and is organized as a circular FIFO, whose read pointer is incremented after each fetch. When all data are transmitted, the QuadSPI module will return from 'busy' to 'idle'. However, this is not true for the external device since the internal programming is still ongoing. It is up to the user to monitor the relevant status information available from the serial flash device and to ensure that the programming is finished properly.

26.7.2.7 Flash Read

Host access to the data stored in the external serial flash device is done in two steps. First, the data must be read into the internal buffers and in the second step these internal buffers can be read by the host.

1. **Reading Serial Flash Data into the QuadSPI Module Internal Buffers**

   A read access to the external serial flash device can be triggered in two different ways:

   - **IP Command Read**: For **reading flash data into the RX Buffer** the user must provide the correct sequence ID in the QuadSPI_IPCR[SEQID] register. The sequence ID points to a sequence in the LUT. It is the responsibility of the software to ensure that a correct read sequence is programmed in the LUT in accordance with the serial flash device connected on board. The user should program the Serial Flash Address Register (QSPI_SFAR) and the IP Configuration Register (QSPI_IPCR) registers. All available read commands supported by the external serial flash are possible.

     Optionally it is possible to clear the RX Buffer pointer prior to triggering the IP Command by writing a 1 into the QuadSPI_MCR[CLR_RXF] field.

     From these inputs, the complete transaction is built when the QSPI_IPCR[SEQID] field is written. The transaction related to the read access starts and the requested number of bytes is fetched from the external serial flash device into the RX Buffer. Since the read access is triggered by an IP command, the IP_ACC status bit and the BUSY bit are both set (both are located in the Status Register (QSPI_SR)). A count of the number of entries currently in the Rx Buffer can be obtained from QSPI_RBSR[RDBFL].

     The communication with the external serial flash is stopped when the specified number of bytes has been read (successful completion of the transaction).

   - **AHB Command Read**: For **reading flash data into the AHB Buffer** the user must set up a read access by a master to the address range in the system memory map which the external serial flash devices are mapped to. The user should also program the buffer registers corresponding to the AHB master initiating the request, this is depends on the configuration of the QSPI_RBCT[RXBRD]. The user should provide the correct sequence ID into the buffer generic configuration register (QSPI_BFGENCR). It is the responsibility of the software to ensure that a correct read sequence in programmed in the LUT in accordance with the serial flash device connected on board. Flash device selection and access mode are determined by the address accessed in the AHB address space associated to the QuadSPI module (refer to Memory Mapped Serial Flash Data - Individual Flash Mode on Flash A,

     On each AHB read access to the memory mapped area the valid data in the AHB Buffer is checked against the address requested in the actual read. When the AHB read request can't be served from the content of the AHB Buffer, the buffer is flushed and the sequence pointed to by the sequence ID is executed by the

controller. The requested number of buffer entries defined in the QSPI_BUFxCR[ADATSZ] field is then fetched from the external serial flash device into the internal AHB Buffer. Since the read access is triggered via the AHB bus, the QSPI_SR[AHB_ACC] status bit is set driving in turn the QSPI_SR[BUSY] bit until the transaction is finished. The communication with the external serial flash is stopped when the specified number of entries has been filled.

2. **Data Transfer from the QuadSPI Module Internal Buffers**

The data read out from the external serial flash device by the QuadSPI module is stored in the internal buffers.The means of accessing the data from the buffer differs depending on which buffer the data has been loaded to. Refer to Block Diagram for details about the two available buffers, the RX Buffer and the AHB Buffer, in the QuadSPI module:

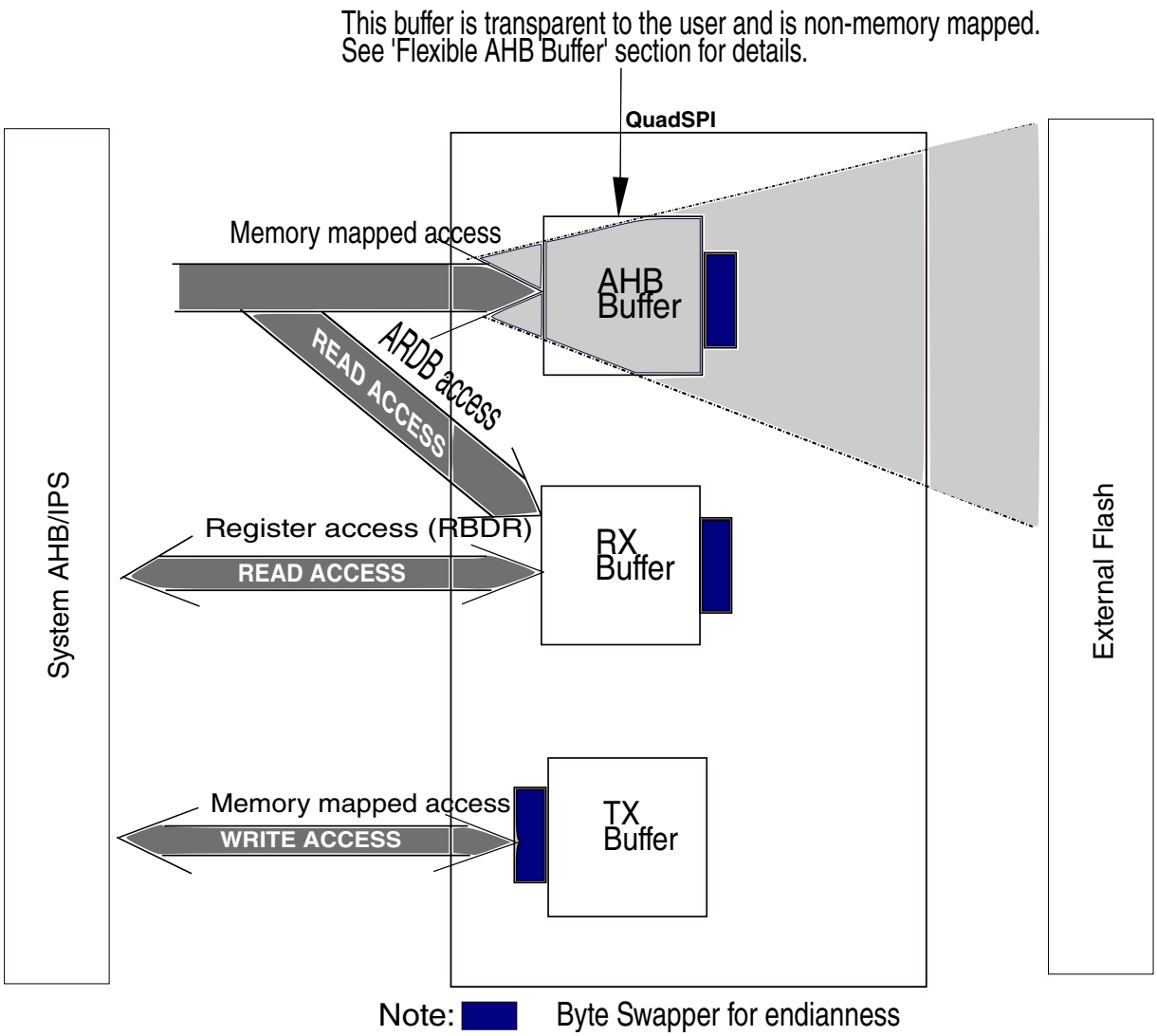


Figure 26-5. QuadSPI memory map

- The RX Buffer is implemented as FIFO of depth 32 entries of 4 bytes. Its content is accessible in two different address areas both referring to the identical data and the same physical memory.

  In the IPS address space in the area associated to QSPI_RBDR0 to QSPI_RBDR31

  In the AHB address space in the area associated to QSPI_ARDB0 to QSPI_ARDB31. Two successive entries are accessed with one single 64 bit AHB read operation.

  RX Buffer operation can be summarized as follows: The QSPI_RBCT[WMRK] field determines at which fill level the RXWE bit is asserted and how many entries are removed from the RX Buffer on each Buffer POP operation. So the QSPI_SR[RXWE] bit indicates that the configured number of data entries is available in the RX Buffer and the QSPI_RBSR[RDBFL] field indicates how many valid entries are available in total. Note that the first entry (QSPI_RBDR0 or QSPI_ARDB0) always corresponds to the first valid entry in the RX Buffer. The software needs to manage the number of valid data bytes itself.

  Further details can be found in RX Buffer Data Register (QuadSPI_RBDR*n*) and in AHB RX Data Buffer (QSPI_ARDB0 to QSPI_ARDB31).

- **Flag-based Data Read of the RX Buffer** is done by polling the QSPI_SR[RXWE] bit. When it is asserted the valid entries can be read either via the IPS address space (QSPI_RBDRn) or the AHB address space (QSPI_ARDBn). A Buffer POP operation must be triggered by the application by writing a 1 into the QSPI_FR[RBDF] bit - this automatically updates the FIFO to point to the next entry as defined by RBCT[WMRK]. For example, if WMRK is set to 3, then the buffer will discard 16 bytes of data.

- **DMA controlled Data Read of the RX Buffer** is done by using the DMA module.The application must ensure that the DMA controller of the related device is programmed appropriately like it is described in DMA Usage.

  DMA controlled read out is triggered fully automatically by the assertion of the QSPI_SR[RXWE] bit. The related Buffer POP operation is also handled completely inside the QuadSPI module. Like in the case above, accessing the RX Buffer content either on QSPI_RBDRn or QSPI_ARDBn related addresses is equivalent.

- **AHB Buffer data read via memory mapped access:** This kind of access is done by reading one of the addresses assigned to the external serial flash device(s) within the range given in Table 26-17 table *under the condition that the data requested are already present in the AHB Buffer or it is currently being read from the serial flash device by the instruction in progress.* If this is not the

case a memory mapped AHB command read is triggered as described above. If the requested data is already available in the AHB Buffer they are provided directly to the host.

When AHB access are made to the flash memory mapped address, the data will be fetched and returned to the AHB interface. Till the data is being fetched the AHB interface would be stalled. As soon as the data from the requested address has been read by the QuadSPI module the AHB read access is served. So it is possible to run sequential reads from the AHB buffer at arbitrary speed without the need to monitor any information about the availability of the data. Nevertheless this access scheme stalls the AHB bus for the time required to read the data from the serial flash device. A better way is (when it is known the access is sequential) to have a prefetch enabled (by programming the ADATSZ field) such that before the next sequential AHB access come, the data is already fetched into the buffer.

As long as the host restricts its accesses to the data already in the buffer and the data currently fetched from the serial flash, it is possible to run the host read from the AHB Buffer in parallel to the serial flash read into the AHB Buffer.

26.7.2.8 Byte Ordering of Serial Flash Read Data

In this paragraph the byte ordering of the serial flash data is given. The basic scheme is that the **first** byte read out of the serial flash device - which is addressed by the QSPI_SFAR[SFADR] field - corresponds to bit position QSPI_RBDR0[0:7] register for IP Command read. In contrast to that for AHB Command read the bytes are always positioned according to the byte ordering of the AHB bus.

- **Byte Ordering in Individual Flash Mode**

  The following table gives the byte ordering scheme of how the byte oriented data space of the serial flash device is mapped into one single 32 bit entry of the RX Buffer or the AHB Buffer. The table is valid within the following context:

  - Flash A in Individual Flash Mode
  - All AHB data read commands with access size of 32 bit

Table 26-22. Byte Ordering in Individual Flash Mode

Serial Flash Byte Numbering	0	1	2	3
Buffer Entry Bit Position [31:0]	[7:0]	[15:8]	[23:16]	[31:24]

Table 26-22. Byte Ordering in Individual Flash Mode

(32 Bit data width)				

Note

For IP Commands the read size can be given in number of bytes. If this number is not a multiple of 4, then the last buffer entry is not completely filled with the missing higher numbered bytes at undefined values.

For AHB Commands, reads, starting from an address not aligned to 32 bit boundaries, the requested bytes are given at the appropriate positions according to the AMBA AHB specification.

- **Buffer Entry Ordering for 64 Bit Read Access**

  For read access via the AHB interface 64 bit access is possible. Each 64 bit access reads 2 32 bit entries simultaneously. The ordering of these 32 bit entries within the 64 bit word is given in the following table.

Table 26-23. 64 Bit Read Access Buffer Entry Ordering

AHB Read Data Bit Position [63:0]	[31:0]	[63:32]
Buffer Entry #	Even (0, 2, 4, ...)	Odd (1, 3, 5, ...)

26.7.2.9 Normal Mode Interrupt and DMA Requests

The QuadSPI module has different flags that can only generate interrupt requests and one flag that can generate interrupt as well as DMA requests. The following table lists the eight conditions. Note that the flags mentioned in the table are related to the Flag Register (QSPI_FR).

Table 26-24. Interrupt and DMA Request Conditions

Condition	Flag(QSPI_FR)	DMA
TX Buffer Fill	TBFF	-
TX Buffer Underrun	TBUF	-
Illegal Instruction Error	ILLINE	-
RX Buffer Drain	RBDF	X

Table continues on the next page...

Table 26-24. Interrupt and DMA Request Conditions (continued)

Condition	Flag(QSPI_FR)	DMA
RX Buffer Overflow	RBOF	-
AHB Buffer Overflow	ABOF	-
AHB Sequence Error	ABSEF	-
IP Command Trigger during AHB Access Error	IPAEF	-
IP Command Trigger could not be executed Error	IPIEF	-
IP Access during AHB Grant Error	IPGEF	-
IP Command related Transaction Finished	TFF	-

Each condition has a flag bit in the Flag Register (QSPI_FR) and a Request Enable bit in the DMA Request Select and Enable Register (QSPI_RSER) . The RX Buffer Drain Flag (RBDF) has separate enable bits for generating IRQ and DMA requests. Note that not all flags have an individual IRQ line. Check the devices Interrupt Vector Table for more details.

- Transmit Buffer Fill Interrupt Request:

  The Transmit Buffer Fill IRQ indicates that the TX Buffer can accept new data. It is asserted if the QSPI_FR[TBFF] flag is asserted and if the corresponding enable bit (QSIP_RSER[TBFIE]) is set. Refer to TX Buffer Operation, for details about the assertion of the QSPI_FR[TBFF] flag.

- Receive Buffer Drain Interrupt or DMA Request:

  The Receive Buffer Drain IRQ derived from the QSPI_FR[RBDF] flag indicates that the RX Buffer of the QuadSPI module has data available from the serial flash device to be read by the host. It remains set as long as the QSPI_RBSR[RXWE] bit is set. The QSPI_RSER[RBDIE] bit enables the related IRQ.

  Aside from the IRQ it is possible to handle RX Buffer drain by DMA. If the QSPI_RSER[RBDDE] bit is set, a DMA request will be triggered when the RX Buffer contains more than QSPI_RBCT[WMRK] valid entries.The application must set the environment appropriately (for example, the DMA controller) for the DMA transfers.

- Buffer Overflow/Underrun Interrupt Request:

  The Buffer Overflow/Underrun IRQ is a combination of the following flags (all located in the QSPI_FR register with the related enable bits in the QSPI_RSER register):

- TBUF - TX Buffer Underrun, enabled by TBUIE
- RBOF - RX Buffer Overflow, enabled by RBOIE
- ABOF - AHB Buffer Overflow, enabled by ABOIE

  The Transmit Buffer Underrun indicates that an underrun condition in the TX Buffer has occurred. It is generated when a write instruction is triggered whilst the Tx Buffer is empty and the QSPI_RSER[TBUIE] bit is set.

  The Receive Buffer Overflow indicates that an overflow condition in the RX Buffer has occurred. It is generated when the RX Buffer is full, an additional read transfer attempts to write into the RX Buffer and the QSPI_RSER[RBOIE] bit is set.

  The AHB Buffer Overflow indicates that an overflow condition in the AHB Buffer has occurred. It is generated when the AHB Buffer is full, an additional read transfer attempts to write into the AHB Buffer and the QSPI_RSER[ABOIE] bit is set.

  The data from the transfers that generated the individual overflow conditions is ignored.

- Serial Flash Command Error Interrupt Request

  If the IPAEF, IPIEF, IPGEF flags in the QSPI_FR are set, and the related interrupt enable bits in the QSPI_RSER are also set, then an interrupt is requested.

- Transaction Finished Interrupt Request

  The IP Command Transaction Finished IRQ indicates the completion of the current IP Command.It is triggered by the QSPI_FR[TFF] flag and is masked by the QSPI_RSER[TFIE] bit.

26.7.2.10 TX Buffer Operation

The TX Buffer provides the data used for page programming. For proper operation it is required to provide at least one entry in the TX Buffer prior to starting the execution of the page programming command. The application must ensure that the required number of data bytes is written into the TX Buffer fast enough as long as the command is executed without a TX Buffer overflow or underrun.

The QuadSPI module sets the QSPI_FR[TBFF] flag so long as the TX Buffer is not full and can accept more data.

When the QuadSPI module tries to pull data out of an empty TX Buffer the TX Buffer underrun is signaled by the QSPI_FR[TBUF] flag. The current IP Command leading to the underrun condition is continued until the specified number of bytes has been sent to the serial flash device, in the underrun condition when QuadSPI module tries to pull out data of empty TX buffer, the data transferred is undefined i.e. once the underrun flag is set, it will return the garbage value until the required number of bytes are not sent. When this Sequence Command is finished, the QSPI_FR[TBFF] flag is asserted indicating that the Tx Buffer is ready to be written again.

The TX Buffer overflow isn't signaled explicitly, but the TX Buffer fill level can be monitored by the QSPI_TBSR[TRBFL] field.

Refer to TX Buffer Status Register (QuadSPI_TBSR) and Flag Register (QuadSPI_FR) for details about the TX Buffer related registers.

26.7.2.11 Address scheme

Earlier serial flash memories supported only 24-bit address space hence restricting the maximum memory size of the serial flash as 16 MB. The new memory specification supports two types of 32-bit addressing mode in addition to legacy 24-bit address mode.

- **Extended Address Mode**

  In this mode, the legacy 24-bit commands are converted to accept 32-bit address commands. The flash memory needs to be configured for 32-bit address mode. Also, while programming the LUT sequence in QuadSPI for 32-bit mode, the ADDR command should be programmed with 8’d32 as the operand value. By default, the QuadSPI is in 24-bit legacy address mode. Each of the memory vendors have a different way of enabling this mode (Refer to the memory specification from memory vendors). For example, the command B7h sent to Macronix flash will enable it for 32-bit address mode.

- **Extended Address register**

  In this mode, the upper 8-bit of the 32-bit address is provided by the Extended address register in the memory itself. The memory provides a specific register which is updated according to the address to be accessed. This effectively converts the legacy 24-bit address command into 32-bit address commands. The memories greater in size than 16 MB, consists of banks of 16 MB. The 8-bit written in the extended address register effectively enables a bank. For example in Spansion memory, when the extended address register is updated with a value of 0x01 with the help of the command 17h, it will open Bank1 of the memory. The consequent 24-bit address

commands will lead to Bank1. The extended address register needs to be update with the respective value for access to other banks. This effectively converts the legacy 24-bit address command into 32-bit address commands.

26.8 Initialization/Application Information

This section provides the initialization and application information of the QuadSPI module.

26.8.1 Power Up and Reset

Note that the serial flash devices connected to the QuadSPI module may require special voltage characteristics of their inputs during power up or reset. It is the responsibility of the application to ensure this.

26.8.2 Available Status/Flag Information

This paragraph gives an overview of the different status and flag information available and their interdependencies for different use cases. Related registers are QSPI_SR and QSPI_FR. Refer to the related descriptions how to set up the QuadSPI module appropriately.

26.8.2.1 IP Commands

Refer to IP Configuration Register (QuadSPI_IPCR) for additional details not explicitly covered in this paragraph.

- **IP Commands - Normal Operation**

  Writing the QSPI_IPCR[SEQID] field triggers the execution of a new IP Command. Given that this is a legal command the QSPI_SR[IPACC] and the QSPI_SR[BUSY] bits are asserted simultaneously, immediately after the execution is started.

  When the instruction on the serial flash device has been finished these bits are de-asserted and the QSPI_FR[TFF] flag is set.

- **IP Commands - Error Situations**

  Refer to Table 26-25 below.

26.8.2.2 AHB Commands

Refer to Section 1, Reading Serial Flash Data into the QuadSPI Module, in Flash Read for additional details not explicitly covered in this paragraph.

- **AHB Commands - Normal Operation**

  Memory mapped read access to a serial flash address not contained in the AHB Buffer, triggers the execution of an AHB Command. Given that this is a legal command the QSPI_SR[AHBACC] and the QSPI_SR[BUSY] bits are asserted simultaneously immediately after the execution is started. When the instruction on the serial flash device has been finished these bits are de-asserted.

- **IP Commands - Error Situations**

  Refer to Table 26-25 below.

26.8.2.3 Overview of Error Flags

The following table gives an overview of the different error flags in the QSPI_FR register and additional error-related details.

Table 26-25. Overview of QSPI_FR Error Flags

Error Category	Error Flag in QSPI_FR	Command Execution on Serial Flash Device TFF Behavior (in case of IP commands only)	Description
AHB Error Flag	ABSEF	Flash transaction is aborted	AHB sequence contains • WRITE instruction
AHB Error Flag	ABOF	Flash transaction continues until it finishes	Set when the module tried to push data into the AHB buffer that exceeded the size of the AHB buffer. Only occurs due to wrong programming of the QSPI_BUFxCR[ADATSZ].
Miscellaneous Error Flag	ILLINE	Flash transaction aborted	Illegal instruction Error Flag – Set when an illegal instruction is encountered by the controller in any of the sequences.
Command Arbitration Error	IPIEF	TFF not asserted in conjunction with that command	IP Command Error - caused when IP access is currently in progress (IP_ACC set) and • write attempt to QSPI_IPCR register.

Table continues on the next page...

Table 26-25. Overview of QSPI_FR Error Flags (continued)

Error Category	Error Flag in QSPI_FR	Command Execution on Serial Flash Device TFF Behavior (in case of IP commands only)	Description
			• write attempt to QSPI_SFAR register. • write attempt to QSPI_RBCT register.
Command Arbitration Error	IPAEF		• AHB Command already running, another IP Command could not be executed. • AHB Command already running, write attempt to QSPI_IPCR[SEQID] field.
Command Arbitration Error	IPGEF		• Exclusive access to the serial flash granted for AHB Commands, write attempt to QSPI_IPCR[SEQID] field.
Buffer Related Error	RBOF	TFF is asserted on completion	• RX Buffer Overrun
Buffer Related Error	TBUF		• TX Buffer Underrun

Note that only the buffer related errors are related to a transaction on the external serial flash. All the other errors do not trigger an actual transaction.

26.8.2.4 IP Bus and AHB Access Command Collisions

There are two flags related to this topic, the QSPI_FR[IPAEF] and QSPI_FR[IPIEF]. Refer to sub-section "Reading Serial Flash Data into the QuadSPI Module" of Flash Read section, for a description of the flags and Command Arbitration , for details about possible command collisions.

26.8.3 Exclusive Access to Serial Flash for AHB Commands

It is possible that several masters need to access the serial flash device connected to the QuadSPI module separately, one master by triggering IP Commands and reading the RX Buffer (via RBDR*n* register) and the other masters by triggering AHB Commands (via ARDB*n* Registers). These two set of buffer (RBDR and ARDB Buffer) points to the same physical buffer.Refer to Figure 26-5 To avoid command collisions resulting in excessive latencies the QuadSPI module implements a request-handshake mechanism

between the master triggering AHB Commands and the QuadSPI module allowing this specific master to request exclusive access to the serial flash device for AHB Commands. If this exclusive access is granted the execution of IP Commands is blocked. This resolves command collisions and excessive times where the AHB interface may be blocked.

If this capability is used in the device there is additional status and flag information available related to this mechanism. The QSPI_SR[AHBGNT] bit reflects the module-internal state that the exclusive access mentioned above is granted, any attempt to trigger an IP Command is rejected and results in the assertion of the QSPI_FR[IPGEF] flag. Refer to the descriptions of the related bit and flag for details.

It is within the responsibility of the application to set up the master using this mechanism appropriately, if used incorrectly no IP Commands at all can be triggered.

Two different cases can be distinguished:

26.8.3.1 RX Buffer Read via QSPI_ARDB Registers

In this case all masters share the AHB bus for RX Buffer as well as for AHB Buffer read. In this case the access to the AHB interface by the master triggering AHB Commands must be deferred until any pending IP Command has been finished **and** the RX Buffer readout has been finished as well. The QSPI ARDB Buffers access the Rx buffer i.e the data from the Rx Buffer is returned and no data from AHB Buffer is touched. This is the conservative use case, corresponding to the reset value 0 of the QSPI_RBCT[RXBRD] bit.

In this case the QSPI_SR[AHBGNT] bit is asserted not earlier than any running IP Command has been finished (QSPI_SR[IP_ACC] is 0), the RX Buffer has been read out completely (QSPI_RBSR[RDBFL] equal to 0) or no DMA read is pending (QSPI_SR[RXDMA] equal to 0 and Rx Buffer readout is via AHB(QSPI_RBCT[RXBRD]) equal to 1.

26.8.3.2 RX Buffer Read via QSPI_RBDR Registers

This is the preferred use case as an access to the AHB buffer (memory mapped flash) does not interfere with any IPS access to read the RBDR buffer. It is not possible that a pending AHB bus access triggered by an AHB Command stalls the AHB bus and blocks the RX Buffer readout since the RX Buffer is read via the IP bus based registers QSPI_RBDR0 to QSPI_RBDR31.

For this case it is recommended to program the QSPI_RBCT[RXBRD] bit to 1. The QSPI_SR[AHBGNT] bit is asserted immediately after any running IP Command has been finished (QSPI_SR[IP_ACC] is 0), the RX Buffer has been read out completely (QSPI_RBSR[RDBFL] equal to 0) or no DMA read is pending (QSPI_SR[RXDMA] equal to 0, allowing the master triggering AHB Commands to trigger AHB Commands as soon as possible without the need to wait for the RX Buffer readout to be finished.

26.8.4 Command Arbitration

In case of overlapping commands, the arbitration scheme is described in the following paragraphs under the assumption that the priority mechanism described in Exclusive Access to Serial Flash for AHB Commands is **not** used:

- During the execution of an IP Command, the running IP Command can't be terminated by issuing another IP Command or AHB Command. The QSPI_FR[IPIEF] flag is asserted when the host tries to write into the QSPI_IPCR register. When the host triggers an AHB Command (refer to sub-section "Reading Serial Flash Data into the QuadSPI Module" of Flash Read section, for details), this command is stalled until the currently running IP Command is finished.
- During the execution of an AHB Command, the running AHB Command can't be terminated by issuing an IP Command. The command is ignored and the QSPI_FR[IPAEF] flag is asserted. Refer to Flag Register (QuadSPI_FR) for the description of these flags.

  When another AHB Command is triggered the address of the memory mapped access is considered. If the requested address is currently read from the serial flash device, the running command is continued. If this is not the case the currently running command is terminated and another AHB Command related to the requested address is executed. Refer to sub-section "Reading Serial Flash Data into the QuadSPI Module" of Flash Read section, for further details.

In case of coinciding commands the IP Command is triggered and the AHB Command is stalled until the IP Command has been finished (QSPI_SR[IP_ACC] has been deasserted).

The IP Commands ignored in case of command collision will not result in the assertion of the QSPI_FR[TFF] flag.

26.8.5 Flash Device Selection

Regardless of the SFM Command (IP or AHB) the access mode is selected by specifying the 32 bit address value for the following SFM Command.

For IP Commands the access mode is selected with the address programmed into the QSPI_SFAR register. Refer to Serial Flash Address Register (QuadSPI_SFAR) for details.

For AHB Commands the access mode is determined by the memory mapped address which is accessed Refer to Flash memory mapped AMBA bus for details.

26.8.6 DMA Usage

For the complete description of the DMA module refer to the related DMA Controller chapter. In this paragraph only the details specific to the DMA usage related to the QuadSPI module are given.

26.8.6.1 DMA Usage in Normal Mode

26.8.6.1.1 Bandwidth considerations

Careful consideration of the throughput rate of the entire chain (serial flash -> AHB bus / IP Bus -> DMA controller) involved in the read data process is essential for proper operation. Such analysis must take into account not only the data rate provided by the serial flash but also the data rate of the AHB bus and the performance of the DMA controller in reading data from the RX buffer.

Two figures must match for proper operation, that means that the data rate provided by the serial flash device must not exceed the average RX Buffer readout data rate. Otherwise, the longer this state persists, a RX Buffer overflow will result.

AHB Bus Side (data read):

The total number of bus cycles for each DMA Minor Loop completion is added from the following components:

- Overhead for each minor loop, given by DMA controller: Assume 10 cycles
- Overhead due to clock domain crossing: Assume 2 cycles
- Number of bus clock cycles required for 8 bytes (64 bit read size): Assume 2 cycles (read/write sequence of DMA controller)

Note that the size of the minor loop is determined by the size of the QSPI_RBCT[WMRK] field, therefore the overhead given above distributes among (QSPI_RBCT[WMRK]+1)/2 read accesses of 64 bit each.

The following table gives some examples for typical use cases:

Table 26-26. Access Duration Examples - Bus Clock Side

QSPI_RBCT[WMRK]	Number of Bytes per DMA Loop [1]	Number of Bus Clock cycles for DMA Minor Loop	Time Duration of DMA Minor Loop for 120Mhz Bus clock Frequency
0	4	12+2 = 14	~117ns
1	8	12+2 = 14	~117ns
3	16	12+4 = 16	~133ns
7	32	12+8 = 20	~167ns
11	48	12+12 = 24	~200ns

1. DMA Loop means one Minor Loop Completion which is equivalent to one.

NOTE

The table figure represents ideal scenario, actual performance will depend on how the system is integrated.

Serial Flash Device Side (data read):

The number of serial flash cycles can be determined in the following way:

- Number of serial flash clock cycles required to read 4 bytes, corresponding to one RX Buffer entry (setup of command and address not considered): , 8 cycles for Quad Mode (SDR) instructions in Individual Flash Mode etc.
- Overhead due to clock domain crossing : 1 cycle.

The following table lists the number of clock cycles required to read the data from the serial flash corresponding to the different settings of the QSPI_RBCT[WMRK] field:

A complementary example would be when the watermark is set to be too high. In such a case, the time taken by the DMA to read out the RX buffer entries should be smaller than the time taken by the controller to push in the remaining entries in the buffer.

NOTE

The tables mentioned above are only examples which must be correlated with the DMA in the system.

26.9 Byte Ordering - Endianness

QuadSPI provides support for swapping the flash read/write data based on the configuration of the QSPI_MCR[END_CFG]. By default the data is always returned in 64 bit BE format on the AHB bus and 32 bit BE format on the IPS interface when read via the RX buffer and written in 32 bit BE format when written via the TX buffer.

The table(QSPI_MCR[END_CFG]) below shows the complete bit ordering. BE signifies Big Endian which means the high order bits of the associated data vectors are associated with low order address positions. LE signifies Little Endian which means the lower order bits of the associated data vectors are associated with low order address positions.Refer to figure Figure 26-5

Table 26-27. QSPI_MCR[END_CFG]

00	64 bit BE
01	32 bit LE
10	32 bit BE
11	64 bit LE

The tables below (Byte ordering configuration in AHB) and (Byte ordering configuration in IPS) show how this configuration is implemented in QSPI AHB and IPS interfaces respectively. B in the table signifies Byte and the index 1-8 refers to the byte position i.e. 1 refer to bits[7:0], 8 refer to bits[63:56] and so on.

Table 26-28. Byte ordering configuration in AHB

64 bit BE	B1	B2	B3	B4	B5	B6	B7	B8
64 bit LE	B8	B7	B6	B5	B4	B3	B2	B1
32 bit BE	B5	B6	B7	B8	B1	B2	B3	B4
32 bit LE	B4	B3	B2	B1	B8	B7	B6	B5

Table 26-29. Byte ordering configuration in IPS

32BE	B1	B2	B3	B4
32LE	B4	B3	B2	B1

The examples below show the byte ordering in 64 bit BE configuration for AHB Buffer and 32 bit BE for TX/RX Buffer:

26.9.1 Programming Flash Data

CPU write instructions to the QSPI_TBDR register like

- Write QSPI_TBDR -> 0x01_02_03_04
- Write QSPI_TBDR -> 0x05_06_07_08

result in the following content of the TX Buffer:

Table 26-30. Example of QuadSPI TX Buffer

TX Buffer Entry	Content
0	32'h01_02_03_04
1	32'h05_06_07_08

Programming the TX Buffer into the external serial flash device results in the following byte order to be sent to the serial flash:

- 01…02…03…04…05…06…07…08

26.9.2 Reading Flash Data into the RX Buffer

Reading the content from the same address provides the following sequence of bytes, identical to the write case:

- 01…02…03…04…05…06…07…08

This results in the RX Buffer filled with:

Table 26-31. Resulting RX Buffer Content

RX Buffer Entry	Content
0	32'h01_02_03_04
1	32'h05_06_07_08

26.9.2.1 Readout of the RX Buffer via QSPI_RBDRn

The RX Buffer content appears at CPU read access via the Peripheral bus interface in the following order:

- Read QSPI_RBDR0 <- 0x01_02_03_04
- Read QSPI_RBDR1 <- 0x05_06_07_08

26.9.2.2 Readout of the RX Buffer via ARDBn

The RX Buffer content appears at read access on the AMBA AHB interface at the QuadSPI module boundary:

- (1a): 32 Bit Access: Read QSPI_ARDB0 <- 0x01_02_03_04
- (2a): 32 Bit Access: Read QSPI_ARDB1 <- 0x05_06_07_08
- (1b/2b): 64 Bit Access: Read QSPI_ARDB0 <- 0x01_02_03_04_05_06_07_08

26.9.3 Reading Flash Data into the AHB Buffer

Reading the content from the same address as it was written to provides the following sequence of bytes, identical to the write case:

- 01…02…03…04…05…06…07…08

This results in the AHB Buffer filled with:

Table 26-32. Resulting AHB Buffer Content

AHB Buffer Entry	Content
0	64'h01_02_03_04_05_06_07_08

26.9.3.1 Readout of the AHB Buffer via Memory Mapped Read

The AHB Buffer content appears at read access on the AMBA AHB interface at the QuadSPI module boundary:

- (1a): 32 Bit Read Access: <- 0x01_02_03_04
- (2a): 32 Bit Read Access: <- 0x05_06_07_08
- (1/2): 64 Bit Read Access: <- 0x01_02_03_04_05_06_07_08

26.10 Serial Flash Devices

Several different vendors make flash devices with a QuadSPI interface. At present there is no set standard for the QuadSPI instruction set. Most common commands currently have the same instruction code for all vendors, however some commands are unique to specific vendors. Some example sequences are provided below.

26.10.1 Example Sequences

This section provides the example sequences of the QuadSPI module.

Table 26-33. Exit 4 x I/O Read Enhance Performance Mode (XIP) (Macronix) and Read Status

INSTR	PAD	OPERAND	COMMENT
CMD	0x0	0xEB	4xIO Read Command
ADDR	0x2	0x18	24 Bit address to be send on 4 pads
MODE	0x2	0x00	2 mode cycles (exit XIP)
DUMMY	0x0	0x04	4 dummy cycles
READ	0x2	0x08	Read 64 bits
CMD	0x0	0x05	Read Status register
READ	0x0	0x01	Status register data
STOP	0x0	0x00	STOP, Instruction over

26.10.1.1 Fast Read Sequence (Macronix/Numonyx/Spansion/ Winbond)

The following table shows the fast read sequence for Macronix/Numonyx/Spansion/ Winbond flashes.

Table 26-34. Fast Read sequence

Instruction	Pad	Operand	Comment
CMD	0x0	0x0B	Fast Read command = 0x0B
ADDR	0x0	0x18	24 Addr bits to be sent on one pad
DUMMY	0x0	0x08	8 Dummy cycles
READ	0x0	0x04	Read 32 Bits on one pad
JMP_ON_CS	0x0	0x00	Jump to instruction 0 (CMD)

26.10.1.2 Fast Read Quad Output (Winbond)

The following table shows the Fast read quad output sequence for Winbond memories

Table 26-35. Fast Read Quad output sequence

Instruction	Pad	Operand	Comment
CMD	0x0	0x6B	Fast read quad output command = 0x6B
ADDR	0x0	0x18	24 Addr bits to be sent on 1 pad
DUMMY	0x0	0x08	8 Dummy cycles
READ	0x2	0x04	Read 32 Bits on 4 pads
JMP_ON_CS	0x0	0x00	Jump to instruction 0 (CMD)

26.10.1.3 4 x I/O Read Enhance Performance Mode (XIP) (Macronix)

The following table shows the 4 x I/O Read Enhance Performance Mode for Macronix flashes. The enhanced performance mode is also known as XIP mode.

Table 26-36. Fast Read Quad output sequence

Instruction	Pad	Operand	Comment
CMD	0x0	0xEB	4xI/O Read command = 0xEB
ADDR	0x2	0x18	24 Addr bits to be sent on 4 pads
MODE	0x2	0xA5	2 mode cycles
DUMMY	0x0	0x04	4 Dummy cycles
READ	0x2	0x04	Read 32 Bits on 4 pads
JMP_ON_CS	0x0	0x01	Jump to instruction 1 (ADDR)

When in XIP mode the software should ensure that all the flashes connected to the controller are in XIP mode. As a part of initializing the controller, all the flashes may be enabled with XIP by carrying out dummy reads.

26.10.1.4 Dual Command Page Program (Numonyx)

The following table shows the Dual command page program sequence for Numonyx flashes.

Table 26-37. Dual Command Page Program sequence

Instruction	Pad	Operand	Comment
CMD	0x1	0x02	Dual command page program = 0x02 on 2 pads
ADDR	0x1	0x18	24 Addr bits to be sent on 2 pads
WRITE	0x1	0x20	Write 32 Bytes on 2 pads
STOP	0x0	0x00	STOP, Instruction over

26.10.1.5 Sector Erase (Macronix/Spansion/Numonyx)

The following table shows the Sector erase sequence for Macronix/Spansion/Numonyx flashes

Table 26-38. Sector Erase sequence

Instruction	Pad	Operand	Comment
CMD	0x0	0xD8	Sector erase command = 0xD8
ADDR	0x0	0x18	24 Addr bits to be sent on 1 pad
STOP	0x0	0x00	STOP, Instruction over

26.10.1.6 Read Status Register (Macronix/Spansion/Numonyx/ Winbond)

The following table shows the Read status register sequence for Macronix/Spansion/ Numonyx/Winbond flashes.

Table 26-39. Read Status Register Sequence

Instruction	Pad	Operand	Comment
CMD	0x0	0x05	Read status register command = 0x05
READ	0x0	0x01	Read status register data
STOP	0x0	0x00	STOP, Instruction over

26.10.2 Boot initialization sequence

The following are the recommended sequence of steps for booting from QuadSPI:

- System out of reset and flash available (300us)

- Clocks still at very low frequency. Clock tree configured, I/O pins configured. First request sent to QuadSPI for address 0x0 of flash.
- The reset command sequence in QuadSPI has 0x03 (basic read command) which is applicable to all flashes at < 50MHz serial flash clock
- The first few bytes of data is read from the flash which contains the following information:
  - The total sizes of all the flashes connected on board
  - Continuous mode entry sequence
  - 24bit or 32bit addressing (assuming 24bit for first accesses)
- All the serial flashes are configured
  - Quad Mode enabled
  - Dummy reads to enter into XIP
- QuadSPI is configured
  - LUT configured for highest performance reads
  - Buffers configured
- Serial flash clock frequency increased.
- Boot reads happen in quad output mode @66MHz.

26.10.3 Serial Flash Clock Frequency Limitations

Certain commands of some serial flash devices are limited in the frequency applied to the serial flash device on command execution. In order to support these commands without having to recalibrate the module clocks, the the serial flash device clock can be divided by 2 (half speed) by setting the QSPI_SMPR[HSENA] bit. The SCLK will return to full speed once the the QSPI_SMPR[HSENA] bit is cleared.

26.11 Sampling of Serial Flash Input Data

26.11.1 Basic Description

QuadSPI is used to read data from the serial flash device. Depending on the actual implementation, there is a delay between the internal clocking in the QuadSPI module and the external serial flash device. Refer to the following figure for an overview of this scheme.

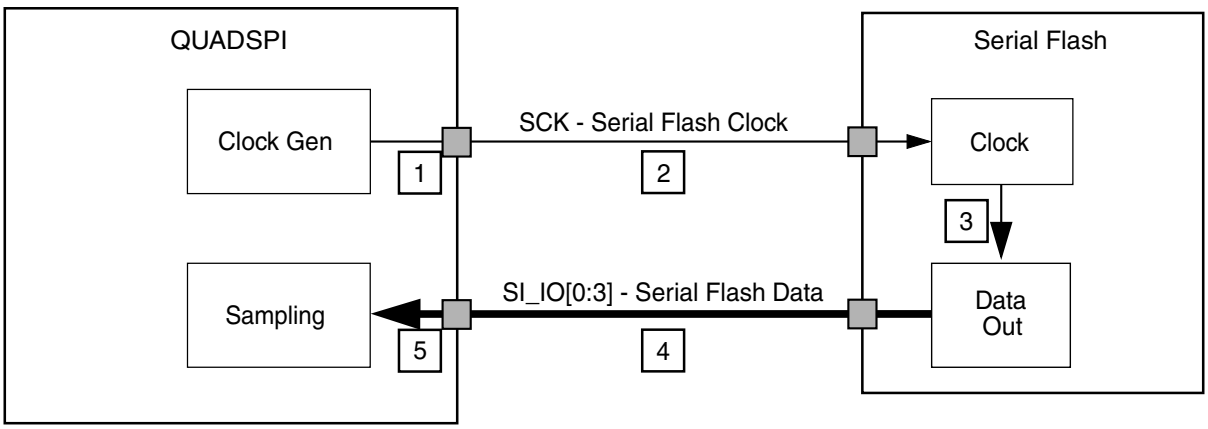


Figure 26-6. Serial Flash Sampling Clock Overview

The rising edge of the internal reference clock is taken as timing reference for the data output of the serial flash. After a time of $t_{Del,total}$ the data arrives at the internal sampling stage of the QuadSPI module. According to the Serial Flash Sampling Clock Overview figure, the following parts of the delay chain contribute to $t_{Del,total}$:

1. Output delay of the serial flash clock output of the device containing the QuadSPI module
2. Wire delay of application/PCB from the device containing the QuadSPI module to the external serial flash device
3. Clock to data out delay of the external serial flash device, including input and output delays
4. Wire delay of application/PCB from the external serial flash device to the device containing the QuadSPI module
5. Device delay corresponding to the input data

NOTE

The amount of total delay $t_{Del,total}$ is specific to the characteristics of the actual implementation. Also, the serial flash device clock (SCK) is inverted with respect to the QuadSPI internal reference clock.

26.11.2 Supported read modes

Some modes listed here may not be available on this chip. See the chip-specific QuadSPI information for the read modes that this chip supports.

26.11.2.1 SDR mode

Most flash memories operate in single data rate (SDR) mode. In SDR mode, the data is transferred only on one edge of the clock signal. The SDR serial flash memories sample the incoming data on the rising edge of serial flash clock and drive the output data on the falling edge of the serial flash clock.

26.11.2.1.1 Internal sampling

NOTE

This mode may not be available on this chip. See the chip-specific QuadSPI information for the read modes that this chip supports.

QuadSPI uses different edges of the internal reference clock for sampling the input data in SDR mode.

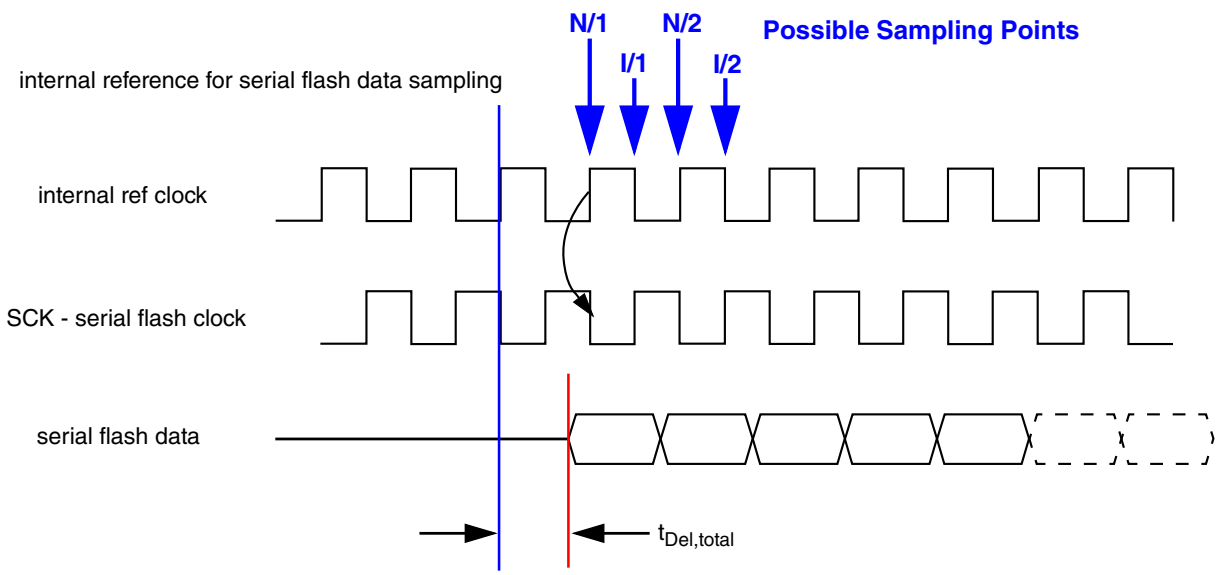


Figure 26-7. Internal sampling in SDR mode

The possible points in time for sampling incoming data are denoted as N/1, I/1, N/2 and I/2 above. The sampling point relevant for the internal sampling is configured in the QSPI_SMPR register. Refer to Sampling Register (QuadSPI_SMPR) for details. The following table gives an overview of the available configurations for the commands running at regular (full) speed:

Table 26-40. Sampling Configuration

Sampling Point	Description	Delay [FSDLY] [HSDLY]	Phase [FSPHS] [HSPHS]	QSPI_SMPR for Full Speed Setting[1]
N/1	sampling with non-inverted clock, 1 sample delay	0	0	0x0000000x
I/1	sampling with inverted clock, 1 sample delay	0	1	0x0000002x
N/2	sampling with non-inverted clock, 2 samples delay	1	0	0x0000004x
I/2	sampling with inverted clock, 2 samples delay	1	1	0x0000006x

1. 'x' is not considered here

Depending on the actual delay and the serial flash clock frequency, the appropriate sampling point can be chosen. The following remarks should be considered when selecting the appropriate setting:

- Theoretically there should be two settings possible to capture the correct data, since the serial flash output is valid for 1 clock cycle, disregarding rise and fall times and timing uncertainties.
- Depending on the timing uncertainties, it may turn out in actual applications that only one possible sample position remains. This is subject to careful consideration depending on the actual implementation.
- The delay $t_{Del,total}$ is an absolute size to shift the point in time when the serial flash data get valid at the QuadSPI input.
- For decreasing frequency of the serial flash clock the distance between the edges increases. So for large differences in the frequency the required setting may change.
- For commands running at half of the regular serial flash clock (QSPI_SMPR[HSENA] bit set) the sampling point must be figured separately to allow for the compensation of the absolute shift in time with respect to the sample-relative setting in the QSPI_SPMPR register.

26.11.2.1.2 DQS sampling method

NOTE

This mode may not be available on this chip. See the chip-specific QuadSPI information for the read modes that this chip supports.

Data sampling in SDR mode can be supported using the DQS sampling method. Refer to Data Strobe (DQS) sampling method for more details.

26.11.2.2 DDR Mode

The increasing requirement of improved throughput has introduced the double data rate (DDR) mode. In DDR mode, the data is transferred on both the rising and falling edges of the serial flash clock. The DDR serial flashes sample as well as drive the data on both rising and falling edges of serial flash clock.

26.11.2.2.1 DQS sampling method

NOTE

This mode may not be available on this chip. See the chip-specific QuadSPI information for the read modes that this chip supports.

Data sampling in DDR mode can be supported using DQS method. Refer to Data Strobe (DQS) sampling method for more details.

26.11.3 Data Strobe (DQS) sampling method

26.11.3.1 Basic Description

In DQS mode, the data strobe signal (DQS/RWDS) is used to sample the read data. Here, both DQS and the data sent by the flash move in the same direction, so it is relatively easier to achieve at higher frequencies.

When using DQS for SDR reads, QuadSPI internally samples the incoming data on rising edge of the strobe signal. FSPHS=0 only.

The figure below shows sampling read data in SDR mode using DQS.

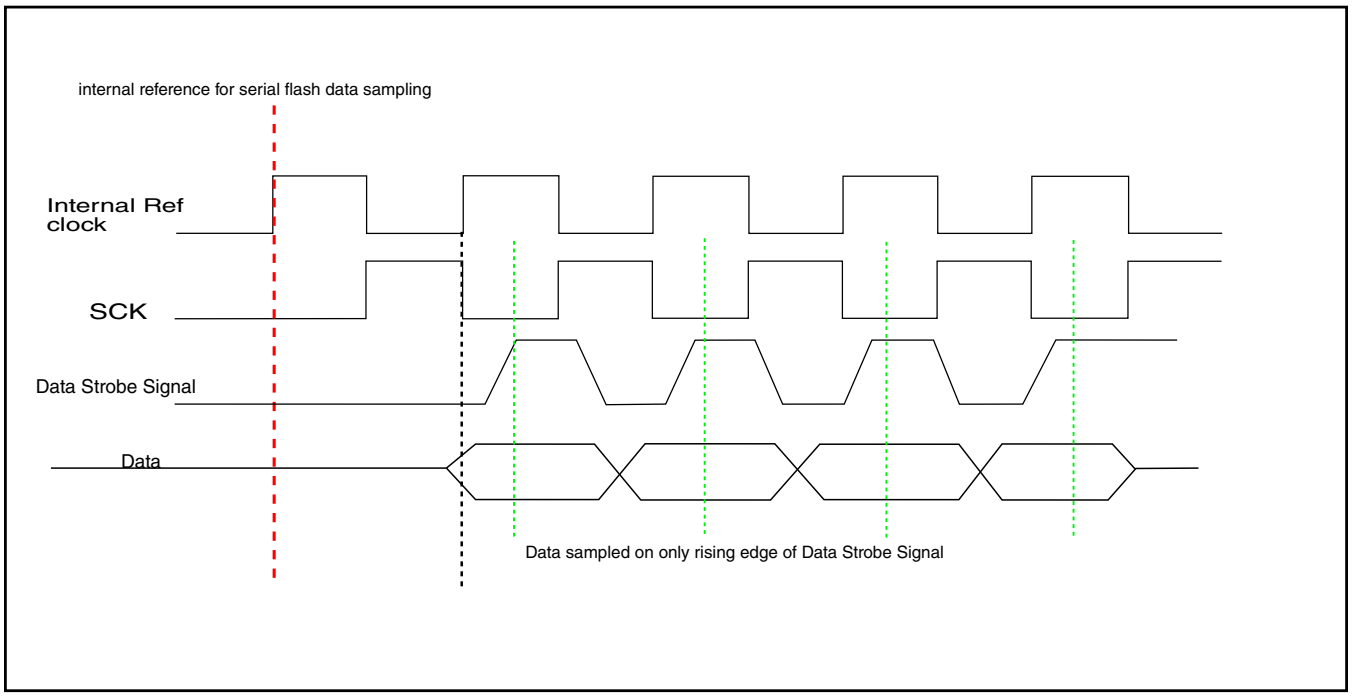


Figure 26-8. Data Strobe functionality in SDR mode

When using DQS for DDR reads, QuadSPI internally samples the incoming data on both the edges of the strobe signal. Refer to the figure below for more detail.

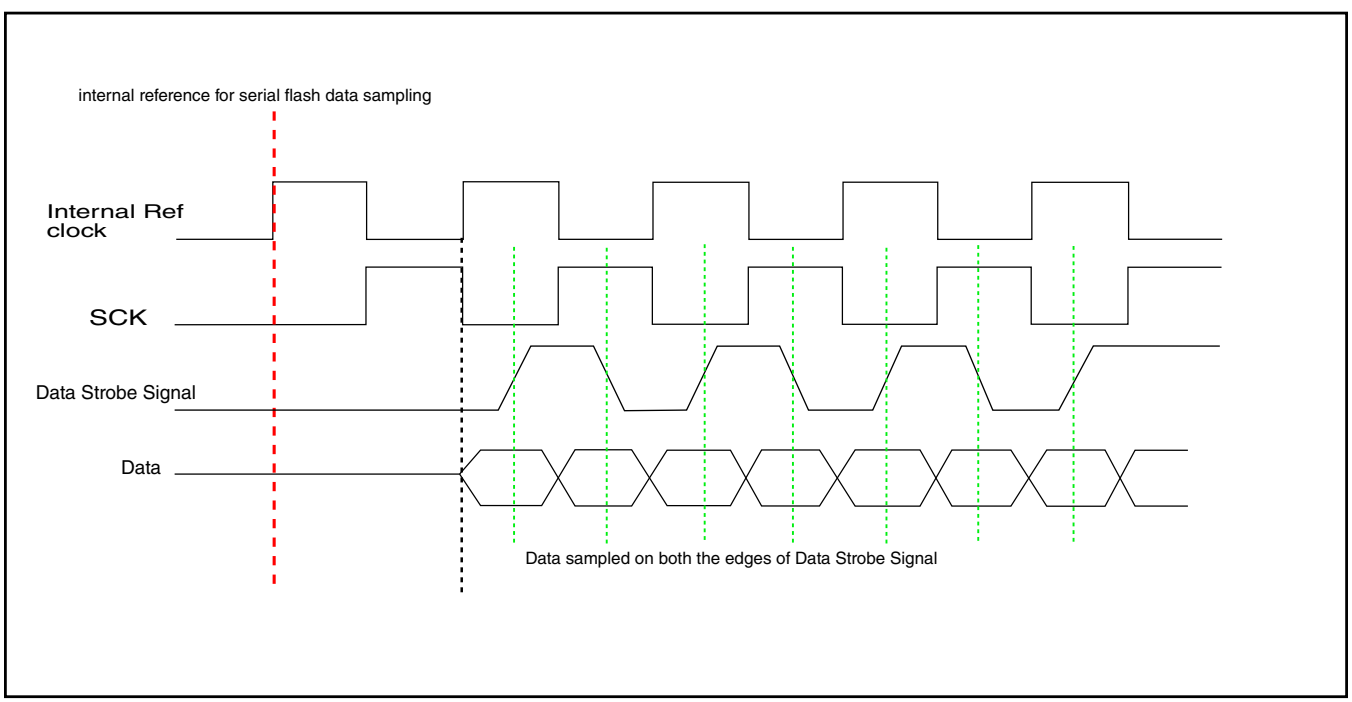


Figure 26-9. Data Strobe functionality in DDR mode

26.11.3.2 Internally generated DQS

NOTE

This mode may not be available on this chip. See the chip-specific QuadSPI information for the read modes that this chip supports.

In this mode, the internal reference clock is fed as a strobe to QuadSPI for read data sampling. The data strobe must be generated in a way such that the data is correctly sampled by the QuadSPI module. This internally generated data strobe signal can be used by QuadSPI to capture the data in:

- SDR mode
- DDR mode

Refer to QuadSPI chip-specific information for additional details about internally generated DQS.

26.11.3.3 External DQS

NOTE

This mode may not be available on this chip. See the chip-specific QuadSPI information for the read modes that this chip supports.

In serial flash memories supporting DQS, the data strobe signal is an output from the flash device that indicates when data is being transferred from the flash to the host controller. The data is then captured by the controller on:

- Only one edge (either rising or falling edge) of DQS signal in SDR mode
- Both rising/falling edge of the DQS signal in DDR mode

Chapter 27
SATA 3.0

27.1 Advanced host controller interface overview

The SATA 3.0 advanced host controller interface (AHCI) is a high-performance SATA solution that delivers comprehensive and fully-compliant generation 3 (1.5 Gb/s - 6.0 Gb/s) serial ATA capabilities, in accordance with the serial ATA revision 3.0 of Serial ATA International Organization.

SATA consists of an AHCI command layer (with a descriptor-based DMA controller) designed to operate in a system that supports command queuing. It supports FIS-based switching for improved performance in systems that employ port multipliers. Under the command layer, SATA contains the transport and link layers, as outlined in the SATA standard.

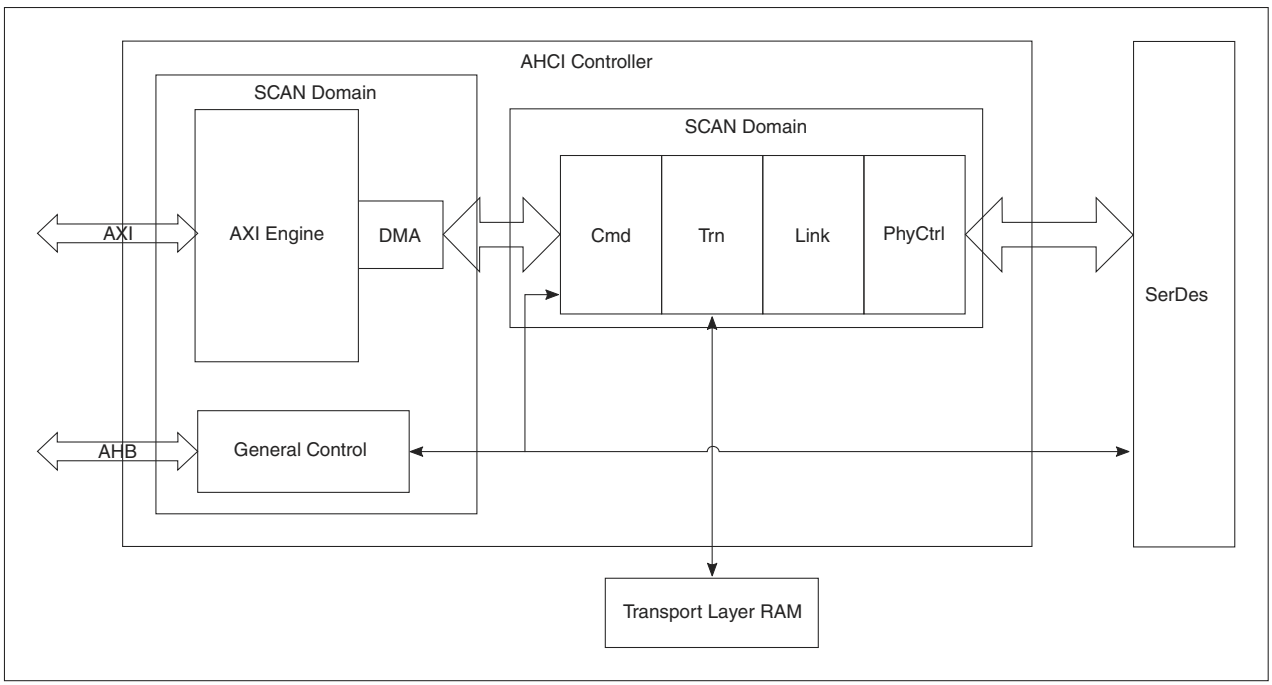


Figure 27-1. SATA controller layer overview

27.2 SATA features summary

The SATA controller includes the following features:

- Complies with the serial ATA 3.0 specification and the AHCI 1.3.1 specification
- Contains a high-speed descriptor-based DMA controller
- Supports the following:
  - Speeds of 1.5 Gb/s (first-generation SATA), 3 Gb/s (second-generation SATA), and 6 Gb/s (third-generation SATA)
  - FIS-based switching
  - Native command queuing (NCQ) commands
  - Port multiplier operation
  - Asynchronous notification
  - SATA Vendor BIST mode

27.3 SATA AHCI register descriptions

The SATA controller is compliant to AHCI 1.3.1 programming model. The SATA memory map includes only those registers which have implementation differences from AHCI Specification. For complete registers list, refer to the SATA AHCI Specification.

The registers are divided into global registers and port control registers, as shown in the table below.

Table 27-1. Register group

Start address	End address	Description
00h	2Bh	Generic host controller
A0h	FFh	Vendor-specific registers
100h	17Fh	Port 0 port control registers

27.3.1 SATA Memory map

SATA base address: 320_0000h

Offset	Register	Width (In bits)	Access	Reset value
0h	HBA capabilities register (CAP)	32	RO	E537_FF80h
4h	Global HBA control register (GHC)	32	RW	8000_0000h
10h	AHCI version register (VS)	32	RO	0001_0301h
14h	Command completion coalescing control register (CCC_CTL)	32	RW	0001_0110h
24h	HBA capabilities extended register (CAP2)	32	RO	0000_000Ch
A4h	Port config register (PCFG)	32	RW	0032_2002h
A8h	Port Phy1Cfg register (PPCFG)	32	RW	A001_FFFEh
ACh	Port Phy2Cfg register (PP2C)	32	RW	2818_4D1Bh
B0h	Port Phy3Cfg register (PP3C)	32	RW	0E08_1906h
B4h	Port Phy4Cfg register (PP4C)	32	RW	064A_0813h
B8h	Port Phy5Cfg register (PP5C)	32	RW	3FFC_96A4h
BCh	AXI cache control register (AXICC)	32	RW	0010_0010h
C0h	Port AXICfg register (PAXIC)	32	RW	0041_0102h
C8h	Port TransCfg register (PTC)	32	RW	0800_0020h
D0h	Port LinkCfg register (PLC)	32	RW	3800_FF34h
D4h	Port LinkCfg1 register (PLC1)	32	RW	0000_0000h
D8h	Port LinkCfg2 register (PLC2)	32	RW	0000_0000h
E0h	Port LinkStatus1 register (PLS1)	32	RW	0000_0000h
E4h	Port CmdConfig register (PCMDC)	32	RW	0000_0000h
E8h	Port PhyControl status register (PPCS)	32	RW	See description.

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
F0h	Timer control register (TCR)	32	RW	0000_0100h
100h	Port x command list base address register (PxCLB)	32	RW	0000_0000h
104h	Port x command list base address upper 32-bit register (PxCLBU)	32	RW	0000_0000h
108h	Port x FIS base address register (PxFB)	32	RW	0000_0000h
10Ch	Port x FIS base address upper 32-bit register (PxFBU)	32	RW	0000_0000h
110h	Port x interrupt status register (PxIS)	32	RW	0000_0000h
118h	Port x command and status register (PxCMD)	32	RW	0040_0000h
128h	Port x SATA status register (PxSSTS)	32	RO	0000_0000h
12Ch	Port x SATA control register (PxSCTL)	32	RW	0000_0300h
130h	Port x SATA error register (PxSERR)	32	RW	See description.
138h	Port x command issue register (PxCI)	32	RW	0000_0000h
140h	Port x FIS-based switching control register (PxFBS)	32	RW	0000_4000h
170h	Port 0 BIST error register (PBERR)	32	RW	0000_0003h

27.3.2 HBA capabilities register (CAP)

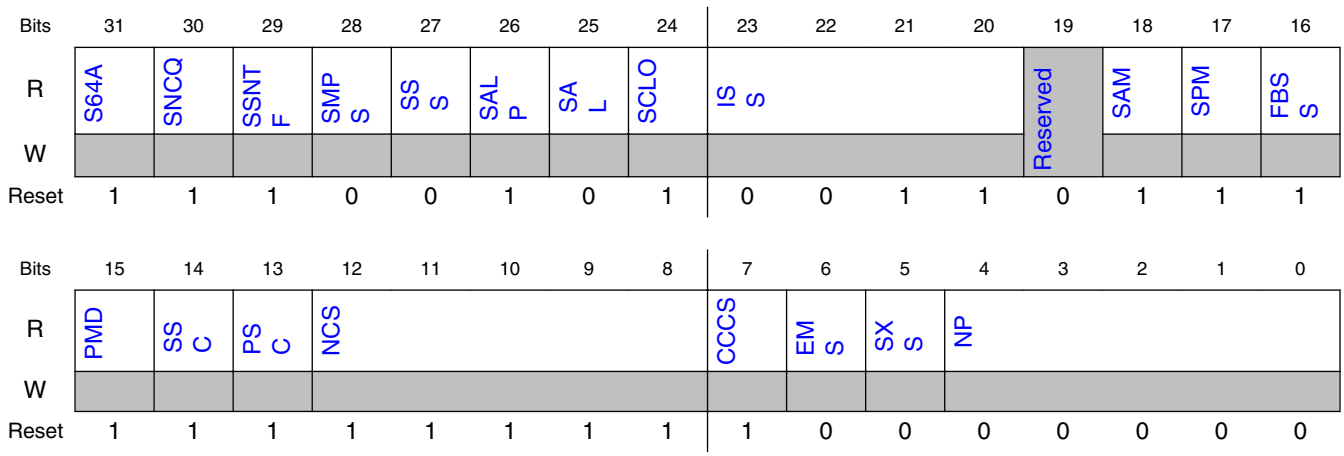
27.3.2.1 Offset

Register	Offset
CAP	0h

27.3.2.2 Function

This register indicates basic capabilities of HBA to the driver software.

27.3.2.3 Diagram



27.3.2.4 Fields

Field	Function
31 S64A	Supports 64-bit addressing Indicates whether the HBA can access 64-bit data structures. 0b - Tthe upper 32-bit bits of the port DMA descriptor, the PRD base, and each PRD entry are read-only and treated as 0 by the HBA. 1b - The HBA makes the upper 32-bit bits of the port DMA descriptor, the PRD base, and each PRD entry read/write.
30 SNCQ	Supports native command queuing Indicates whether the HBA supports SATA native command queuing. 0b - The native command queuing is not supported and software should not issue any native command queuing commands. 1b - The HBA handles DMA Setup FISes natively and the auto-activate optimization through that FIS.
29 SSNTF	Supports SNotification register 0b - The HBA does not support the PxSNTF[SNotification] register and its associated functionality. Refer to Section 10.11.1 of the SATA AHCI Specification for more details. Asynchronous notification with a directly attached device is always supported. 1b - The HBA supports the PxSNTF[SNotification] register and its associated functionality.
28 SMPS	Supports mechanical presence switch 0b - This function is not supported. This value is loaded by the BIOS prior to OS initialization. 1b - The HBA supports mechanical presence switches on its ports for use in hot plug operations.
27 SSS	Supports staggered spin-up 0b - This function is not supported. This value is loaded by the BIOS prior to OS initiallization. 1b - The HBA supports staggered spin-up on its ports, for use in balancing power spikes.
26 SALP	Supports aggressive link power management 0b - This function is not supported and software treats the PxCMD[ALPE] and PxCMD[ASP] bits as reserved. 1b - The HBA supports auto-generating link requests to the partial or slumber states if there are no commands to process.

Table continues on the next page...

Field	Function
25 SAL	Supports activity LED 0b - This function is not supported. 1b - The HBA supports a single activity indication output pin. This pin can be connected to an LED on the platform to indicate device activity on any drive.
24 SCLO	Supports command list override 0b - The HBA is not capable of clearing the PxTFD[BSY] and PxTFD[DRQ] bits in the status register to issue a software reset if these bits are still set from a previous operation. 1b - The HBA supports the PxCMD[CLO] bit and its associated function.
23-20 ISS	Interface speed support This bit indicates the maximum speed that the HBA can support on its ports. These encodings match the system software programmable PxSCTL[DET] and PxSCTL[SPD] bits. The settings not shown below are reserved. 0001b - Gen 1 (1.5 Gbps) 0010b - Gen 2 (3 Gbps) 0011b - Gen 3 (6 Gbps)
19 —	Reserved
18 SAM	Supports AHCI mode only The SATA controller may optionally support AHCI access mechanisms only. 0b - In addition to the native AHCI mechanism (through ABAR), the SATA controller implements a legacy, task-file based register interface such as SFF-8038i. 1b - The SATA controller does not implement a legacy, task-file based register interface.
17 SPM	Supports port multiplier This bit indicates whether the HBA supports a port multiplier. 0b - A port multiplier is not supported and a port multiplier may not be attached to this HBA. 1b - A port multiplier using command-based switching is supported and FIS-based switching may be supported.
16 FBSS	FIS-based switching supported This bit must be set if the SPM bit is set. 0b - The HBA does not support FIS-based switching. 1b - The HBA supports port multiplier FIS-based switching.
15 PMD	PIO multiple DRQ block In AHCI 1.2 HBAs, this bit is set to 1. 0b - The HBA only supports single DRQ block data transfers for the PIO command protocol. 1b - The HBA supports multiple DRQ block data transfers for the PIO command protocol.
14 SSC	Slumber state capable This bit indicates whether the HBA supports transitions to the slumber state. 0b - Software must neither allow the HBA to initiate transitions to the slumber state through agressive link power management nor the PxCMD[ICC] and PxSCTL[IPM] bits in each port must be programmed to disallow device initiated slumber requests. 1b - The HBA and device initiated slumber requests are supported.
13 PSC	Partial state capable This bit indicates whether the HBA can support transitions to the partial state. 0b - Software must neither allow the HBA to initiate transitions to the partial state through agressive link power management nor the PxCMD[ICC] and PxSCTL[IPM] bits in each port must be programmed to disallow device initiated partial requests. 1b - The HBA and device initiated partial requests are supported.
12-8 NCS	Number of command slots

Table continues on the next page...

Field	Function
	This bit provides value indicating the number of command slots per port supported by this HBA. A minimum of 1 and maximum of 32 slots per port can be supported. The same number of command slots is available on each implemented port.
7 CCCS	Command completion coalescing supported 0b - The HBA does not support command completion coalescing and the CCC_CTL and CCC_PORTS global HBA registers are not implemented. 1b - The HBA supports command completion coalescing as defined in Section 11 of the SATA AHCI Specification. When command completion coalescing is supported, the HBA has implemented the CCC_CTL and the CCC_PORTS global HBA registers.
6 EMS	Enclosure management supported 0b - The HBA does not support enclosure management and the EM_LOC and EM_CTL global HBA registers are not implemented. 1b - The HBA supports enclosure management as defined in Section 12 of the SATA AHCI Specification. When enclosure management is supported, the HBA has implemented the EM_LOC and EM_CTL global HBA registers.
5 SXS	Supports external SATA If this bit is set to 1, software may refer to the PxCMD[ESP] bit to determine whether a specific port has its signal connector externally accessible as a signal only connector (i.e. power is not part of that connector). 0b - The HBA has no SATA port that has a signal only connector externally accessible. 1b - This bit indicates that the HBA has one or more SATA ports that has a signal only connector that is externally accessible (for example, eSATA connector).
4-0 NP	Number of ports This bit provides value indicating the maximum number of ports supported by the HBA silicon. A maximum of 32 ports can be supported. A value of 0h, indicating one port, is the minimum requirement. **NOTE:** The number of ports indicated in this BIT may be more than the number of ports indicated in the PI register.

27.3.3 Global HBA control register (GHC)

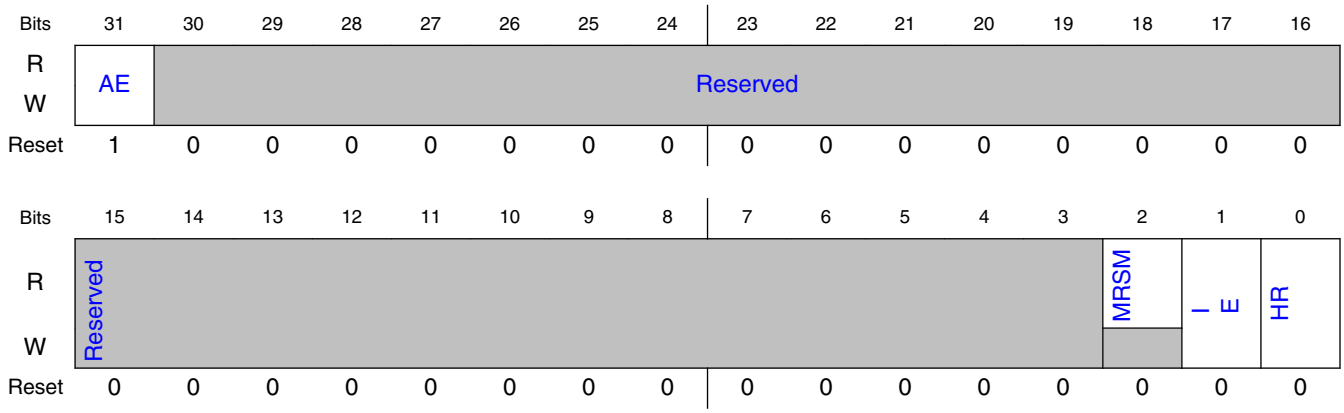
27.3.3.1 Offset

Register	Offset
GHC	4h

27.3.3.2 Function

This register controls various global actions of the HBA.

27.3.3.3 Diagram



27.3.3.4 Fields

Field	Function
31 AE	AHCI enable This is used by the HBA that supports both legacy mechanisms (such as, SFF-8038i) and AHCI to know when the HBA is running under the AHCI driver. When set, software only communicates with the HBA using AHCI. When cleared, software only communicates with the HBA using legacy mechanisms. Software sets this bit to 1 before accessing other AHCI registers. The implementation of this bit is dependent upon the value of the CAP[SAM] bit. If CAP[SAM] is 0, then GHC[AE] is read-write with a reset value of 0. If CAP[SAM] is 1, then GHC[AE] is read-only with a reset value of 1. 0b - FISes are not posted to memory and no commands are sent through AHCI mechanisms. 1b - Communication to the HBA happens through AHCI mechanisms.
30-3 —	Reserved
2 MRSM	MSI revert to single message This bit is set to 1 only when the following conditions hold: • MC[MSIE] = 1 (MSI is enabled) • MC[MMC] > 0 (multiple messages requested) • MC[MME] > 0 (more than one message allocated) • MC[MME] != MC[MMC] (messages allocated not equal to number requested) When this bit is set to 1, single MSI mode operation is in use and software is responsible for clearing bits in the IS register to clear interrupts. This bit is cleared to '0' by hardware when any of the above four conditions stated is false. This bit is also cleared to 0 when MC[MSIE] = 1 and MC[MME] = 0. In this case, the hardware has been programmed to use single MSI mode, and is not reverting to that mode. 0b - The HBA has not reverted to single MSI mode (i.e. hardware is already in single MSI mode, software has allocated the number of messages requested, or hardware is sharing interrupt vectors if MC[MME] < MC[MMC]). The HBA may revert to single MSI mode when the number of vectors allocated by the host is less than the number requested. 1b - The HBA requested more than one MSI vector but has reverted to using the first vector only.
1 IE	Interrupt enable This bit enables interrupts from the HBA.

Table continues on the next page...

Field	Function
	0b - All interrupt sources from all ports are disabled (default) 1b - Interrupts are enabled.
0 HR	HBA reset When set by software, this bit causes an internal reset of the HBA. All state machines that relate to data transfers and queuing return to an idle condition, and all ports are re-initialized through COMRESET (if staggered spin-up is not supported). If staggered spin-up is supported, then it is the responsibility of software to spin-up each port after the reset has completed. When the HBA has performed the reset action, it resets this bit to 0. A software write of 0 has no effect.

27.3.4 AHCI version register (VS)

27.3.4.1 Offset

Register	Offset
VS	10h

27.3.4.2 Function

This register indicates the major and minor version of the AHCI specification that the HBA implementation supports. The upper two bytes represent the major version number and the lower two bytes represent the minor version number.

27.3.4.3 Diagram

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

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

27.3.4.4 Fields

Field	Function
31-16	Major version number
MJR	This bit indicates that the major version is 1.
15-0	Minor version number
MNR	This bit indicates that the minor version is “3.1”.

27.3.5 Command completion coalescing control register (CCC_CTL)

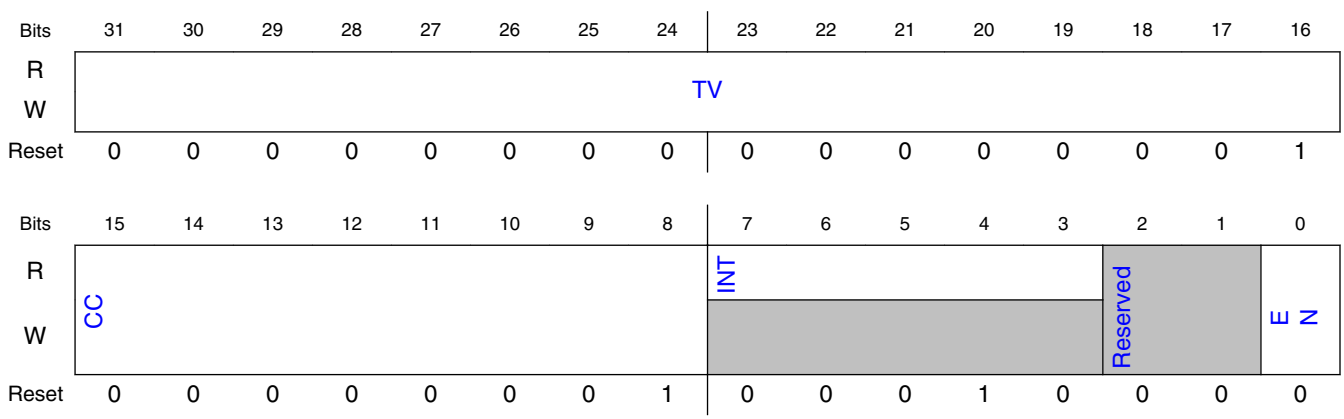
27.3.5.1 Offset

Register	Offset
CCC_CTL	14h

27.3.5.2 Function

This register configures the command completion coalescing feature for the entire HBA.

27.3.5.3 Diagram



27.3.5.4 Fields

Field	Function
31-16 TV	Timeout value The timeout value is specified in 1 ms intervals. The timer accuracy must be within 5%. hCccTimer is loaded with this timeout value. hCccTimer is only decremented when commands are outstanding on selected ports, as defined in Section 11.2 of the SATA AHCI Specification. The HBA signals a CCC interrupt when hCccTimer has decremented to 0. hCccTimer is reset to the timeout value on the assertion of each CCC interrupt. A timeout value of 0 is reserved.
15-8 CC	Command completions This bit specifies the number of command completions that are necessary to cause a CCC interrupt. The HBA has an internal command completion counter and hCccComplete. hCccComplete is incremented by one each time a selected port has a command completion. When hCccComplete is equal to the command completions value, a CCC interrupt is signaled. The internal command completion counter is reset to 0 on the assertion of each CCC interrupt. A value of 0 for this field disables CCC interrupts being generated based on the number of commands completed, which means CCC interrupts are only generated based on the timer in this case.
7-3 INT	Interrupt This bit specifies the interrupt used by the CCC feature. This interrupt must be marked as unused in the ports implemented (PI) register by setting the corresponding bit to 0. Therefore, the CCC interrupt corresponds to the interrupt for an unimplemented port on the controller. When a CCC interrupt occurs, the IS[IPS[INT]] bit is asserted to 1. This field also specifies the interrupt vector used for MSI.
2-1 —	Reserved
0 EN	Enable Software changes only the contents of the TV and CC fields when EN is cleared. When this bit is set, any updated values for the TV and CC fields take effect. 0b - The command completion coalescing feature is disabled and no CCC interrupts are generated. 1b - The command completion coalescing feature is enabled and CCC interrupts may be generated based on timeout or command completion conditions.

27.3.6 HBA capabilities extended register (CAP2)

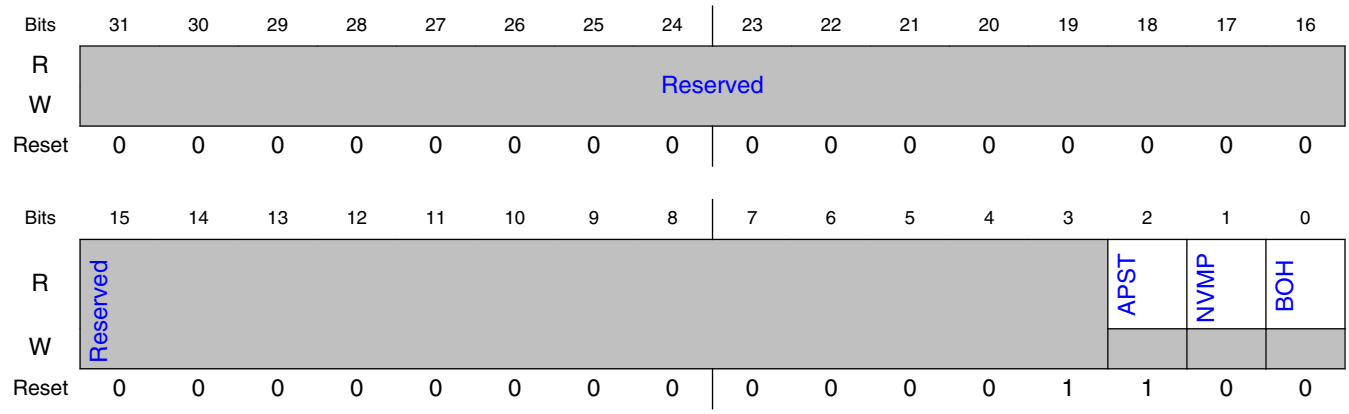
27.3.6.1 Offset

Register	Offset
CAP2	24h

27.3.6.2 Function

This register indicates capabilities of the HBA to driver software.

27.3.6.3 Diagram



27.3.6.4 Fields

Field	Function
31-3 —	Reserved
2 APST	Automatic partial to slumber transitions 0b - Automatic partial to slumber transitions are not supported. 1b - The HBA supports automatic partial to slumber transitions.
1 NVMP	NVMHCI present 0b - The HBA does not support NVMHCI. 1b - The HBA includes support for NVMHCI and the registers at offset 60h-9Fh are valid.
0 BOH	BIOS/OS handoff 0b - The HBA does not support BIOS/OS handoff and the BOHC global HBA register is not implemented. 1b - The HBA supports the BIOS/OS handoff mechanism defined in Section 10.6 of the SATA AHCI Specification and implements the BOHC global HBA register.

27.3.7 Port config register (PCFG)

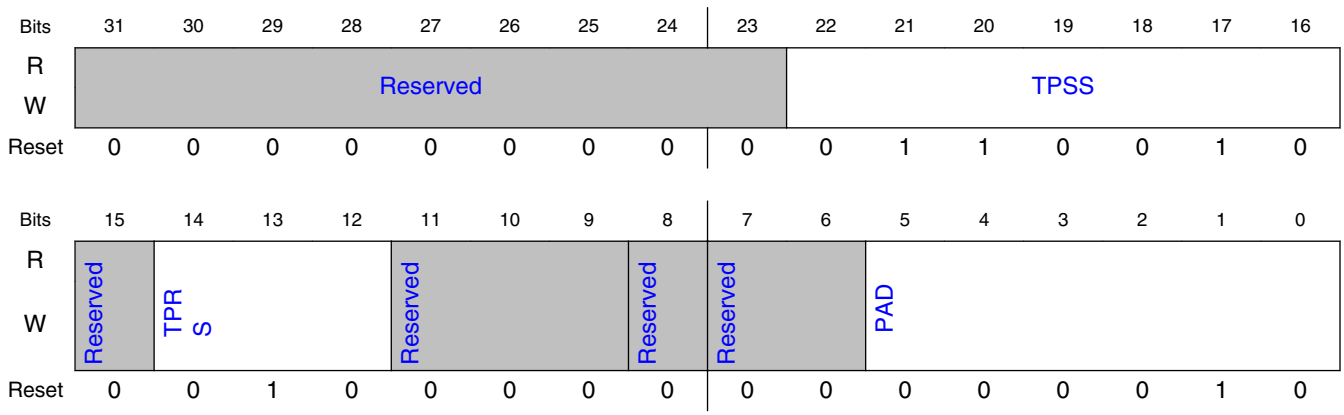
27.3.7.1 Offset

Register	Offset
PCFG	A4h

27.3.7.2 Function

This register is used to program and configure the controller port.

27.3.7.3 Diagram



27.3.7.4 Fields

Field	Function
31-23 —	Reserved
22-16 TPSS	Microsecond timer post scaler
15 —	Reserved
14-12 TPRS	Microsecond timer pre scaler
11-9 —	Reserved
8 —	Reserved
7-6 —	Reserved
5-0 PAD	Port address.The settings not defined below are reserved. 000010b - Address configuration/status register set

27.3.8 Port Phy1Cfg register (PPCFG)

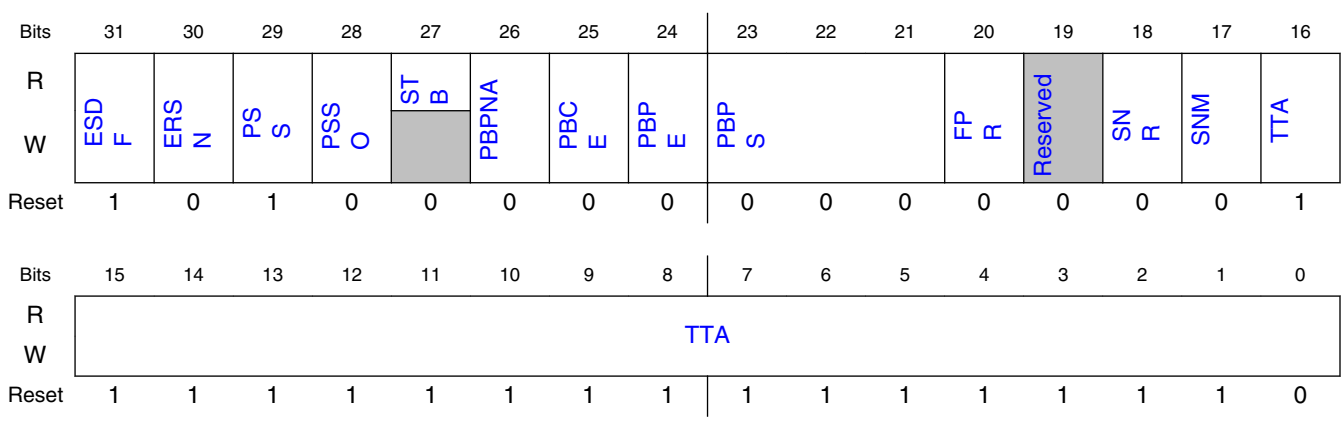
27.3.8.1 Offset

Register	Offset
PPCFG	A8h

27.3.8.2 Function

This register controls the configuration of the PHY control layer 1 for port 0. This register holds configuration values for both the OOB generator and OOB detector for each of the two OOB pulses, COMINIT/COMRESET and COMWAKE. The separate values allow multiple configurations of the clocks for running either the OOB generator or detector. The PHY control layer implements the OOB generation logic synchronous TXCLK, which is a 75 MHz input clock to the host controller. The OOB detector logic works synchronous to PMCLK, which is a fixed clock of 150 MHz and used solely for the detector clock.

27.3.8.3 Diagram



27.3.8.4 Fields

Field	Function
31 ESDF	Enable signal detect filter When set, a single de-assertion of signal detect or the coreRxDataValid, when the PHY Init state machine is in the PHYReady state, causes the state machine to exit the PHYReady state and return to a PhyNotReady state. This results in the OOB and speed negotiation running again.
30 ERSN	Enable reset speed negotiation When set, the PHY control layer enables only a single speed on the RX path during speed negotiation. This speed is determined as the fastest support for the first round falling to the lowest speed for the final round. Each round of speed negotiation is terminated by the host issuing a COMRESET and rerunning OOB before beginning the next round of speed negotiation as detailed in reset speed negotiation (RSN).
29 PSS	PhyControl SerDes slumber This bit selects SerDes slumber CMU during link slumber. When this bit is set and the controller enters slumber, an extra control signal is applied to the SerDes to slumber the clock block within SerDes. This yields an extra power savings that is SerDes specific.
28 PSSO	PhyControl select SerDes OOB This bit selects SerDes OOB or internally decoded OOB signalling as input. 0b - Select SerDes decoded OOB signalling 1b - Select internally decoded OOB signalling
27 STB	Status bit This bit provides the status of the Gen fixed clocks parameter. This bit indicates if the PHY control layer is running from a fixed frequency clock or a variable clock derived from the TX clock of the SerDes.
26 PBPNA	PhyControl BIST pattern no aligns Setting this bit causes the PHY control pattern generator to transmit each pattern continuously.
25 PBCE	PhyControl BIST clear error When a pattern mismatch occurs, this bit needs to be set and then negated to clear the error. 0b - Pattern match error bit is not cleared 1b - Clears the pattern match error bit
24 PBPE	PhyControl BIST pattern enable This bit controls enabling/disabling the PHY control test pattern generation. For more information, refer to PhyControl BIST modes 0b - Disables the PHY control test pattern generation 1b - Enables the PHY control test pattern generation
23-21 PBPS	PhyControl BIST pattern select The settings not defined below are reserved. 000b - LBP (generator clock only) 001b - LFTP 010b - MFTP 011b - HFTP 100b - PRBS pattern 101b - BIST pattern (default)
20 FPR	Force PHY Ready This bit determines how PHY Ready is driven. 0b - Normal operation mode 1b - FrcPhyRdy: In this mode, the OOB and speed negotiation states of the PHY Init state machine are bypassed. The PHY Init state machine directly enters PHY Ready after reset. The Tx buffer IDLE control is forced off.

Table continues on the next page...

Field	Function
19 —	Reserved
18 SNR	Speed negotiation rate When this bit is set set to 1, the speed negotiation runs only at the rate programmed in the PxSCTL[SPD] bit.
17 SNM	Speed negotiation method If the SATA host controller supports Gen3 speed, this bit must be set to 1. If the SATA host controller is limited to Gen2 or Gen1 speed, this bit may use the default value 0. 0b - The speed negotiation runs starting at the fastest supported speed down to the Gen1 speed. 1b - The speed negotiation runs starting at the Gen1 speed up to the fastest supported speed.
16-0 TTA	This bit determines the time period that the controller transmits and waits for ALIGNp during speed negotiation. This value is derived for the PMCLK period.

27.3.9 Port Phy2Cfg register (PP2C)

27.3.9.1 Offset

Register	Offset
PP2C	ACh

27.3.9.2 Function

This register controls the configuration of the PHY control OOB timing for the COMINIT parameters for port 0. For more info on OOB timing setup, refer to PHY control configuration register OOB timing setup.

The default value of this register is overridden by 2818_4D1F.

Table 27-2. OOB specifications

Interval	Specification value (ns)	OOB generator TXCLK (75 MHz)	OOB detector PMCLK (150 MHz)
Negate minimum	≥ 525.0	28h	-
Gap nominal	= 320.0	18h	-
Gap maximum	≤ 525.0	-	4Dh
Gap minimum[1]	≥ 200.0	-	1Fh

1. While calculating the CIBGMN value, add 25 ns to the specification value.

27.3.9.3 Diagram

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

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CIBGMX								CIBGMN							
Reset	0	1	0	0	1	1	0	1	0	0	0	1	1	0	1	1

27.3.9.4 Fields

Field	Function
31-24 CINMP	COMINIT negate minimum period OOB function: generator clock
23-16 CIBGN	COMINIT burst gap nominal OOB function: generator clock
15-8 CIBGMX	COMINIT burst gap maximum OOB function: detector clock
7-0 CIBGMN	COMINIT burst gap minimum OOB function: detector clock

27.3.10 Port Phy3Cfg register (PP3C)

27.3.10.1 Offset

Register	Offset
PP3C	B0h

27.3.10.2 Function

This register controls the configuration of the PHY control OOB timing for the COMWAKE parameters for port 0. For more info on OOB timing setup, refer to PHY control configuration register OOB timing setup.

The following table provides the OOB specifications for the chip.

Table 27-3. OOB specifications

Interval	Specification value (ns)	OOB generator TXCLK (75 MHz)	OOB detector PMCLK (150 MHz)
Negate minimum	≥ 175.0	0Eh	-
Gap nominal	= 106.6	08h	-
Gap maximum[1]	≤ 175.0	-	15h
Gap minimum	≥ 35.0	-	09h

1. While calculating the CWBGMX value, subtract 25 ns from the specification value.

27.3.10.3 Diagram

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

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CWBGMX								CWBGMN							
Reset	0	0	0	1	1	0	0	1	0	0	0	0	0	1	1	0

27.3.10.4 Fields

Field	Function
31-24 CWNMP	COMWAKE negate minimum period OOB function: generator clock
23-16 CWBGN	COMWAKE burst gap nominal OOB function: generator clock
15-8 CWBGMX	COMWAKE burst gap maximum OOB function: detector clock

Table continues on the next page...

Field	Function
7-0 CWBGMN	COMWAKE burst gap minimum OOB function: detector clock

27.3.11 Port Phy4Cfg register (PP4C)

27.3.11.1 Offset

Register	Offset
PP4C	B4h

27.3.11.2 Function

This register controls the configuration of the PHY control burst timing for the COM parameters for port 0. For more info on OOB timing setup, refer to PHY control configuration register OOB timing setup.

The following table provides the OOB specifications for the chip.

Table 27-4. OOB specifications

Interval	Specification value (ns)	OOB generator TXCLK (75 MHz)	OOB detector PMCLK (150 MHz)
Signal failure detection	-	-	4Ah
Burst nominal	= 106.6	08h	-
Burst maximum	≤ 109.9 non 113.33	-	0Fh

27.3.11.3 Diagram

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

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BNM								BMX							
Reset	0	0	0	0	1	0	0	0	0	0	0	1	0	0	1	1

27.3.11.4 Fields

Field	Function
31-24 PTST	Partial to slumber timer This bit provides a value that specifies the delay that the controller should apply while in partial before entering slumber. The value is based on the system clock divided by 128, total delay = (system clock period) * PTST * 128.
23-16 SFD	Signal failure detection OOB function: detector clock If the signal detection de-asserts for a time greater than this, the OOB detector determines this as a line idle and causes the PhyInit state machine to exit the PHY Ready state.
15-8 BNM	COM burst nominal OOB function: generator clock
7-0 BMX	COM burst maximum OOB function: detector clock

27.3.12 Port Phy5Cfg register (PP5C)

27.3.12.1 Offset

Register	Offset
PP5C	B8h

27.3.12.2 Function

This register controls the configuration of the PHY control retry interval timing for port 0. For more info on OOB timing setup, refer to PHY control configuration register OOB timing setup.

Table 27-5. OOB specifications

Interval	Specification value	Calculated value for PMCLK (150 MHz)
Rate change timer	≥ 54.2 µs / 4	03FFh
Retry interval	≥ 10.0 ms	0C96A8h

27.3.12.3 Diagram

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

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

27.3.12.4 Fields

Field	Function
31-20 RCT	Rate change timer This bit provides a value based on the 54.2 µs for which a SATA device transmits at a fixed rate ALIGNp after OOB has completed. For a fast SerDes, this value is recommended to be 54.2 µs / 4. OOB function: detector clock
19-0 RIT	Retry interval timer The calculated value is divided by two and the lower digit of precision is not needed. OOB function: detector clock

27.3.13 AXI cache control register (AXICC)

27.3.13.1 Offset

Register	Offset
AXICC	BCh

27.3.13.2 Function

This register controls the value of the AWCACHE and ARCACHE used to distinguish each address operation on the address bus.

AWCACHE/ARCACHE control (Non AHCI standard)

Both AWCACHE and ARCACHE can be controlled during header, command, and status of the data phase AXI bus operations. The values driven onto these signals are controlled by the AXI cache control register for operations related to the header, physical region description, command, and status operations. The values driven onto these signals during a data operation can be further controlled from the relevant PRDT entry as shown in Table 27-13. Note that this control is not an AHCI standard.

The following table summarizes each of the possible attributes that can be set through the cache bits.

Table 27-6. Attributes list

ARCACHE[3:0]/AWCACHE[3:0]				Transaction attributes
WA	RA	C	B	
0	0	0	0	Non-cacheable and non-bufferable
0	0	0	1	Bufferable only
0	0	1	0	Cacheable, but do not allocate
0	0	1	1	Cacheable and bufferable, but do not allocate
0	1	0	0	Reserved
0	1	0	1	Reserved
0	1	1	0	Cacheable write-through, allocate on reads only

Table continues on the next page...

Table 27-6. Attributes list (continued)

ARCACHE[3:0]/AWCACHE[3:0]				Transaction attributes
0	1	1	1	Cacheable write-back, allocate on reads only
1	0	0	0	Reserved
1	0	0	1	Reserved
1	0	1	0	Cacheable write-through, allocate on writes only
1	0	1	1	Cacheable write-back, allocate on writes only
1	1	0	0	Reserved
1	1	0	1	Reserved
1	1	1	0	Cacheable write-through, allocate on both reads and writes
1	1	1	1	Cacheable write-back, allocate on both reads and writes

27.3.13.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved		EARC	EAWC	AWCF				AWCD				AWCFD			
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ARCP				ARCH				ARCF				ARCA			
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

27.3.13.4 Fields

Field	Function
31-30 —	Reserved
29	Enable ARCACHE

Table continues on the next page...

Field	Function
EARC	This bit provides the control from the PRDT entries used during the data phase of this operation (global control).
28 EAWC	Enable AWCACHE This bit provides the control from the PRDT entries used during the data phase of this operation (global control).
27-24 AWCF	Address write cache FIS This bit provides the value driven onto AWCACHE when the AXI master is posting a status FIS write address to the memory controller.
23-20 AWCD	Address write cache data This bit provides the value driven onto AWCACHE when the AXI master is posting a data burst write address to the memory controller when the data burst is not the final burst in the transfer.
19-16 AWCFD	Address write cache final data This bit provides the value driven onto AWCACHE when the AXI master is posting a data burst write address to the memory controller when the data burst is the final burst in the transfer.
15-12 ARCP	Address read cache PRD This bit provides the value driven onto ARCACHE when the AXI master is posting a PRD read address to the memory controller.
11-8 ARCH	Address read cache header This bit provides the value driven onto ARCACHE when the AXI master is posting a header or physical region descriptor read address to the memory controller.
7-4 ARCF	Address read cache FIS This bit provides the value driven onto ARCACHE when the AXI master is posting a command FIS read address to the memory controller.
3-0 ARCA	Address read cache ATAPI This bit provides the value driven onto ARCACHE when the AXI master is posting a data burst.

27.3.14 Port AXICfg register (PAXIC)

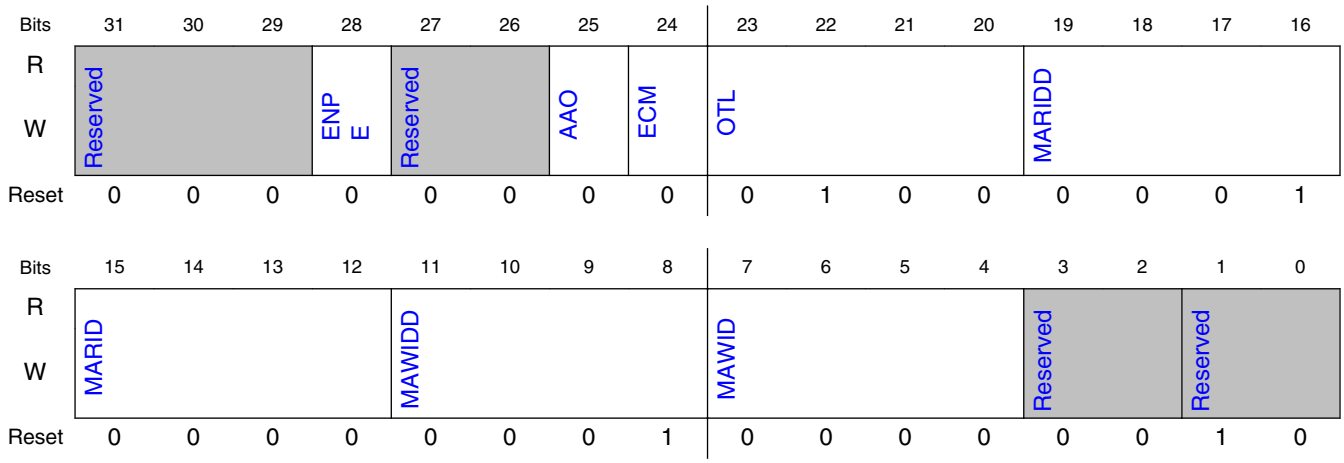
27.3.14.1 Offset

Register	Offset
PAXIC	C0h

27.3.14.2 Function

This register controls the configuration of the AXI bus operation for port 0. The AXIPE and ADBW bits can only be configured through access to port 0 configuration space.

27.3.14.3 Diagram



27.3.14.4 Fields

Field	Function
31-29 —	Reserved
28 ENPE	Enable non-zero 4 MB PRD entries
27-26 —	Reserved
25 AAO	Allow address overwrite This bit allows the PxCLB[9:0] and PxFB[7:0] bits to be overwritten.
24 ECM	Enable the context management This bit enables the context management system in the memory.
23-20 OTL	Outstanding transfer limit This bit limits the maximum number of outstanding transfers supported. • For the 264 DWord transport layer implementation, this can be programmed between 1 and 16 • For the 136 DWord transport layer implementation, this can be programmed between 1 and 8 • For the 72 DWord transport layer implementation, this can be programmed between 1 and 4 The settings not defined below are reserved. 0000b - Transfers limited by the transport layer FIFO space/fill level.
19-16 MARIDD	Memory address read ID This bit provides memory address read ID for data transfers.
15-12 MARID	Memory address read ID This bit provides memory address read ID for non data transfers.

Table continues on the next page...

Field	Function
11-8 MAWIDD	Memory address write ID This bit provides memory address write ID for data transfers.
7-4 MAWID	Memory address write ID This bit provides memory address write ID for non data transfers.
3-2 —	Reserved
1-0 —	Reserved

27.3.15 Port TransCfg register (PTC)

27.3.15.1 Offset

Register	Offset
PTC	C8h

27.3.15.2 Function

This register controls the configuration of the transport layer for port 0.

27.3.15.3 Diagram

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

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved						ITM	ENBD	Reserved	RXWM						
Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

27.3.15.4 Fields

Field	Function
31-10 —	Reserved
9 ITM	Initialize transport memories 0b - The machine runs writing 0 to all locations within the memory 1b - Causes the memory initialize hardware state machine to enter reset
8 ENBD	Enable back down When a port multiplier is attached and an attempt to send a command to an attached drive results in three consecutive retries due to R_ERR receptions, the command removes for the transport layer and returns to the pending queue. This is to avoid unnecessary retries owing to the device trying to send a set device bits FIS while the host is trying to send a command. This feature improves performance as it allows the controller to queue any pending commands to the other drives.
7 —	Reserved
6-0 RXWM	RxWaterMark This bit sets the minimum number of free location within the RX FIFO before the watermark is exceeded. The transport layer in turn instructs the link layer to transmit HOLDS to the transmitting end. **NOTE:** It can take some time for the HOLDs to get to the other end and in the interim period there must be enough room in the FIFO to absorb all data that could arrive. An initial value of 7'h29 is recommended.

27.3.16 Port LinkCfg register (PLC)

27.3.16.1 Offset

Register	Offset
PLC	D0h

27.3.16.2 Function

This register controls the configuration of the link layer for port 0.

27.3.16.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PMPRA					POE	PRT									
Reset	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	AIR								EPNRT	S4A	RXSE	TXSE	TXPJ	TXC	RXBC	TXBC
Reset	1	1	1	1	1	1	1	1	0	0	1	1	0	1	0	0

27.3.16.4 Fields

Field	Function
31-27 PMPRA	Power management primitive rate acknowledge This bit determines the number of PMACK primitives sent when a power management state transition is requested by the host.
26 POE	Primitive override enable When set, this bit enables the replacement of a single primitive, as specified by override primitive/CD, when the link layer state machine is in the Prim override state. This bit must be set to enable this feature.
25-16 PRT	PHY Ready timer This bit specifies the timeout value of the PHY Ready timer. If EnPhyReadyTimeOut is set, the link layer counts down on every rising edge of TX clock, as long as PHY Ready is de-asserted. When the counter reaches zero, a PhyReset is issued to the PHY to try and re-establish communications with the far-end. The timer is initially loaded with a value equal to the concatenation of { PHY Ready Timer, 9'h000}.
15-8 AIR	ALIGN insertion rate The SATA AHCI Specification requires that the link layer sends a pair of ALIGN primitives at least every 254 DWords of data. This is achieved by setting ALIGN insertion rate to "11111111". However, for test purposes it is possible to send ALIGNs at a higher rate. This can be achieved by setting ALIGN insertion rate to a lower value i.e. (ALIGN insertion rate-1). DWords are sent by the link layer between each set of ALIGN primitive pairs. **NOTE:** If the S4A bit is set, the ALIGN insertion rate should not be set to four or less. If S4A is not set, the ALIGN insertion rate should not be set to two or less.
7 EPNRT	Enable PHY not ready timer If PHY Ready is de-asserted for a long time as specified by PRT bit, then this bit, when asserted, enables the link layer to re-issue a PhyReset, thereby re-initiating OOB.
6 S4A	Send 4 Aligns If this bit is asserted, four ALIGN primitives are transmitted at the specified rate, instead of the normal two ALIGN primitives.
5 RXSE	Rx scramble enable

Table continues on the next page...

Field	Function
	If this bit is asserted, the de-scrambling of the receive data is enabled as per the SATA AHCI Specification.
4 TXSE	Tx scramble enable If this bit is asserted, the scrambling of the transmit data is enabled as per the SATA AHCI Specification.
3 TXPJ	Tx Prim junk If this bit is de-asserted, the scrambled junk data is sent after a CONT primitive, as per the SATA AHCI Specification. If this bit is asserted, then the single character 32'hDEADBEEF is sent continuously to aid debug.
2 TXC	Tx CONT If this bit is asserted, the transmission of CONT primitives is enabled. If de-asserted, then long sequences of repeated primitives can be sent by the link layer.
1 RXBC	Rx bad CRC When a rising edge is detected on this bit, it causes a bad CRC to be detected for the current frame. This bit has must be set to enable this feature.
0 TXBC	Tx bad CRC A bad CRC (inverted value of the correct CRC) value is transmitted for one FIS only by the link layer when a rising edge is detected on this signal. This bit must be set to enable this feature.

27.3.17 Port LinkCfg1 register (PLC1)

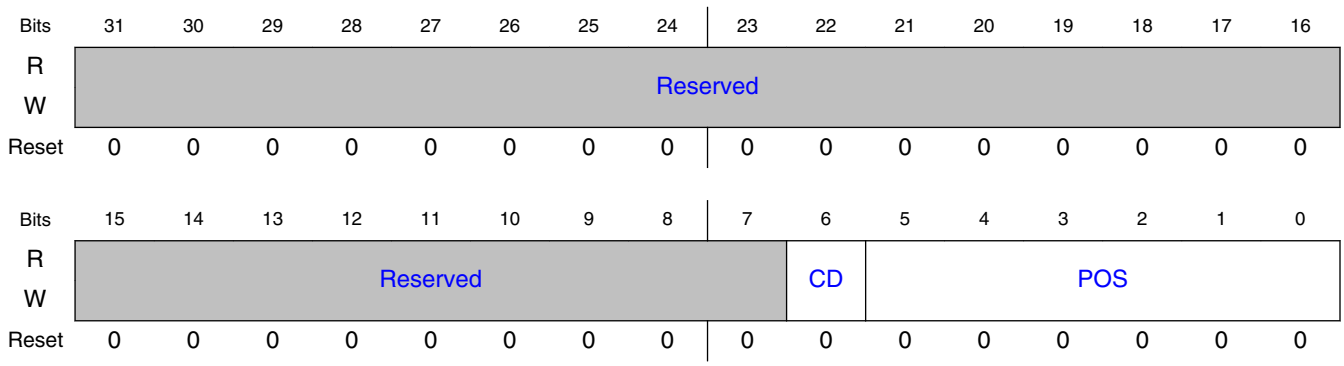
27.3.17.1 Offset

Register	Offset
PLC1	D4h

27.3.17.2 Function

This register controls the configuration of the link layer for port 0.

27.3.17.3 Diagram



27.3.17.4 Fields

Field	Function
31-7 —	Reserved
6 CD	Data character or primitive This bit specifies whether the data used during the primitive override should be a data character or a primitive. For example, if CD = 1, Prim override state = L_SendEOF and override primitive = WTRM, then a WTRM primitive is inserted into the data stream instead of an EOF (whenever a rising edge is seen on PLC[POE]. If CD = 0, then a normal data character (as specified by PLC2[OP]) is inserted into the data stream instead of the EOF.
5-0 POS	Primitive override state This bit is used in the primitive override debug functionality. When the link layer detects a positive edge on PLC[POE], it overrides the next primitive that would be inserted during the PLC1[POS], with the data specified by the PLC2[OP] and PLC1[CD] configuration bits.

27.3.18 Port LinkCfg2 register (PLC2)

27.3.18.1 Offset

Register	Offset
PLC2	D8h

27.3.18.2 Function

This register controls the configuration of the link layer for port 0.

27.3.18.3 Diagram

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

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

27.3.18.4 Fields

Field	Function
31-0 OP	Override primitive This bit specifies the data to be used in the primitive override debug functionality that is described in the definition of Port LinkCfg1 register (PLC1).

27.3.19 Port LinkStatus1 register (PLS1)

27.3.19.1 Offset

Register	Offset
PLS1	E0h

27.3.19.2 Function

This register indicates the status of the link layer for port 0. This register acts as an accumulator for the SerDes errors. Each counter can be cleared by writing 8'hFF to the individual byte.

27.3.19.3 Diagram

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

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CEC								DEC							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

27.3.19.4 Fields

Field	Function
31-24 KCEC	Kchar error count This bit provides the number of DWords that have been received from the PHY, where one or more control character errors have been detected. A value of 255 indicates an error count of 255 or more as this counter does not wrap around to zero. The count value is updated with its current value each time the LinkStatus1 register is read.
23-16 PIEC	PHY internal error count This bit provides the number of DWords that have been received from the PHY, where one or more internal errors have been detected. A value of 255 indicates an error count of 255 or more as this counter does not wrap around to zero. The count value is updated with its current value each time the LinkStatus1 register is read.
15-8 CEC	Code error count This bit provides the number of DWords that have been received from the PHY, where one or more code errors have been detected. A value of 255 indicates an error count of 255 or more as this counter does not wrap around to zero. The count value is updated with its current value each time the LinkStatus1 register is read.
7-0 DEC	Disparity error count This bit provides the number of DWords that have been received from the PHY, where one or more disparity errors have been detected. A value of 255 indicates an error count of 255 or more as this counter does not wrap around to zero. The count value is updated with its current value each time the LinkStatus1 register is read.

27.3.20 Port CmdConfig register (PCMDC)

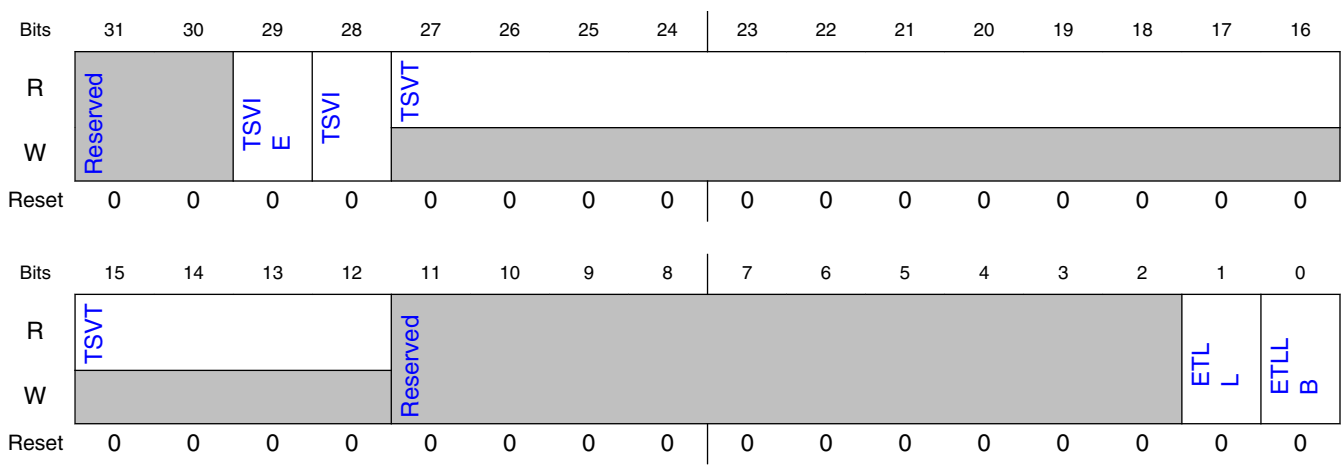
27.3.20.1 Offset

Register	Offset
PCMDC	E4h

27.3.20.2 Function

This register controls the operation of the command layer for port 0.

27.3.20.3 Diagram



27.3.20.4 Fields

Field	Function
31-30 —	Reserved
29 TSVIE	Trustzone slave ID violation interrupt enable The interrupt is seen on bit 3 of the interrupt status register in the general register group. Refer to the SATA AHCI Specification for more details.
28	Trustzone slave ID violation interrupt

Table continues on the next page...

Field	Function
TSVI	
27-12 TSVT	Trustzone Slave ID of violating transaction When the slave port rejects a read or write as slave error due to security violation, this register records the AXI ID of the violating transaction.
11-2 —	Reserved
1 ETLL	Enable transport layer loopback
0 ETLLB	Enable transport layer loopback in the BIST-L mode This bit must be set to 1 before performing the BIST-L test.

27.3.21 Port PhyControl status register (PPCS)

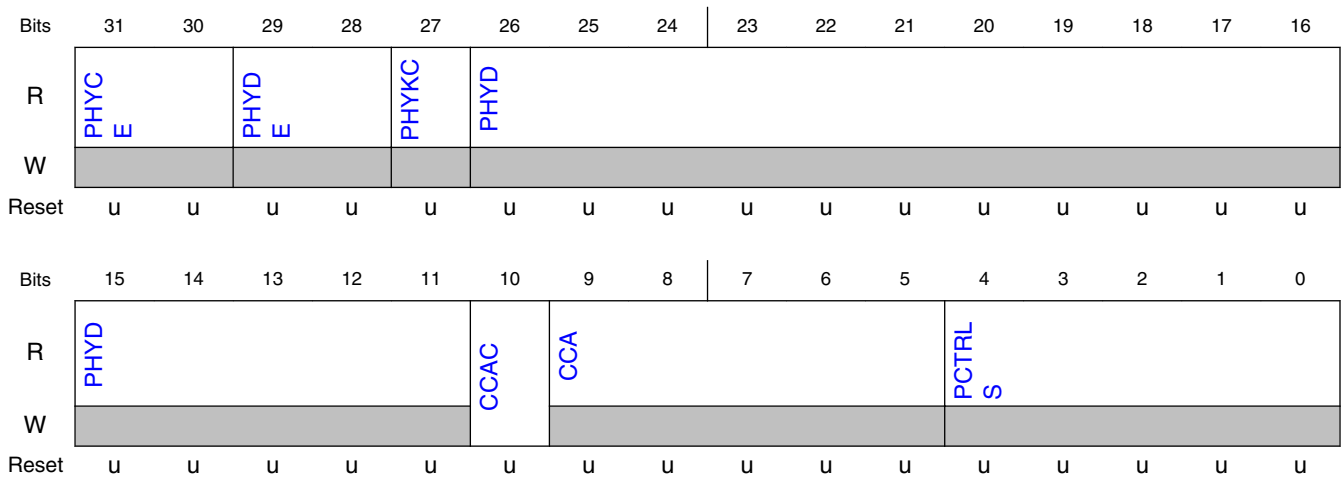
27.3.21.1 Offset

Register	Offset
PPCS	E8h

27.3.21.2 Function

This register controls the operation of the status of the phy control layer for port 0.

27.3.21.3 Diagram



27.3.21.4 Fields

Field	Function
31-30 PHYCE	Current 2-bit code error This bit provides snapshot within the PHY control layer.
29-28 PHYDE	Current 2-bit disparity error This bit provides snapshot within the PHY control layer.
27 PHYKC	Current 1-bit K character This bit provides snapshot within the PHY control layer.
26-11 PHYD	Current 16-bit data This bit provides snapshot within the PHY control layer.
10 CCAC	Comma alignment has changed
9-5 CCA	Current comma alignment
4-0 PCTRLS	PHY control state

27.3.22 Timer control register (TCR)

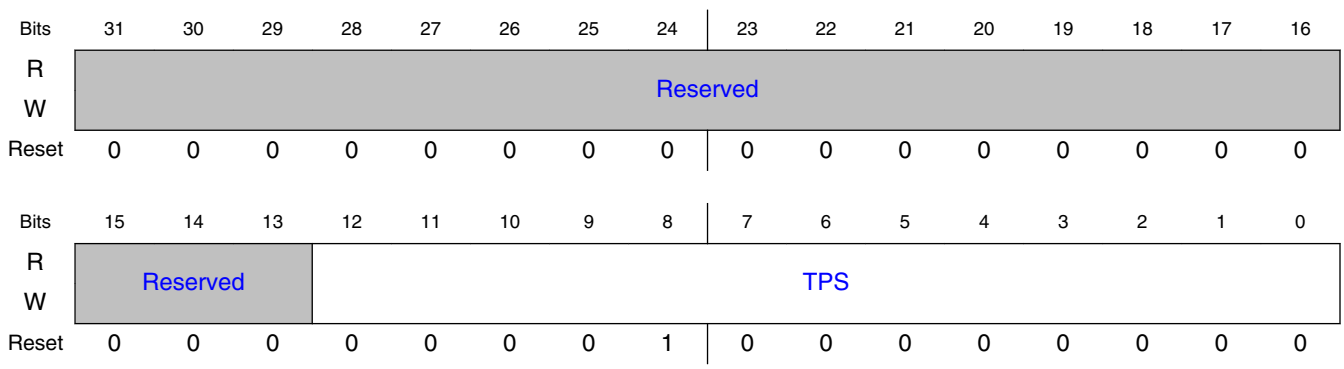
27.3.22.1 Offset

Register	Offset
TCR	F0h

27.3.22.2 Function

This register controls the operation of the timer prescaler that configures a 10 µs pulse generator used to control the operation of the slumber timer. This pulse generator is used in port 0.

27.3.22.3 Diagram



27.3.22.4 Fields

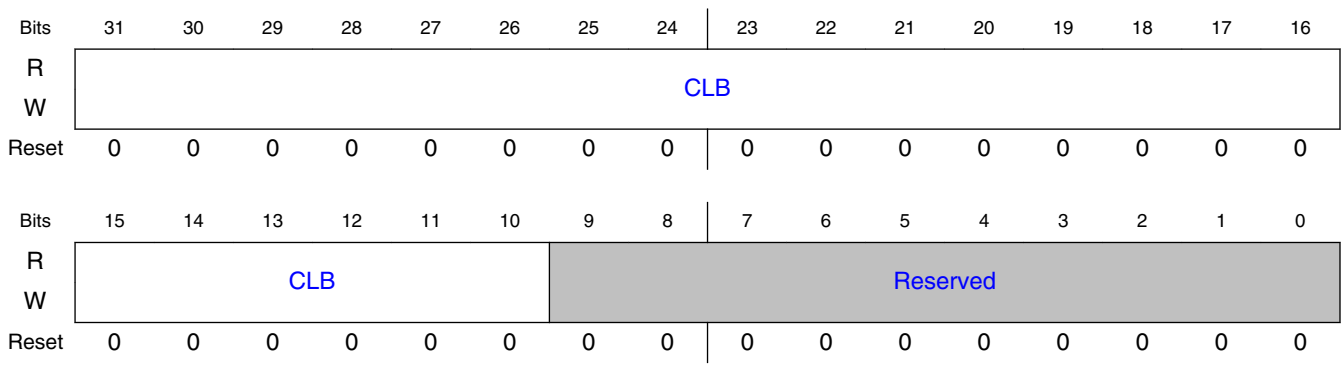
Field	Function
31-13 —	Reserved
12-0 TPS	Timer preScalar value The system clock is divided by the ratio to generate a 10 µs clock pulse to run the port layers.

27.3.23 Port x command list base address register (PxCLB)

27.3.23.1 Offset

Register	Offset
PxCLB	100h

27.3.23.2 Diagram



27.3.23.3 Fields

Field	Function
31-10 CLB	Command list base address This bit 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.
9-0 —	Reserved

27.3.24 Port x command list base address upper 32-bit register (PxCLBU)

27.3.24.1 Offset

Register	Offset
PxCLBU	104h

27.3.24.2 Diagram

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

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

27.3.24.3 Fields

Field	Function
31-0 CLBU	Command list base address upper This bit indicates the upper 32 bits for the command list base physical address for this port. This base is used when fetching commands to execute. This bit is read only '0' for HBAs that do not support 64-bit addressing.

27.3.25 Port x FIS base address register (PxFB)

27.3.25.1 Offset

Register	Offset
PxFB	108h

27.3.25.2 Diagram

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

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	FB								Reserved							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

27.3.25.3 Fields

Field	Function
31-8 FB	FIS base address This bit indicates the 32-bit base physical address for received FISes. The structure pointed to by this address range is 256 bytes in length. This address must be 256 bytes aligned as indicated by bits [7:0] being read only. When FIS-based switching is in use, this structure is 4 KB in length and the address is 4 KB aligned.
7-0 —	Reserved

27.3.26 Port x FIS base address upper 32-bit register (PxFBU)

27.3.26.1 Offset

Register	Offset
PxFBU	10Ch

27.3.26.2 Diagram

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

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

27.3.26.3 Fields

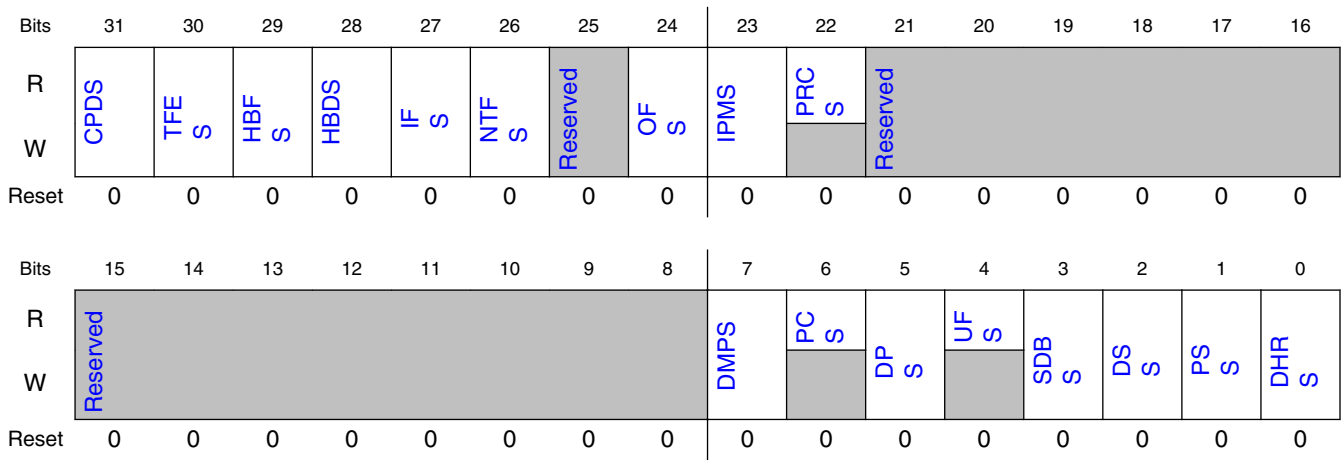
Field	Function
31-0 FBU	FIS base address upper This bit indicates the upper 32 bits for the received FIS base physical address for this port. This register is read only '0' for HBAs that do not support 64-bit addressing.

27.3.27 Port x interrupt status register (PxIS)

27.3.27.1 Offset

Register	Offset
PxIS	110h

27.3.27.2 Diagram



27.3.27.3 Fields

Field	Function
31 CPDS	Cold port detect status When set, a device status has changed as detected by the cold presence detect logic. This bit can either be set due to a non-connected port receiving a device or a connected port having its device removed. This bit is only valid if the port supports cold presence detect as indicated by PxCMD[CPD] set to 1.
30 TFES	Task file error status This bit is set whenever the status register is updated by the device and the error bit (bit 0 of the Status field in the received FIS) is set.
29 HBFS	Host bus fatal error status This bit indicates that the HBA encountered a host bus error that it cannot recover from, such as a bad software pointer. In PCI, such an indication would be a target or master abort.
28 HBDS	Host bus data error status This bit indicates that the HBA encountered a data error (uncorrectable ECC/parity) when reading from or writing to system memory.
27 IFS	Interface fatal error status This bit indicates that the HBA encountered an error on the SATA interface which caused the transfer to stop. Refer to Section 6.1.2 of the SATA AHCI Specification fore more details.
26 NTFS	Interface non-fatal error status This bit indicates that the HBA encountered an error on the SATA interface but was able to continue operation. Refer to Section 6.1.2 of the SATA AHCI Specification for more details.
25 —	Reserved
24 OFS	Overflow status This bit indicates that the HBA received more bytes from a device than was specified in the PRD table for the command.

Table continues on the next page...

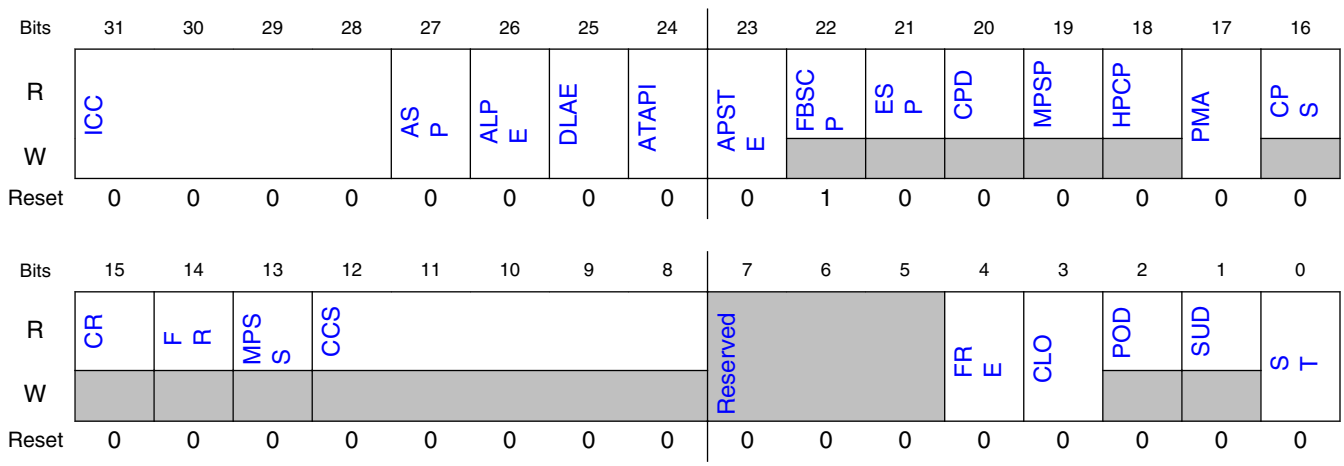
Field	Function
23 IPMS	Incorrect port multiplier status This bit indicates that the HBA received a FIS from a device that did not have a command outstanding. This bit is set during enumeration of devices on a port multiplier due to the normal port multiplier enumeration process. It is recommended that IPMS only be used after enumeration is complete on the port multiplier. IPMS is not set when an asynchronous notification is received (a set device bits FIS with the notification 'N' bit set to 1).
22 PRCS	PhyRdy change status When set, this bit indicates the internal PhyRdy signal changed state. This bit reflects the state of PxSERR[DIAG[N]]. To clear this bit, software must clear PxSERR[DIAG[N]] to 0.
21-8 —	Reserved
7 DMPS	Device mechanical presence status When set, this bit indicates that a mechanical presence switch associated with this port has been opened or closed, which may lead to a change in the connection state of the device. This bit is only valid if both CAP[SMPS] and PxCMD[MPSP] are set to 1.
6 PCS	Port connect change status This bit reflects the state of PxSERR[DIAG[X]]. This bit is only cleared when PxSERR[DIAG[X]] is cleared. 0b - No change in current connect status 1b - Change in current connect status
5 DPS	Descriptor processed A PRD with the 'I' bit set has transferred all of its data. Refer to Section 5.4.2 of the SATA AHCI Specification for more details.
4 UFS	Unknown FIS interrupt When set, this bit indicates that an unknown FIS was received and has been copied into system memory. This bit is cleared to 0 by software clearing the PxSERR[DIAG[F]] bit to 0. **NOTE:** This bit does not directly reflect the PxSERR[DIAG[F]] bit. PxSERR[DIAG[F]] is set immediately when an unknown FIS is detected, whereas this bit is set when that FIS is posted to memory. Software should wait to act on an unknown FIS until this 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. This bit is set even if 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.

27.3.28 Port x command and status register (PxCMD)

27.3.28.1 Offset

Register	Offset
PxCMD	118h

27.3.28.2 Diagram



27.3.28.3 Fields

Field	Function
31-28 ICC	Interface communication control This bit is used to control power management states of the interface. If the Link layer is currently in the L_IDLE state, writes to this field causes the HBA to initiate a transition to the interface power management state requested. If the Link layer is not currently in the L_IDLE state, writes to this field has no effect. When system software writes a non-reserved value other than No-Op (0h), the HBA performs the action and update this bit back to Idle (0h). If software writes to this bit 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 HBA takes no action and returns this bit to Idle. If the interface is in a low power state and software wants to transition to a different low power state, software must first bring the link to active and then initiate the transition to the desired low power state. The settings not defined below are reserved. 0000b - No-Op/Idle: When software reads this value, it indicates the HBA is ready to accept a new interface control command, although the transition to the previously selected state may not yet have occurred. 0001b - Active: Causes the HBA to request a transition of the interface into the active state. 0010b - Partial: Causes the HBA to request a transition of the interface to the partial state. The SATA device may reject the request and the interface remains in its current state. 0110b - Slumber: Causes the HBA to request a transition of the interface to the slumber state. The SATA device may reject the request and the interface must remain in its current state.

Table continues on the next page...

Field	Function
27 ASP	Aggressive slumber/partial If CAP[SALP] is cleared to 0, software treats this bit as reserved. Refer to Section 8.3.1.3 of the SATA AHCI Specification for more details. 0b - When ALPE is set, the HBA aggressively enters the partial state when it clears the PxCI register and the PxSACT register is cleared or when it clears the PxSACT register and the PxCI register is cleared. 1b - When ALPE is set, the HBA aggressively enters the slumber state when it clears the PxCI register and the PxSACT register is cleared or when it clears the PxSACT register and the PxCI register is cleared.
26 ALPE	Aggressive link power management enable Software only sets this bit to 1 if CAP[SALP] is set to 1. If CAP[SALP] is cleared to 0, software treats this bit as reserved. Refer to Section 8.3.1.3 of the SATA AHCI Specification for more details. 0b - The aggressive link power management is disabled 1b - The HBA aggressively enters a lower link power state (partial or slumber) based upon the setting of the ASP bit.
25 DLAE	Drive LED on ATAPI enable Refer to Section 10.11 of the SATA AHCI Specification for more details. **NOTE:** The controller does not support ATAPI functionality. 0b - The HBA only drives the LED pin active for commands if PxCMD[ATAPI] is set to 0. 1b - The HBA drives the LED pin active for commands regardless of the state of PxCMD[ATAPI].
24 ATAPI	Device is ATAPI This bit is used by the HBA to control whether or not to generate the desktop LED when commands are active. Refer to Section 10.11 of the SATA AHCI Specification for more details. **NOTE:** The controller does not support ATAPI functionality. 0b - The connected device is not an ATAPI device 1b - The connected device is an ATAPI device.
23 APSTE	Automatic partial to slumber transitions enabled Software only sets this bit to 1 if CAP2[APST] is set to 1. If CAP2[APST] is cleared to 0, software treats this bit as reserved. 0b - The HBA does not perform automatic partial to slumber transitions. 1b - The HBA performs automatic partial to slumber transitions.
22 FBSCP	FIS-based switching capable port This bit may only be set to 1 if both CAP[SPM] and CAP[FBSS] are set to 1. 0b - This port does not support FIS-based switching. 1b - This port supports port multiplier FIS-based switching.
21 ESP	External SATA port When this bit is set to 1, the CAP[SXS] bit is set to 1. ESP is mutually exclusive with the HPCP bit in this register. If ESP is set to 1, then the port may experience hot plug events. 0b - This port's signal connector is not externally accessible on a signal only connector. 1b - This port's signal connector is externally accessible on a signal only connector (e.g. eSATA connector).
20 CPD	Cold presence detection When this bit is set to 1, PxCMD[HPCP] must also be set to 1. 0b - Platform does not support cold presence detection on this port. 1b - Platform supports cold presence detection on this port.
19 MPSP	Mechanical presence switch attached to port When this bit is set to 1, PxCMD[HPCP] should also be set to 1. 0b - The platform does not support a mechanical presence switch attached to this port.

Table continues on the next page...

Field	Function
	1b - The platform supports the mechanical presence switch attached to this port.
18 HPCP	Hot plug capable port HPCP is mutually exclusive with the ESP bit in this register. 0b - Indicates that this port's signal and power connectors are not externally accessible through a joint signal and power connector. 1b - Indicates that this port's signal and power connectors are externally accessible through a joint signal and power connector for blindmate device hot plug.
17 PMA	Port multiplier attached This bit is read/write for HBAs that support a port multiplier (CAP[SPM] = 1). This bit is read-only for HBAs that do not support a port multiplier (CAP[SPM] = 0). Software is responsible for detecting whether a port multiplier is present; hardware does not auto-detect the presence of a port Multiplier. Software only sets this bit to 1 when PxCMD[ST] is cleared to 0. 0b - A port multiplier is not attached to the HBA for this port. 1b - A port multiplier is attached to the HBA for this port.
16 CPS	Cold presence state The CPS bit reports whether a device is currently detected on this port through cold presence detection. 0b - The HBA detects through cold presence that there is no device attached to this port. 1b - The HBA detects through cold presence that a device is attached to this port.
15 CR	Command list running When set, the command list DMA engine for the port is running. Refer to AHCI state machine in Section 5.3.2 of the SATA AHCI Specification for more details.
14 FR	FIS receive running When set, the FIS receive DMA engine for the port is running. Refer to Section 10.3.2 of the SATA AHCI Specification for more details.
13 MPSS	Mechanical presence switch state This bit reports the state of a mechanical presence switch attached to this port. If CAP[SMPS] is set to 0, this bit is cleared to 0. Software uses this bit only if both CAP[SMPS] and PxCMD[MPSP] are set to 1. 0b - If CAP[SMPS] is set to 1, the mechanical presence switch is closed. 1b - If CAP[SMPS] is set to 1, the mechanical presence switch is open.
12-8 CCS	Current command slot This bit is valid when PxCMD[ST] is set to 1 and is set to the command slot value of the command that is currently being issued by the HBA. When PxCMD[ST] transitions from 1 to 0, this bit resets to 0. After PxCMD[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 PxCMD[CCS] + 1. For example, after the HBA has issued its first command, if CCS = 0h and PxCI is set to 3h, the next command that will be issued is from command slot 1.
7-5 —	Reserved
4 FRE	FIS receive enable System software must not set this bit until PxFB (and PxFBU) have been programmed with a valid pointer to the FIS receive area. If software wishes to move the base, this bit must first be cleared, and software must wait for the FR bit in this register to be cleared. Refer to Section 10.3.2 of the SATA AHCI Specification for more details. 0b - Received FISes are not accepted by the HBA, except for the first D2H register FIS after the initialization sequence, and no FISes are posted to the FIS receive area. 1b - The HBA posts received FISes into the FIS receive area pointed to by PxFB (and for 64-bit HBAs, PxFBU).
3	Command list override

Table continues on the next page...

Field	Function
CLO	This bit is only set to 1 immediately prior to setting the PxCMD[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. Software must wait for CLO to be cleared to 0 before setting PxCMD[ST] to 1. 0b - The HBA sets this bit to 0 when PxTFD[STS[BSY]] and PxTFD[STS[DRQ]] have been cleared to 0. A write to this register with a value of 0 has no effect. 1b - Causes PxTFD[STS[BSY]] and PxTFD[STS[DRQ]] to be cleared to 0. This allows a software reset to be transmitted to the device regardless of whether the BSY and DRQ bits are still set in the PxTFD[STS] register. The HBA sets this bit to 0 when PxTFD[STS[BSY]] and PxTFD[STS[DRQ]] have been cleared to 0. A write to this register with a value of 0 has no effect. This bit is only set to 1 immediately prior to setting the PxCMD[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. Software must wait for CLO to be cleared to 0 before setting PxCMD[ST] to 1.
2 POD	Power-on device This bit is read/write for HBAs that support cold presence detection on this port as indicated by PxCMD[CPD] set to 1. This bit is read only '1' for HBAs that do not support cold presence detect. When set, the HBA sets the state of a pin on the HBA to 1 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 for HBAs that support staggered spin-up through CAP[SSS]. This bit is read only '1' for HBAs that do not support staggered spin-up. On an edge detect from 0 to 1, the HBA starts a COMRESET initialization sequence to the device. Clearing this bit to 0 does not cause any OOB signal to be sent on the interface. When this bit is cleared to 0 and PxSCTL[DET] = 0h, the HBA enters listen mode as detailed in Section 10.10.1 of the SATA AHCI Specification.
0 ST	Start When this bit is changed from 0 to 1, the HBA starts processing the command list at entry 0. When this bit is changed from 1 to 0, the PxCI register is cleared by the HBA upon the HBA putting the controller into an idle state. This bit is only set to 1 by software after PxCMD[FRE] has been set to 1. Refer to Section 10.3.1 of the SATA AHCI Specification for more details. 0b - The HBA does not process the command list. 1b - The HBA processes the command list.

27.3.29 Port x SATA status register (PxSSTS)

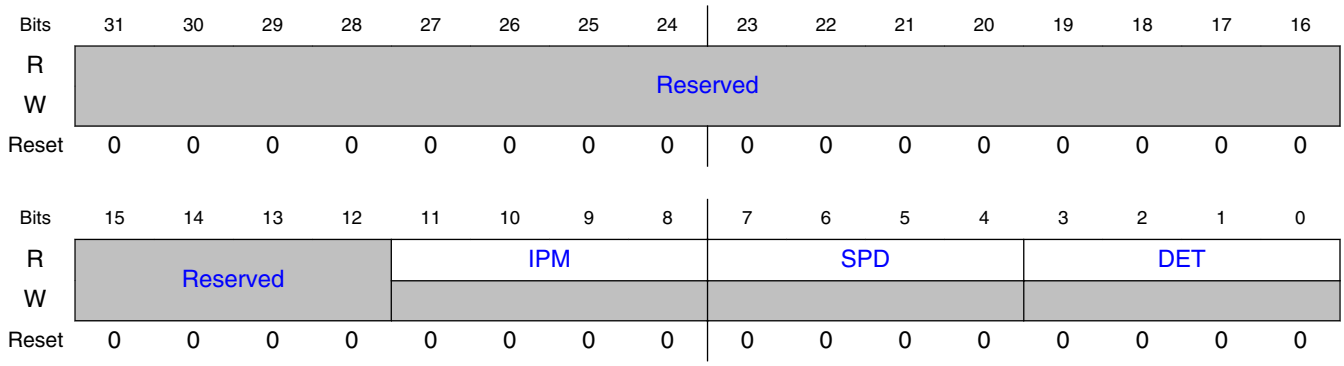
27.3.29.1 Offset

Register	Offset
PxSSTS	128h

27.3.29.2 Function

This register conveys the current state of the interface and host. The HBA updates it continuously and asynchronously. When the HBA transmits a COMRESET to the device, this register is updated to its reset values.

27.3.29.3 Diagram



27.3.29.4 Fields

Field	Function
31-12 —	Reserved
11-8 IPM	Interface power management This bit indicates the current interface state. The settings not defined below are reserved. 0000b - Device not present or communication not established 0001b - Interface in active state 0010b - Interface in partial power management state 0110b - Interface in slumber power management state
7-4 SPD	Current interface speed This bit indicates the negotiated interface communication speed. The settings not defined below are reserved. 0000b - Device not present or communication not established 0001b - Generation 1 communication rate negotiated 0010b - Generation 2 communication rate negotiated 0011b - Generation 3 communication rate negotiated
3-0 DET	Device detection This bit indicates the interface device detection and PHY state. The means by which the implementation determines device presence may be vendor specific. However, device presence must always be indicated anytime a COMINIT signal is received. The settings not defined below are reserved. 0000b - No device detected and PHY communication not established 0001b - Device presence detected but PHY communication not established 0011b - Device presence detected and PHY communication established 0100b - PHY in offline mode as a result of the interface being disabled or running in a BIST loopback mode

27.3.30 Port x SATA control register (PxSCTL)

27.3.30.1 Offset

Register	Offset
PxSCTL	12Ch

27.3.30.2 Function

The register is used by software to control SATA capabilities. Writes to this register result in an action being taken by the host adapter or interface. Reads from the register return the last value written to it.

27.3.30.3 Diagram

Bits	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

Bits	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	1	1	0	0	0	0	0	0	0	0

27.3.30.4 Fields

Field	Function
31-12 —	Reserved
11-8 IPM	Interface power management transitions allowed Indicates which power states the HBA is allowed to transition to. If an interface power management state is disabled, the HBA is not allowed to initiate that state and the HBA must PMNAK$_P$ any request from the device to enter that state. The settings not defined below are reserved. 0000b - No interface restrictions 0001b - Transitions to the partial state is disabled 0010b - Transitions to the slumber state is disabled

Table continues on the next page...

Field	Function
	0011b - Transitions to both partial and slumber states are disabled
7-4 SPD	Speed allowed This bit indicates the highest allowable speed of the interface. The settings not defined below are reserved. 0000b - No speed negotiation restrictions 0001b - Limit speed negotiation to Generation 1 communication rate 0010b - Limit speed negotiation to a rate not greater than Generation 2 communication rate 0011b - Limit speed negotiation to a rate not greater than Generation 3 communication rate
3-0 DET	Device detection initialization This bit controls the HBA's device detection and interface initialization. This bit may only be modified when PxCMD[ST] is 0. Changing this bit while the PxCMD[ST] bit is set to 1 results in an undefined behavior. When PxCMD[ST] is set to 1, this field should have a value of 0h. **NOTE:** It is permissible to implement any of the SATA defined behaviors for transmission of COMRESET when DET = 1h. The settings not defined below are reserved. 0000b - No device detection or initialization action requested 0001b - Perform interface communication initialization sequence to establish communication. This is functionally equivalent to a hard reset and results in the interface being reset and communications reinitialized. While this field is 1h, COMRESET is transmitted on the interface. Software should leave the DET field set to 1h for a minimum of 1 ms to ensure that a COMRESET is sent on the interface. 0100b - Disable the SATA interface and put PHY in offline mode

27.3.31 Port x SATA error register (PxSERR)

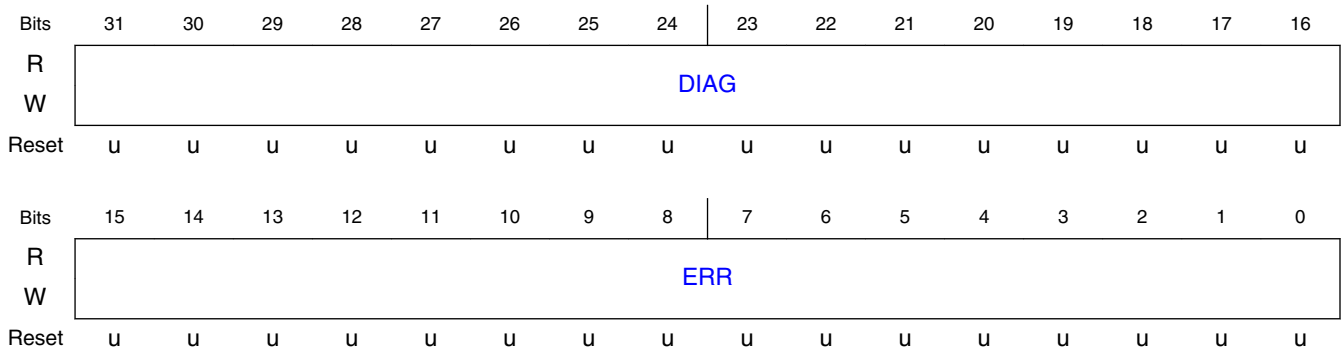
27.3.31.1 Offset

Register	Offset
PxSERR	130h

27.3.31.2 Function

This register contains the diagnostics (DIAG) and error (ERR) bits that contains diagnostic error information and error information.

27.3.31.3 Diagram



27.3.31.4 Fields

Field	Function
31-16 DIAG	Diagnostic This bit contains diagnostic error information for use by diagnostic software in validating correct operation or isolating failure modes.

Table 27-7. Diagnostic error information

Bit	Description
31:27	Reserved
26	Exchanged (X): When set to 1, this bit indicates that a change in device presence has been detected since the last time this bit was cleared. The means by which the implementation determines that the device presence has changed is vendor specific. This bit must always be set to 1 anytime a COMINIT signal is received. This bit is reflected in the PxIS[PCS] bit.
25	Unknown FIS Type (F): Indicates that one or more FISes were received by the transport layer with good CRC, but had a type field that was not recognized.
24	Transport state transition error (T): Indicates that an error has occurred in the transition from one state to another within the transport layer since the last time this bit was cleared.
23	Link Sequence Error (S): Indicates that one or more link state machine error conditions was encountered. The link layer state machine defines the conditions under which the link layer detects an erroneous transition.
22	Handshake Error (H): Indicates that one or more R_ERR handshake response was received in response to frame transmission. Such errors may

Table continues on the next page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
Table 27-7. Diagnostic error information (continued)
<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td></td><td>be the result of a CRC error detected by the recipient, a disparity or 8b/10b decoding error, or other error condition leading to a negative handshake on a transmitted frame.</td></tr>
<tr><td>21</td><td>CRC Error (C): Indicates that one or more CRC errors occurred with the link layer.</td></tr>
<tr><td>20</td><td>Disparity Error (D): This field is not used by AHCI.</td></tr>
<tr><td>19</td><td>10B to 8B Decode Error (B): Indicates that one or more 10B to 8B decoding errors occurred.</td></tr>
<tr><td>18</td><td>Comm Wake (W): Indicates that a Comm Wake signal was detected by the PHY.</td></tr>
<tr><td>17</td><td>PHY Internal Error (I): Indicates that the PHY detected some internal error.</td></tr>
<tr><td>16</td><td>PhyRdy Change (N): Indicates that the PhyRdy signal changed state. This bit is reflected in the PxIS[PRCS] bit.</td></tr>
</table>
</td></tr>
<tr><td>15-0
ERR</td><td>Error
This bit contains error information for use by host software in determining the appropriate response to the error condition.
Table 27-8. Error information
<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>15:12</td><td>Reserved</td></tr>
<tr><td>11</td><td>Internal Error (E): The host bus adapter experienced an internal error that caused the operation to fail and may have put the host bus adapter into an error state. The internal error may include a master or target abort when attempting to access system memory, an elasticity buffer overflow, a primitive mis-alignment, a synchronization FIFO overflow, and other internal error conditions. Typically when an internal error occurs, a non-fatal or fatal status bit in the PxIS register is also set to give software guidance on the recovery mechanism required.</td></tr>
<tr><td>10</td><td>Protocol Error (P): A violation of the SATA protocol was detected.</td></tr>
<tr><td>9</td><td>Persistent Communication or Data Integrity Error (C): A communication error, which was not recovered, occurred that is expected to be persistent. Persistent communications errors may arise from faulty interconnect with the device, from a device that has been removed or has failed, or a number of other causes.</td></tr>
</table>
</td></tr>
</table>

Field	Function
	Table 27-8. Error information (continued)

Bit	Description
8	Transient Data Integrity Error (T): A data integrity error occurred that was not recovered by the interface.
7:2	Reserved
1	Recovered Communications Error (M): Communications between the device and host was temporarily lost but was re-established. This can arise from a device temporarily being removed, from a temporary loss of PHY synchronization, or from other causes and may be derived from the PhyNRdy signal between the PHY and Link layers.
0	Recovered Data Integrity Error (I): A data integrity error occurred that was recovered by the interface through a retry operation or other recovery action.

27.3.32 Port x command issue register (PxCI)

27.3.32.1 Offset

Register	Offset
PxCI	138h

27.3.32.2 Diagram

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

Bits	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

27.3.32.3 Fields

Field	Function
31-0 CI	Command issued This bit is bit significant. Each bit corresponds to a command slot, where bit 0 corresponds to command slot 0. This bit is set by software to indicate to the HBA that a command has been built in system memory for a command slot and may be sent to the device. When the HBA 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 must only be set to 1 by software when PxCMD[ST] is set to 1. This field is also cleared when PxCMD[ST] is written from a 1 to a 0 by software.

27.3.33 Port x FIS-based switching control register (PxFBS)

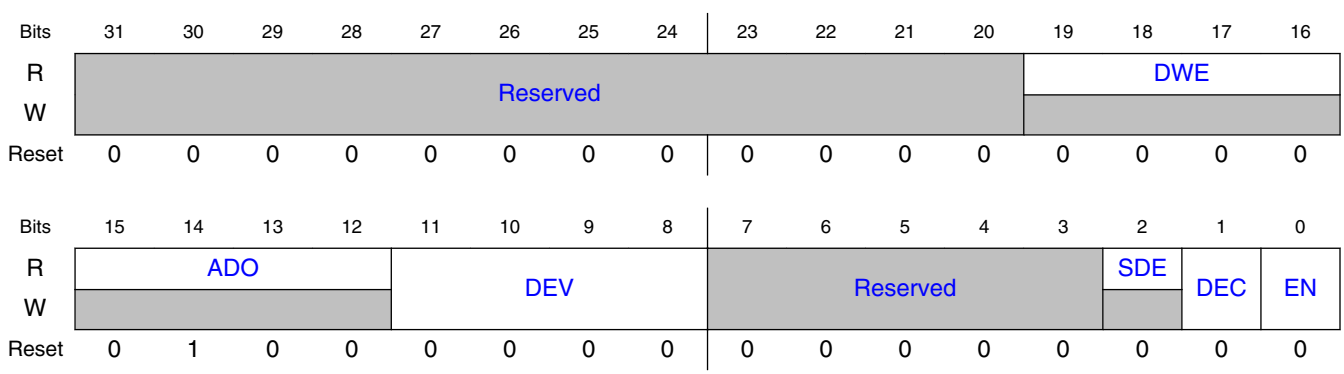
27.3.33.1 Offset

Register	Offset
PxFBS	140h

27.3.33.2 Function

This register controls and obtains the status for port multiplier FIS-based switching.

27.3.33.3 Diagram



27.3.33.4 Fields

Field	Function
31-20 —	Reserved
19-16 DWE	Device with error Set by hardware to the value of the port multiplier port number of the device that experienced a fatal error condition. This bit is only valid when PxFBS[SDE] is set to 1.
15-12 ADO	Active device optimization This bit exposes the number of active devices that the FIS-based switching implementation has been optimized for. When there are more devices active than indicated in this bit, throughput of concurrent traffic may degrade. For optimal performance, software should limit the number of active devices based on this value. The minimum value for this bit should be 2h, indicating that at least two devices may be active with high performance maintained.
11-8 DEV	Device To issue Set by software to the port multiplier port value of the next command to issue. This field enables hardware to know the port the command is to be issued to without fetching the command header. Software must not issue commands to multiple port multiplier ports on the same write of the PxCI register.
7-3 —	Reserved
2 SDE	Single device error This bit is cleared on PxFBS[DEC] being set to 1 or on PxCMD[ST] being cleared to 0. 0b - When a fatal error condition has occurred, the error applies to the entire port. To clear the error, PxCMD[ST] must be cleared to 0 by software. 1b - When a fatal error condition has occurred, hardware believes the error is localized to one device such that software's first error recovery step should be to utilize the PxFBS[DEC] functionality.
1 DEC	Device error clear Software only sets this bit to 1 if PxFBS[EN] is set to 1 and PxFBS[SDE] is set to 1. 0b - When hardware has completed error recovery actions, hardware clears the bit to 0. Write to 0 by software has no effect. 1b - The HBA clears the device-specific error condition and the HBA flushes any commands outstanding for the device that experienced the error, including clearing the PxCI and PxSACT bits for that device to 0.
0 EN	Enable Software only changes the value of the EN bit when PxCMD[ST] is cleared to 0. 0b - FIS-based switching is not being used. 1b - A port multiplier is attached and the HBA uses FIS-based switching to communicate with it.

27.3.34 Port 0 BIST error register (PBERR)

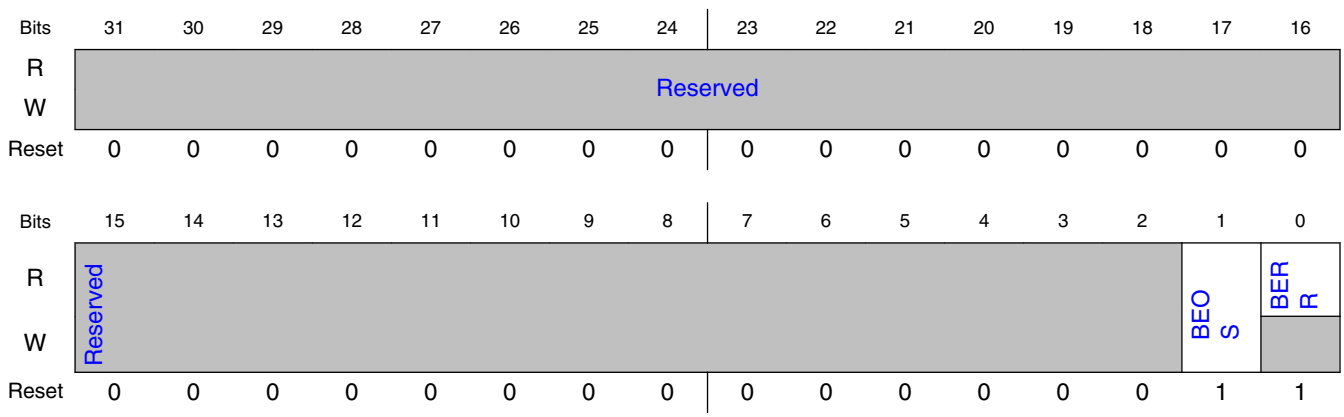
27.3.34.1 Offset

Register	Offset
PBERR	170h

27.3.34.2 Function

The BIST operation as programmed by SATA must report the operation as passing or failing. This register reflects the status of the BIST operation running in the link layer.

27.3.34.3 Diagram



27.3.34.4 Fields

Field	Function
31-2 —	Reserved
1 BEOS	BIST error one shot bit 0b - BIST operation passed 1b - BIST operation failed
0 BERR	BIST error 0b - BIST operation passing 1b - BIST operation failing

27.4 Command layer

The operation of the command layer is defined by the AHCI specification.

27.4.1 Local port context management

When the AHCI controller is connected to a port multiplier supporting FIS-based switching, a local context store can be enabled to avoid the process of lookup of the related memory addressing for data transactions. This feature facilitates quick context switching and allows the AHCI to operate with multiple devices in a seamless manner. Each context stores information about the memory address and SATA block address of any executing command on the first four ports of a port multiplier.

27.4.2 Vendor-specific BIST operation

As part of the host self-diagnostic operation, a vendor-specific BIST mode is supported. This mode, in conjunction with a SerDes that supports serial loop-back, allows for the test of the host controller operation. When programmed, the host fetches a command from memory loop that commands FIS through the transport and the link layers and then posts the payload to the receive FIS area for checking. This mode exercises the following paths.

- DMA controller FIS transmission
- Command layer FIS transmission
- Transport layer TX FIFO FIS transmission
- Link layer FIS transmission
- PHY modes
- Link layer FIS reception
- Transport layer Rx FIFO and FIS reception
- Command layer FIS reception
- Host DMA controller FIS reception

To run this self-test, the software needs to perform the following operation:

1. Build the vendor-specific header and descriptor as detailed out in the Vendor Specific BIST Header and Descriptor. For more information, refer Table 27-10.
2. Place the PHY in the loop-back mode. For more information, refer to SerDes module chapter for the PHY loop.
3. Set PPCFG[FPR] to 1. Issue this command to the host controller.

When the host controller indicates the command has completed, the software examines the contexts of the command descriptors stats field. If the preprogrammed values are present, it considers the test passed.Vendor-specific BIST header, descriptor, and register settings are described in the subsequent sections.

27.4.2.1 Command list structure

To support the vendor BIST operation, the command header structure has been modified. The DW0[11] bit has been designated as VBIST. Setting this bit indicates that the associated command is to be used as the payload for a vendor BIST operation.

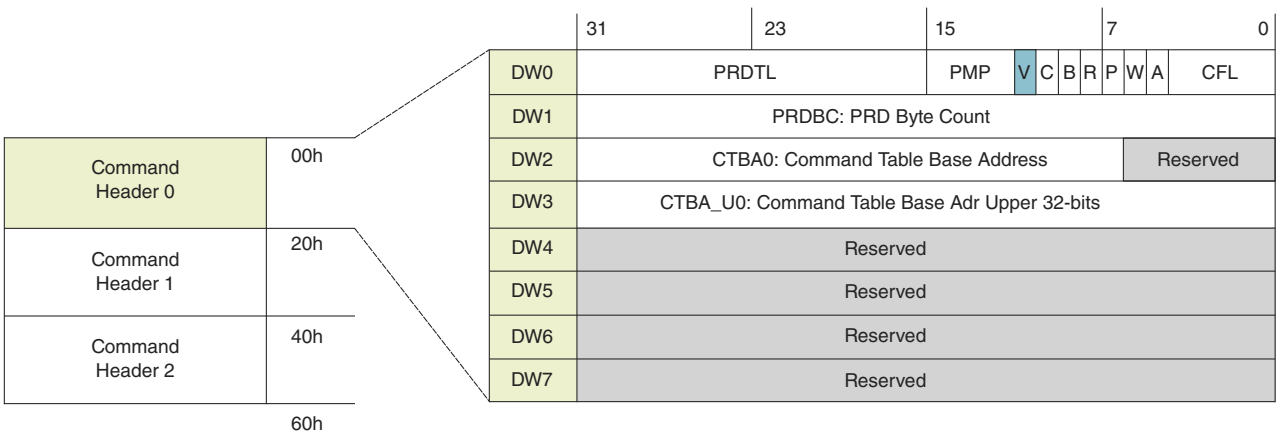


Figure 27-2. Command list structure

This table describes the fields in the command header.

Table 27-9. DW 0 - Description information

Bit	Description
31:12	Per the AHCI specification
11	VBIST Setting this bit indicates that the associated command is to be used as payload for a vendor BIST operation.
10:0	Per the AHCI specification

Table 27-10. Vendor BIST command header

DWord number	Hexadecimal value
DW0	32'h0000_0805
DW1	32'h0000_0000

Table continues on the next page...

Table 27-10. Vendor BIST command header (continued)

DWord number	Hexadecimal value
DW2	System Address (usage dependent)
DW3	System Address (usage dependent)

27.4.2.2 Vendor BIST command descriptor command FIS

The A and B values can be filled in by the user according to the pattern that needs to be tested.

Table 27-11. Command descriptor command

DWord number	Hexadecimal value
DW0	32'h0001_0058
DW1	32'hAAAA_A034
DW2	32'hBBBB_B034
DW3	Reserved - 32’h0000_0000
DW4	Reserved - 32’h0000_0000
DW5	Reserved - 32’h0000_0000
DW6	Reserved - 32’h0000_0000
DW7	Reserved - 32’h0000_0000

27.4.2.3 Receive FIS area reg D2H RFIS structure

The table below provides the receive FIS area reg D2H RFIS structure.

Table 27-12. Receive FIS area reg D2H RFIS structure

DWord number	Hexadecimal Value	Hexadecimal Value
DW0	32'hAAAA_A034	32'hBBBB_B034
DW1	32'hBBBB_B034	32'hAAAA_A034
DW2	32'hAAAA_A034	32'hBBBB_B034
DW3	32'hBBBB_B034	32'hAAAA_A034
DW4	32'hAAAA_A034	32'hBBBB_B034

27.5 PhyControl BIST modes

The BIST modes within the PhyControl layer are designed to cover two main functions.

- Verification of the interface between PhyControl (controller layer) and SerDes
- Generation of limited patterns for compliance testing

The following programming steps are performed when the system software places the PhyControl into one of its BIST modes:

1. Initialize the Phy1Cfg register to place the system into a state of Force Ready and Near End Retimed Loopback, to enable PhyControl BIST Pattern and PhyControl BIST Clear Error, and to select the required BIST pattern .
2. The software brings the device online. This operation causes the controller to de-assert the reset to the PHY and allow the setup start.
3. The software inserts a delay of 10 seconds before proceeding. This is to allow the setup to stabilise and the Rx part of the BIST checker lock to the incoming pattern.
4. Once the delay has elapsed, the software clears the PhyControl BIST Clear Error bit of the register.
5. When these steps are completed, the software monitors the BIST error bit of the Port Vendor Specific registers 0. If this bit remains low the PhyControl passes BIST. If this bit is set then the PhyControl fails BIST. The bit remains set until the software clears it using the PhyControl BIST Clear Error bit of the Phy1Cfg register.

The PhyControl layer supports a number of test patterns. Each pattern is applied in an un-encoded format to the PHY which encodes 8b/10b , serialises, loopbacks, de-serialises and decodes 10b/8b.

- LBP (Lone bit pattern): The lone-bit pattern is designed to be used to generate a pattern only.
- LFTP (Low frequency test pattern): This pattern provides a low frequency, which is allowed within the Serial ATA encoding rules. Pattern 32'hD30.3_D30.3_D30.3_D30.3. The pattern is repetitive.
- MFTP (Mid frequency test pattern): This pattern provides a middle frequency which is allowed within the Serial ATA encoding rules. Pattern encoded 32'hD24.3_D24.3_D24.3_D24.3. The pattern is repetitive.
- HFTP (High frequency test pattern): This pattern provides the maximum frequency allowed within the Serial ATA encoding rules. Pattern encoded 32'hD10.2_D10.2_D10.2_D10.2. The pattern is repetitive.
- PRBS Pattern: This pattern is a long running 32 bit PRBS pattern initially seeded to 32'h1234_5678; The PRBS algorithm is defined as: {PRBSTest[30:0],

(PRBSTest[31] ^ PRBSTest[30] ^ PRBSTest[29] ^ PRBSTest[28] ^ PRBSTest[27] ^ PRBSTest[21])};

- BIST pattern: This pattern provides the counting pattern that allows maximum combinations of characters be tested. It consists of Pattern:
  - 32'hD0.0_ D1.0_D2.0_D3.0
  - 32'hD4.0_ D5.0_D6.0_D7.0
  - 32'hD8.0_ D9.0_D10.0_D11.0
  - 32'hD12.0 D13.0_D14.0_D15.0.
  - 32'hD28.7_ D29.7_D30.7_D31.7

BIST pattern transmission.

1. Seven ALIGN primitives are transmitted to force the Rx path to lock to the transmitted data.
2. A special lock primitive 32'h4A_4A_4A_7C is transmitted to start the pattern match on the Rx side.
3. The selected pattern is transmitted for 256 byte intervals and then the circuit starts from state 1.

27.6 PHY control configuration register OOB timing setup

In SATA systems three out of band (OOB) signals COMRESET, COMINIT, and COMWAKE are used during link initialization. These are achieved by transmission of either a burst of four Gen1 ALIGNp primitives or a burst composed of sixteen D24.3 characters, each burst having a duration of 160 UIOOB (106.67 ns). The individual bursts are followed by idle periods (at common-mode levels), having durations as depicted in the figure and the table below. The duration of the idle period is used to decode the type of OOB signal.

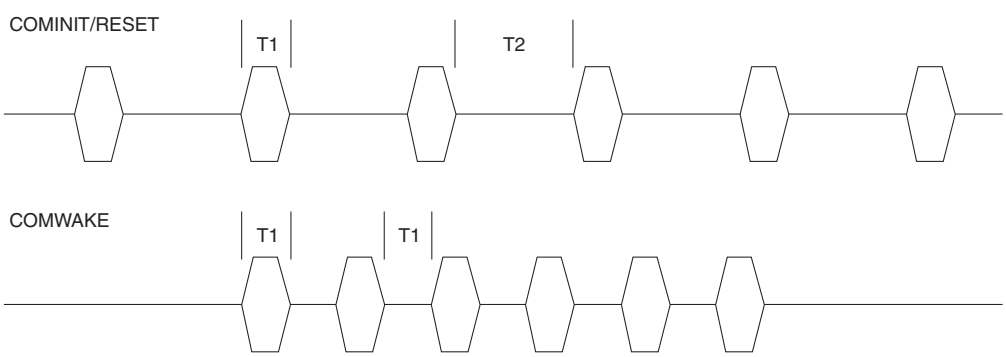


Figure 27-3. OOB signaling timing

Parameter	Units	Limit	Specification
UIOOB, UI During OOB Signaling	Ps	Nominal	666.67
COMINIT/COMRESET and COMWAKE Transmit Burst Length	Ns	Nominal (T1)	106.7
COMINIT/COMRESET Transmit Gap Length	Ns	Nominal (T2)	320.0
COMWAKE Transmit Gap Length	Ns	Nominal (T1)	106.7
COMWAKE Gap Detection Windows	Ns	May Detect Shall detect Shall not detect	35 ≤ T < 175 101.3 ≤ T ≤ 112 T < 35 or T ≥175
COMINIT/COMRESET Gap Detection	Ns	May Detect Shall detect Shall not detect	175 ≤ T < 525 304 ≤ T ≤ 336 T < 175 or T ≥ 525

27.7 Modifications to the AHCI standard PRDT entry

In order to support PRDT control of the AW/RCACHE, the DW2 field of the standard PDRT entry has a new bit designation as follows:

Table 27-13. DW 2 – PRDT Re-designation for AXI operation

Bit	Description
31:00	Reserved
Re-designation	
31:18	Reserved
17	Reserved
16	CACHE bits controlled from PRD (Sub Control)
15:11	Reserved
10:8	Reserved
7:4	This is the value driven onto A[R/W]CACHE when the AXI master is posting a data burst read / write address associated with this PRD entry to the memory controller
3:2	Reserved
1	Inject a CRC error into the associated Received Data FIS
0	Inject a CRC error into the associated Transmitted Data FIS

When a PRDT entry is processed to move data through the AXI master, first the global control field of the AXI Cache Control register bits 12 and 28 are examined to determine the values for the CACHE control signals. If the global control indicates that the values should be controlled from the PRDT then the sub control determines which values control the CACHE signals.

Table 27-14. Global control field

Global	Sub	AR/WCACHE
0	0	AXI Cache Control Register
0	1	AXI Cache Control Register
1	0	AXI Cache Control Register
1	1	DW 2 – PRDT Re-designation for AXI operation

27.8 Transport layer architectural overview

The function of the SATA transport layer is to interface between the command and link layers in the transmission and reception of the Frame Information Structures (FIS).

During transmission, the transport layer frames the FIS placed into the TX FIFO. The FISs are framed based on a programmed length for non-data FIS and/or a configurable length for data FIS. When the transport layer is instructed to send a non-data FIS, it employs a retry policy until the far end signals accepts the transmitted FIS.

On reception, the transport layer de-frames the FISes and places these into the RX FIFO. When an FIS is received, the transport layer informs the command layer.

For a non-data FIS, the FIS is considered received when an EOF is signaled by the link layer and FIS is received with a good CRC.

For a short vendor-specific FIS, the FIS is considered as a non-data FIS.

For longer vendor-specific FIS, the FIS reception is signaled when the RX FIFO reaches its water mark.

For a data FIS, the FIS is considered received when the First DWord (header) is written into the FIFO.

The transport layer is responsible for crossing the clock domain between the transport layer txDWord and the rxDWord clocks and command layer clock domain. The receive FIFO is written to the transport layer receive DWord clock with data contained in the FIS sent by the link layer. Once the data is stable at the output of the receive FIFO on the command layer clock domain, the command layer can take the data. If the command

layer is not ready to accept the data, the data builds up in the receive FIFO. When the receive FIFO exceeds its threshold, the transport layer stalls the link layer, which in turn sends HOLD primitives to the far end to stall it. This threshold takes into consideration the latency involved in getting the far end to stop transmitting the data. This threshold is programmable to allow for the use of high latency repeaters or re-timers in between the host and the device.

The transmit FIFO is written to the command layer clock, with data to be sent in the FIS transferred by the DMA controller. Once the data is stable at the output of the transmit FIFO on the transmit DWord clock domain, the link layer can take the data. If the transmit FIFO cannot supply data to the link layer, the transport layer stalls the link layer, which in turn sends HOLD primitives to the far end to stall it.

27.9 Link layer overview

27.9.1 General operation

The function of the SATA link layer is to interface between the transport and PhyCtrl layers in the transmission and reception of frames and primitives. The link layer utilizes the two unidirectional links provided by the SATA interface to maintain coordinated communication between the host and the device. Payload data can only be transmitted in one direction at a time.

On transmit, the link layer first communicates with the peer far end link layer to determine if it is ready to receive. Assuming the far end link layer can receive data, the local link layer begins to take data in the form of DWords from its transport layer. It inserts Start-of-Frame (SOF) before the start of the data portion of a frame, calculates and inserts the CRC after the data portion of a frame, and inserts the End-of-Frame (EOF) primitive at the end. The link layer scrambles the contents of the frame, including the calculated CRC, but excluding the SOF and EOF delimiters, and any other embedded primitives. The 8B/10B encoding of the data is done either in the PHY control layer or in the PHY layer, depending on the nature of the PHY employed. At the end of the transmission, the link layer reports transmission status to the transport layer.

On receive, the link layer first acknowledges its readiness to receive its peer link layer. Then it awaits reception of the SOF primitive that marks the start of the received data. Following detection of the SOF primitive, the link layer proceeds to accept the incoming data. The 8B/10B decoding of the data is done either in the PHY control layer or in the PHY layer. After this, the link layer removes all primitives including the SOF and EOF delimiters. It then descrambles the contents of the frame. The link layer also calculates the CRC on the incoming frame between the SOF and EOF delimiters, and compares this

calculated value to the received value. Any mismatch is reported to the transport layer. During frame reception, disparity or code errors are reported to the command layer, and appropriate action is taken in the link layer. The descrambled and decoded receive data stream is passed to the transport layer as the frame is being received. Finally at the end of the frame, the link layer reports reception status to the transport layer.

The link layer also partakes in flow control between the local and remote ends. The layer supports flow control actions based on the local FIFO status (which is located in the transport layer), or in response to receiving flow control messages from the remote end.

The transmit side of the link layer is also responsible for inserting a pair of ALIGN primitives every 254 DWords, or more frequently, if programmed by the user.

27.9.2 List of functions

The link layer is composed of a number of functions, as listed below:

- Link layer state machines
- Frame content scrambler and descrambler
- CRC generation and checking
- Bus interfaces to PhyCtrl and transport layer
- CONT primitive processing
- ALIGN insertion on transmit
- Debug functionality
- BIST support

27.9.3 Link layer state machines

Functionally, there are four link layer state machines. These are listed below and described in the subsequent sub-sections:

- Link idle state machine
- Transmit state machine
- Receive state machine
- Power mode change state machine

27.9.3.1 Link idle state machine

The link idle state machine performs the following functions:

- It is responsible for detecting a transmit request from the transport layer, or a frame reception request from the far end, and arbitrating if these two events coincide. The SATA specification defines that the host end always backs down in this case.
- It interprets power mode change requests from both transport and PhyCtrl layers, and initiates actions to enable the power mode change.
- It also detects negation and assertion of PHY_READY from the PhyCtrl layer and notifies the transport layer of the change.

27.9.3.2 Transmit state machine

The transmit state machine manages frame transmission to the PhyCtrl layer. The state machine places the Start-of-Frame (SOF) and End-of-Frame (EOF) headers on each frame, calculates the CRC and inserts it before the EOF delimiter. Between the SOF and CRC markers, the link layer accepts the current DWord from the transport layer and uses this as the next DWord of the frame. The link layer also inserts a pair of ALIGN primitives every 254 DWords of the frame data. Finally, at the end of the frame transmission, the state machine waits for status from the far end link layer through the received R_OK or R_ERR primitives. If the far end receives the frame correctly, the local link layer signals TX_OK to the transport layer; else, it signals TX_NOT_OK to the transport layer.

This state machine also partakes in the flow control actions if necessary, during packet transmission. If the transport layer cannot supply a new DWord and the frame is not finished, the transmit state machine responds by sending HOLD primitives until such time as the transport layer is ready with valid frame data. Also, during frame transmission, if the state machine detects a received HOLD primitive from the PHY Layer, it interrupts the current frame transmission and sends HOLDA primitives to the PHY to be transmitted to the far end.

The current frame transmission can only be aborted by two events.

- On reception of a DMAT primitive from the far end. In this case, the link layer state machine stops the current transfer and calculates and inserts the current CRC. This is a controlled termination.
- When the transport layer wishes to send a control register frame, signaled through TRANSMIT_CRF.

If at any point in the frame transmission process, the link layer detects error conditions, it signals these to the command layer. The errors can occur if:

- PHY_READY negates
- The link layer receives SYNC primitives during frame transmission

27.9.3.3 Receive state machine

The receive state machine is responsible for frame reception from the PHY layer. It removes the Start-of-Frame (SOF) and End-of-Frame (EOF) headers and other primitives from each frame, calculates the CRC and compares it to the received CRC. Between the SOF and CRC markers, the link layer accepts the current DWord from the PhyCtrl layer and uses this as the next DWord of the frame, transferring it to the transport layer. At the end of the frame reception, if the calculated CRC is not the same as the received CRC, the link layer signals an error to the transport layer. This is done through RX_CRC_OK and RX_CRC_NOT_OK statuses. During frame reception, if no errors are detected, the link layer transmits R_IP primitives to the far end peer link layer. Finally, at the end of the frame reception, the link layer sends the R_OK primitive if no error is detected during reception. If an error is detected, it sends an R_ERR primitive instead.

The receive state machine also partakes in flow control actions if necessary, during FIS reception. If the transport layer cannot accept a new DWord, (because its receive FIFO has reached its watermark level), and the FIS is not finished, the receive state machine responds by sending HOLD primitives on the back channel until such time as the transport layer is ready to accept FIS data again. Also, during FIS reception, if the state machine detects a received HOLD primitive from the far end, it responds by sending HOLDA primitives to the far end.

The current frame reception can be interrupted if the transport layer wishes to send a control register frame, signaled through TRANSMIT_CRF.

If at any point in the frame reception process, the link layer detects error conditions, it signals these to the command layer. The errors can occur if

- PHY_READY negates
- The link layer receives SYNC primitives
- The link layer receives WTRM primitives before EOF

27.9.3.4 Power mode change state machine

This state machine handles change of power mode requests. These requests can come from the command layer Super Set registers or the far end. The command layer signals a change request by asserting CFG_LINK_GO_PARTIAL or CFG_LINK_GO_SLUMBER. This state machine responds by transmitting PMREQ_P/ PMREQ_S primitives to the far end and waiting for PMACK primitives from it in

response. Once PMACK is received, the state machine instructs the PHY layer to enter either partial or slumber state, by driving PHY_GO_PARTIAL or PHY_GO_SLUMBER signals. These signals must remain active for the duration of the new power mode.

A write of '1' to the CFG_LINK_WAKEUP bit or reception of a COMWAKE from the far end will initiate a resume to active power mode.

Note: CFG_LINK_WAKEUP should be set back to '0' once PhyReady is asserted again.

If the link layer receives a PMREQ_P/PMREQ_S primitive from a peer link layer, and is enabled to perform power management modes (CFG_EN_PARTIAL/SLUMBER = '1'), it responds by sending at least four PMACK primitives, and then asserts PHY_GO_PARTIAL or PHY_GO_SLUMBER to the PhyCtrl block. A write of '1' to the CFG_LINK_WAKEUP bit or reception of a COMWAKE from the far end initiates a resume to the active power mode.

Note: CFG_LINK_WAKEUP should be set back to '0' once PhyReady is asserted again.

If the link layer receives an XRDY primitive from the far-end while it is in the partial or slumber state, it returns to idle and signals a link sequence error to the command layer, that is LINK_SEQ_ERROR = 1. Automatic partial to slumber transitions can be set by programming the CAP2 register (offset 24h)

27.9.4 Frame content scrambler and descrambler

Two separate scramblers are used in serial ATA - one for the data payload and another for repeated primitive suppression.

The contents of each DWord of data (excluding all primitives) between SOF and EOF must be scrambled before 8B/10B encoding. Scrambling is performed on DWord quantities according to the following polynomial:

G(X) = X16 + X15 + X13 + X4 + 1

The scrambler is initialized with a seed value of 16'hFFFF at each SOF transmission and rolls over every 2048 DWords. Payload data is scrambled prior to transmission, by XORing the data to be transmitted with the output of this scrambler.

If a CONT primitive is transmitted, the intervening data between the last CONT primitive and a subsequent primitive must be scrambled as well. This scrambler uses the same polynomial as defined above for data payload scrambling and is reset to the initial value upon detection of a COMINIT or COMRESET event (. PHY_GOT_CIOR = '1'). If a CONT primitive is transmitted or received during a frame transfer, the current data payload scrambler value at the last DWord is held.

When payload data is received by the link layer it is de-scrambled by XORing it with the output of its descrambler. The descrambler is re-seeded at the beginning of the received data payload, that is at each SOF reception. The descrambler uses the same polynomial as the scrambler.

27.9.5 CRC generator and checker

A 32 bit CRC is calculated on the data contents of each frame, and is inserted in the DWord before the EOF. The CRC covers all data bytes in the frame excluding any primitives, such as SOF, EOF, HOLD, HOLDA, DMAT, SYNC, X_RDY, R_RDY or ALIGNs.

The CRC generator works on DWord quantities. Any padding to the boundary is done in the transport layer. The polynomial used for the CRC is as follows:

G(X) = X32 + X26 + X23 + X22 + X16 + X12 + X11 + X10 + X8 + X7 + X5 + X4 + X2 + X + 1

The CRC is initialized with a seed value of 32'h52325032 at each SOF.

The CRC generation or checking does not apply to primitives (as stated above) or to CONT'ed primitives. If a CONT primitive is transmitted or received, the intervening data between the last CONT primitive and a subsequent primitive is not included in the CRC calculation for a frame. If this happens during a frame transfer, the current CRC scrambler value at the last DWord is held.

27.9.6 8B/10B encode and decode

All data and primitives must be encoded prior to transmission on the line. The standard implementation has the 8B/10B encoder and decoder located in the PHY control layer.

27.9.7 CONT primitive processing

The link layer is capable of replacing repetitive primitive streams with scrambled data, by use of the CONT primitive. This reduces EMI emissions, as primitives are not scrambled.

The link layer can transmit a CONT primitive at a point where it knows it must transmit a number of repeating primitives. After a CONT primitive has been transmitted, the link layer transmits scrambled junk data to the PHY layer. The content of this junk data is disregarded. At the far end link layer, the reception of a CONT primitive implies that the

last received valid primitive will be repeated until it receives the next valid non-align primitive. Transmission of a new valid primitive halts the current CONT processing; reception of a new valid non-align primitive halts the current CONT processing.

This action can occur on transmit and receive. The link layer supports both transmission and reception of CONT primitives. The scrambler and the CRC generator/checker actions are changed during the transmission and reception of CONT sequences as described in Frame content scrambler and descrambler and CRC generator and checker.

Transmission of CONT primitives are not allowed until the link layer state machine has initialized correctly, and has transmitted 16 non-align primitives.

For diagnostic purposes CONT insertion can be suppressed in the transmit path; the receive path can always receive those.

27.9.8 ALIGN insertion

The link layer is responsible for ALIGN insertion and removal at a fixed frequency. A pair of ALIGN primitives are inserted into the transmit data stream every 254 DWords. These primitives are used in the PHY layer to maintain character alignment and to help in elastic buffering, if required.

At the receive end, the ALIGN primitives are stripped from the incoming data stream in the link layer.

For diagnostic purposes, the rate of ALIGNs can be increased as much as two ALIGNS per one DWord, that is: ALIGN, ALIGN, data, ALIGN, ALIGN.

Also, the CFG_SEND_4_ALIGNS bit can be set to instruct the link layer to send four aligns at a time, instead of two.

27.9.9 Debug functionality

There are a number of useful features designed into the ink layer to aid debug. These include:

- A configuration bit in the command layer (CFG_TX_BADCRC) can be set to force the link layer to transmit a bad CRC at the end of a frame. One bad CRC is transmitted every time the bit is configured. To force transmission of a new bad CRC, the configuration bit needs to be cleared and written to again.
- A configuration bit in the command layer (CFG_RX_BADCRC) can be set to force the link layer to detect a bad CRC value at the end of a frame. Only one bad CRC is

be detected for one setting. The configuration bit must be cleared and written to again to force detection of another bad CRC.
- A configuration bit in the command layer (CFG_TXPRIM_JUNK) can be set to force the junk data that is transmitted after a CONT primitive to be equal to 32'hDEADBEEF.
- The align insertion rate can be increased using the CFG_ALIGN_RATE register field in the command layer.
- Four error counters can be monitored by issuing register reads to the command layer. These error counters include: Disparity error counter, Code error counter, PHY Internal error counter and Control Character error counter (see signal descriptions for detail).
- A number of configuration bits in the command layer can be used to override normal primitive insertion. For example,
  - Set CFG_PRIM_OVR_STATE = the L_SendHold state (16)
  - Set CFG_PRIM = 32'hb5b5957c , that is a SYNC primitive
  - During the transfer, set CFG_PRIM_OVR_EN = 1. When the link layer detects a rising edge on CFG_PRIM_OVR_EN, it inserts one SYNC primitive into the datastream in place of the HOLD, when the LINK_STATE reaches the L_SendHold state. Only one HOLD primitive is overridden - the CFG_PRIM_OVR_EN must be cleared and written to again to force another override to occur.

27.9.10 BIST support

The transmit and receive sub-blocks of the link layer contain logic to support BIST activate FIS functionality.

When a BIST activate FIS is either received or transmitted successfully by the transport layer, it issues a request to the link layer to enter BIST mode (LINK_BIST_REQ = 1). This forces the link layer to enter a BIST state in its state machine as soon as it receives a SYNC primitive from the far-end. When the BIST state is entered, an acknowledge signal is sent to the transport layer, that is LINK_BIST_ACK = 1. In the BIST state, the link layer transmits a data sequence as specified by the two BIST data patterns in the BIST activate FIS. The link layer also monitors the incoming data from the PhyCtrl block to detect if the BIST data pattern is the same as specified in the BIST activate FIS. When it detects the correct data sequence, the BIST_ERR output is de-asserted. This signal stays de-asserted unless an error occurs in the datastream from the far-end. The BIST_ERR output is sent to the command layer from where it can be read by the host software. The link layer also outputs a signal called BIST_ACTIVE to the command layer. This singnal is asserted when the link layer enters its BIST state and de-asserted when BIST mode is exited due to a COMRESET being received.

27.10 PHY control layer overview

The PHY control layer operates between the SerDes and link layers.

The main functions of the PHY control layer are

- Data path operation
  - Rx data path
  - Tx data path
- PHY Initialization state machine
  - Out of band processing
  - Speed negotiation

On receive, the PHY Control layer converts the encoded 20-bit or decoded 16-bit parallel data from the SerDes to a 32-bit DWord, which it presents to the link layer. The PHY control layer aligns the control word of the SATA primitive to the lowest word position of the DWord. The two main configurations of the PHY control layer operate on either encoded data or decoded data, as sourced from the SerDes. In the case of encoded data, the PHY control contains an 8b/10b decoder function and decodes the incoming data into data, control/data and code or disparity error. In the case of decoded data, the PHY control sources data, control/data and code or disparity error from the SerDes.

The receive DWord clock is generated from the receive core clock outputted by the SerDes. This is generated as part of the DWord alignment process.

On transmit, the PHY control layer takes in the 32-bit transmit data from the link layer and converts the data into encoded 20-bit or un-encoded 16-bit parallel data from the SerDes. The control/data bit from the link layer (which is always assumed to be associated with the lowest byte position of the transmit DWord) is also passed onto the SerDes with the appropriate word. The PHY control layer takes the transmit word clock outputted by the SerDes and converts it to a DWord transmit clock which it sends to the link layer.

The PHY control layer implements the PHY initialization state machine. The state machine implements the device PHY initialization state machine as defined in the Serial ATA Revision 3.1 June 2, 2009.

27.11 Reset

The SATA AHCI controller supports three levels of reset:

- Software reset - a single device on one of the ports is reset, but the HBA and physical communication remain intact. This is the least intrusive.
- Port reset – the physical communication between the HBA and device on a port are disabled. This is more intrusive.
- HBA reset – the entire HBA is reset, and all ports are disabled. This is the most intrusive.

When an HBA or port reset occurs, PHY communication is re-established with the device through a COMRESET followed by the normal out-of-band communication sequence defined in SATA. At the end of the reset, the device, if working properly, will send a D2H Register FIS, which contains the device signature. When the HBA receives this FIS, it updates the PxTFD[STS] and PxTFD[ERR] bits, and updates the PxSIG register with the signature.

Software is recommended to follow a staggered reset approach, starting from a less intrusive reset mechanism and then using a more intrusive reset mechanism only if the less intrusive mechanism does not succeed in clearing the condition.

27.11.1 Software reset

Legacy software contained a standard mechanism for generating a reset to a SATA device – setting the SRST (software reset) bit in the Device Control register. SATA has a more robust mechanism called COMRESET, also referred to as port reset. A port reset is the preferred mechanism for error recovery and should be used in place of software reset.

To issue a software reset in AHCI, 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 set 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 HBA to perform a SYNC escape if 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 the SRST bit set to '0' in the Control field of the Register FIS, the 'C' bit is set 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'.

Refer to the Serial ATA Revision 2.6 specification for more information on the software reset FIS sequence.

Table 27-15. Register FIS host to device SRST On

	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	
0	Features								Command								C	R	R	R	PM Port				FIS Type (27h)								
	0	0	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	1	1	32'h 00000027
1	Device								LBA High								LBA Mid								LBA Low								
	0	0	0	0	0	0	0	0	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'h 00000000
2	Features (exp)								LBA High (exp)								LBA Mid (exp)								LBA Low (exp) (0)								
	0	0	0	0	0	0	0	0	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'h 00000000
3	Control								Reserved (0)								Sector Count (exp)								Sector Count								
	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	32'h 04000000
4	Reserved (0)								Reserved (0)								Reserved (0)								Reserved (0)								
	0	0	0	0	0	0	0	0	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'h 00000000

Table 27-16. Register FIS host to device SRST Off

	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		
0	Features								Command								C	R	R	R	PM Port				FIS Type (27h)									
	0	0	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	1	1	32'h	00000027
1	Device								LBA High								LBA Mid								LBA Low									
	0	0	0	0	0	0	0	0	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'h	00000000
2	Features (exp)								LBA High (exp)								LBA Mid (exp)								LBA Low (exp) (0)									
	0	0	0	0	0	0	0	0	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'h	00000000
3	Control								Reserved (0)								Sector Count (exp)								Sector Count									
	0	0	0	0	0	0	0	0	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'h	00000000
4	Reserved (0)								Reserved (0)								Reserved (0)								Reserved (0)									
	0	0	0	0	0	0	0	0	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'h	00000000

When issuing a software reset sequence, there should not be other commands in the command list. Before issuing the software reset, software must clear PxCMD[ST], wait for the port to be idle (PxCMD[CR] = '0'), and then re-set PxCMD[ST].

PxTFD[STS[BSY]] and PxTFD[STS[DRQ]] must be cleared prior to issuing the reset. If PxTFD[STS[BSY]] or PxTFD[STS[DRQ]] is still set based on the failed command, then a port reset should be attempted or command list override (PxCMD[CLO]) should be used if supported. Note that a device-to-host FIS from a previously failed command may be received after the PxCMD[ST] bit has been cleared and/or re-set. It is recommended that the HBA ignore such a FIS or SYNC Escape that FIS (as directed by the CH[R] bit being set in the first software reset H2D Register FIS).

27.11.2 Port reset

If a port is not functioning properly after a software reset, software may attempt to re-initialize communication with the port through a COMRESET. It must first clear PxCMD[ST] and wait for PxCMD[CR] to clear to '0' before re-initializing communication. However, if PxCMD[CR] does not clear within a reasonable time (500 milliseconds), it may assume the interface is in a hung condition and may continue with issuing the port reset.

Software causes a port reset (COMRESET) by writing 4'h1 to the PxSCTL[DET] bit to invoke a COMRESET on the interface and start a re-establishment of PHY layer communications. Software shall wait at least 1 millisecond before clearing PxSCTL[DET] to 4'h0; this ensures that at least one COMRESET signal is sent over the interface. After clearing PxSCTL[DET] to 4'h0, software should wait for Serial ATA AHCI 1.3 Specification 101 communication to be re-established as indicated by bit 0 of PxSSTS[DET] being set to 4'h1. Then software should write all 1s to the PxSERR register to clear any bits that were set as part of the port reset.

When PxSCTL[DET] is set to 4'h1, the HBA shall reset PxTFD[STS] to 8'h7F and shall reset PxSSTS[DET] to 4'h0. When PxSCTL[DET] is set to 4'h0, upon receiving a COMINIT from the attached device, PxTFD[STS[BSY]] shall be set to 1'h1 by the HBA.

27.11.3 HBA reset

If the HBA becomes unusable for multiple ports, and a software reset or port reset does not correct the problem, software may reset the entire HBA by setting GHC[HR] to '1'. When software sets the GHC[HR] bit to '1', the HBA shall perform an internal reset action. The bit shall be cleared to '0' by the HBA when the reset is complete. A software write of '0' to GHC[HR] shall have no effect. To perform the HBA reset, software sets GHC[HR] to '1' and may poll until this bit is read to be '0', at which point software knows that the HBA reset has completed.

If the HBA has not cleared GHC[HR] to '0' within 1 second of software setting GHC[HR] to '1', the HBA is in a hung or locked state.

When GHC[HR] is set to '1', GHC[AE], GHC[IE], the IS register, and all port register fields (except PxFB/PxFBU/PxCLB/PxCLBU) that are not HwInit in the HBA's register memory space are reset. The HBA's configuration space and all other global registers/ bits are not affected by setting GHC[HR] to '1'. Any HwInit bits in the port specific registers are not affected by setting GHC[HR] to '1'. The port specific registers PxFB, PxFBU, PxCLB, and PxCLBU are not affected by setting GHC[HR] to '1'. If the HBA supports staggered spin-up, the PxCMD[SUD] bit will be reset to '0'; software is responsible for setting the PxCMD[SUD] and PxSCTL[DET] fields appropriately such that communication can be established on the SATA link. If the HBA does not support staggered spin-up, the HBA reset shall cause a COMRESET to be sent on the port.

Chapter 28
SerDes Module

28.1 The SerDes module as implemented on the chip

This section provides details about how the SerDes module is implemented on the chip.

28.1.1 SerDes lane assignments and multiplexing

The chip supports one SerDes with three lanes.

28.1.1.1 Configuration of networking SerDes

The correct configuration involves selecting the correct values for the SerDes RCW fields (see RCW Field Definitions).

- SerDes configuration:
  - Protocol(s): Selected using RCW[SRDS_PRTCL_S1]
  - PLLs: Enabled using RCW[SRDS_PLL_PD_S1]
  - PLL reference clock: RCW[SRDS_PLL_REF_CLK_SEL_S1]
  - SerDes internal reference clock: RCW[SRDS_INT_REFCLK]
  - SerDes PLL2 reference clock: RCW[SRDS_REFCLK_SEL]
  - Rates are determined by the protocols selected using the RCW bit fields.

28.1.1.2 SerDes protocols

The following table shows the SerDes lanes corresponding to the SD1_TXn_P/N and SD1_RXn_P/N signals and the lane assignments for the controllers that can use the SerDes.

The following notation conventions are used in the figure:

- SGMII notation for PFE:
  - sg.mn means SGMII (1 lane @ 1.25 GBaud or 3.125 GBaud)
  - "m" indicates that MAC is from PFE.
  - "n" indicates which MAC on the PFE.
  - For example, "sg.m2," indicates SGMII for MAC 2 on PFE.
- PCI Express :
  - PEX*n* (5/2.5) means PCI Express operating up to 5 or 2.5 GT/s depending on maximum rate selection and training.
- SATA
  - SATA*n* (6/3/1.5) means SATA operating at 6 or 3 or 1.5 Gbps depending on rate selection. The rate of selection is performed by PxSCTL[SPD] register as described in SATA 3.0.

Table 28-1. Lane assignments for SerDes

SerDes Lane	A	B	C	D
Signal	SD1_RX0_P/	SD1_RX1_P/	Unused	SD1_RX2_P/
	SD1_RX0_N/	SD1_RX1_N/		SD1_RX2_N/
	SD1_TX0_P/	SD1_TX1_P/		SD1_TX2_P/
	SD1_TX0_N	SD1_TX1_N		SD1_TX2_N
Controller Lane Assignments	-	PCI Express (x1)		PCI Express (x1)
	TX_CLK	TX_CLK		-
	-	-		SATA 1
	sg.m1 (1G/2.5G)	sg.m2 (1G/2.5G)		-

NOTE

The names of lanes A, B, C, and D are interchangeably used with 0, 1, 2, and 3 respectively.

The following table shows the supported networking protocol options for the SerDes module. The SerDes protocols are configured through software and not at power-on-reset.

The SerDes lane A and lane B for SGMII are driven by MAC1 and MAC2, respectively. However, the RGMII interface is driven by Ethernet MAC2 of PFE. If SerDes is configured for two SGMII interfaces, the RGMII interface is not used. If only the SerDes lane A is used for SGMII, RGMII is available at MAC2.

The two lanes of SGMII are completely independent. The MAC connected to the SerDes lane B is the same MAC connected to the RGMII interface. The only exception to the SerDes lane being selected independently is that PCI Express is only supported on one SerDes lane and cannot be enabled at two lanes simultaneously.

Table 28-2. Supported SerDes options

SRDS_PRTCL_S1 RCW[128:143] (in hex)	A	B	C	D	RGMII	Per lane PLL mapping
0000	Unused				MAC2	-
0005	Unused		Unused	PCI Express (x1)	MAC2	2222
2208	sg.m1 (2.5G)	sg.m2 (2.5G)		SATA	-	1122
0008	Unused			SATA	MAC2	1122
3508	sg.m1 (1G)	PCI Express (x1)		SATA	MAC2	1122
3305	sg.m1 (1G)	sg.m2 (1G)		PCI Express (x1)	-	2222
2205	sg.m1 (2.5G)	sg.m2 (2.5G)		PCI Express (x1)	-	1122
2305	sg.m1 (2.5G)	sg.m2 (1G)		PCI Express (x1)	-	1222
9508	TX_CLK	PCI Express (x1)		SATA	MAC2	1112
3905	sg.m1 (1G)	TX_CLK		PCI Express (x1)	MAC2	1112
9305	TX_CLK	sg.m2 (1G)		PCI Express (x1)	-	1112

NOTE

- SerDes PLL1 and PLL2 use same clock source chosen by RCW[SerDes_INT_REFCLK]. Input clock to SerDes PLLs should meet all the requirements of all the protocols selected by RCW[SRDS_PRTCL_S1].
- When PLL1 is not required, it can be powered down if the reference clock source is internal. If the reference clock source is external, the PLL1 cannot be powered down.
- TX_CLK is an output signal. It provides a 100 MHz differential reference clock that can be used by the PCI Express Endpoint.

28.1.1.2.1 Disabling unused SerDes modules

In order to disable the SerDes1 module, the following RCW bits should be configured:

- SRDS_PLL_PD_S1 = 2'b11 (both PLLs configured as powered down)
- SRDS_PLL_REF_CLK_SEL_S1 = 2'b00
- SRDS_PRTCL_S1=2 (no other values are permitted when both PLLs are powered down)

28.1.1.3 Powering down unused SerDes lanes

The power down of SerDes data lanes can be achieved by software using registers in the SerDes control block. Refer RCW[SRDS_PLL_PD_S1] bit definition to see how lanes can be powered down based on the powering down of individual PLLs.

28.1.2 SerDes clocking

28.1.2.1 Valid reference clocks and PLL configurations for SerDes protocols

Each supported SerDes protocol allows for a finite set of valid SerDes-related RCW fields and reference clock frequencies, as shown in the following table:

Table 28-3. Valid SerDes RCW encodings and reference clocks

SerDes protocol (given lane)	Valid reference clock frequency	Legal setting for SRDS_PRTCL_Sn	Legal setting for SRDS_PLL_REF_CLK_SEL_S1		Reference clock
			PLL1	PLL2	
TXCLK (100 MHz differential clock)	100 MHz	TXCLK	0: 100 MHz	0: 100 MHz	External only
	125 MHz		1: 125 MHz	1: 125 MHz	External or internal
PCI Express 2.5 GT/s (doesn't negotiate upwards)	100 MHz	Any PCI Express	0: 100 MHz	0: 100 MHz	External only
	125 MHz		1: 125 MHz	1: 125 MHz	External or internal
PCI Express 5 GT/s (can negotiate up to 5 GT/s)	100 MHz	Any PCI Express	0: 100 MHz	0: 100 MHz	External only
	125 MHz		1: 125 MHz	1: 125 MHz	External or internal
SATA (1.5, 3, 6 Gbps)	100 MHz	Any SATA	-	0: 100 MHz	External only
	125 MHz		-	1: 125 MHz	External or internal
SGMII (1.25 GBaud)	100 MHz	SGMII @ 1.25 GBaud	0: 100 MHz	0: 100 MHz	External only
	125 MHz		1: 125 MHz	1: 125 MHz	External or internal
2.5G SGMII (3.125 GBaud)	125 MHz	SGMII @ 3.125 GBaud	0: 125 MHz	-	External or internal

Notes:

1. A spread-spectrum reference clock is permitted for PCI Express, provided that common SerDes reference clock must be used for both link partners. Whenever the spread-spectrum clocking is used for the SerDes reference clock of a single PCI Express controller, the same SerDes PLL must not be used concurrently for any other protocols.
2. SerDes lanes configured as SATA initially operate at 3 Gbps. The 1.5 Gbps operation may later be enabled through the SATA module itself. It is possible for software to set SATA at different rates.

28.2 Overview

The SerDes module implements link serialization/deserialization and PCS functions for high speed serial interfaces from 1.25 Gbaud to 6 Gbaud.

28.2.1 Features

The SerDes module includes 3 data lanes and 2 PLLs and supports the following protocols:

- PCI Express 3.0 (November, 2010) @ 2.5, 5 Gbaud
- SGMII ENG-46158, Revision 1.9 (July, 2009) @ 1.25, 3.125 Gbaud
- IEEE 802.3-2008 1000Base-KX @ 1.25 Gbaud
- Serial ATA Revision 3.0 (June, 2009) @ 1.5, 3.0, 6.0 Gbaud

28.3 Modes of Operation

The SerDes supports several combinations of protocols and frequencies. See SerDes Lane Assignments and Multiplexing, for details on the combinations.

28.4 External Signals Description

The table below lists the external signals for the SerDes.

Table 28-4. SerDes Interface Signals

Pin Name	Description	No. of Signals	I/O
SDn_TX[0:2]_P	Transmitter serial output, positive data	3	O
SDn_TX_B[0:2]_N	Transmitter serial output, negative data	3	O
SDn_IMP_CAL_TX	Tx Impedance Calibration	1	I
SDn_RX[0:2]_P	Receiver serial output, positive data	3	I
SDn_RX_B[0:2]_N	Receiver serial output, negative data	3	I
SDn_IMP_CAL_RX	Rx Impedance Calibration	1	I
x=1, 2			
SDn_REFx_CLK_P	Reference clock input to PLLx	2	I

Table continues on the next page...

Table 28-4. SerDes Interface Signals (continued)

Pin Name	Description	No. of Signals	I/O
SDn_REFx_CLK_N	Reference clock-bar input to PLLx	2	I

28.5 SerDes register descriptions

The SerDes module is programmed by control/status registers (CSRs). The CSRs are used for mode control, and to extract status information. All accesses to and from the registers must be made as 32-bit accesses. There is no support for accesses of sizes other than 32 bits. Writes to reserved register bits must always preserved the previous value; that is, the register should be programmed by reading the value, modifying appropriate fields, and writing back the value. Unless otherwise specified, the read value of unmapped registers or of reserved bits in mapped registers is not defined, and must not be assumed to be 0.

The table below lists the address, name, and a cross-reference to the complete description of each register. The offsets to the memory map table are defined for each SerDes module from 0x0000 to 0x1FFF (8 KB).

Unlisted register addresses are reserved.

- Reserved fields are always ignored for the purposes of determining access type.
- Reserved registers and fields must be preserved on writes.
- R (read only) indicates that all the non-reserved fields in a register are read-only.
- R/W (read/write) indicates all of the non-reserved fields in a register are either read/write or read-only, with at least one read/write field. The register description indicates which fields are read-write and which read-only.
- RC (read clear) indicates all of the fields in a register are cleared on read, are are not otherwise writeable.
- W1C indicates that all of the non-reserved fields in a register are cleared by writing ones to them.
- Mixed indicates a combination of access types.
- Special is used when no other category applies. In this case the register figure and field description provide the details for access.

This section describes the control, status, and test registers for SerDes and Protocol PCS layers.

All reserved register fields must be preserved on register writes (read-modified-write).

28.5.1 SerDes Memory map

SerDes base address: 1EA_0000h

Offset	Register	Width (In bits)	Access	Reset value
0h	SerDes PLL1 Reset Control Register (PLL1RSTCTL)	32	RW	0000_0020h
4h	SerDes PLL1 Control Register 0 (PLL1CR0)	32	RW	8080_0008h
8h	SerDes PLL1 Control Register 1 (PLL1CR1)	32	RW	0800_0000h
18h	SerDes PLL1 Control Register 5 (PLL1CR5)	32	RW	0000_0000h
20h	SerDes PLL2 Reset Control Register (PLL2RSTCTL)	32	RW	0000_0020h
24h	SerDes PLL2 Control Register 0 (PLL2CR0)	32	RW	8080_0008h
28h	SerDes PLL2 Control Register 1 (PLL2CR1)	32	RW	0800_0000h
38h	SerDes PLL2 Control Register 5 (PLL2CR5)	32	RW	0000_0000h
90h	SerDes Transmit Calibration Control Register (TCALCR)	32	RW	0000_0000h
94h	SerDes Transmit Calibration Control Register 1 (TCALCR1)	32	RW	0000_0000h
A0h	Receive Calibration Control Register (RCALCR)	32	RW	0000_0000h
A4h	SerDes Receive Calibration Control Register 1 (RCALCR1)	32	RW	0000_0000h
B0h	General Control Register 0 (GR0)	32	RW	0000_0000h
200h	Protocol Configuration Register 0 (PCCR0)	32	RW	0000_0000h
208h	Protocol Configuration Register 2 (PCCR2)	32	RW	0000_0000h
220h	Protocol Configuration Register 8 (PCCR8)	32	RW	0000_0000h
800h	General Control Register 0 - Lane A (LNAGCR0)	32	RW	1104_0000h
804h	General Control Register 1 - Lane A (LNAGCR1)	32	RW	004C_4011h
80Ch	Speed Switch Control Register 0 - Lane A (LNASSCR0)	32	RW	9800_2B00h
810h	Receive Equalization Control Register 0 - Lane A (LNARECR0)	32	RW	0000_001Fh
814h	Receive Equalization Control Register 1- Lane A (LNARECR1)	32	RO	0000_0008h
818h	Transmit Equalization Control Register 0 - Lane A (LNATECR0)	32	RW	1028_3000h
81Ch	Speed Switch Control Register 1- Lane 0 (LNASSCR1)	32	RW	0000_0000h
820h	TTL Control Register 0 - Lane A (LNATTLCR0)	32	RW	0000_0400h
83Ch	Test Control/Status Register 3 - Lane A (LNATCSR3)	32	RW	0400_0000h
840h	General Control Register 0 - Lane B (LNBGCR0)	32	RW	1104_0000h
844h	General Control Register 1 - Lane B (LNBGCR1)	32	RW	004C_4011h
84Ch	Speed Switch Control Register 0 - Lane B (LNBSSCR0)	32	RW	9800_2B00h
850h	Receive Equalization Control Register 0 - Lane B (LNBRECR0)	32	RW	0000_001Fh
854h	Receive Equalization Control Register 1- Lane B (LNBRECR1)	32	RO	0000_0008h
858h	Transmit Equalization Control Register 0 - Lane B (LNBTECR0)	32	RW	1028_3000h
85Ch	Speed Switch Control Register 1- Lane 0 (LNBSSCR1)	32	RW	0000_0000h
860h	TTL Control Register 0 - Lane B (LNBTTLCR0)	32	RW	0000_0400h

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
87Ch	Test Control/Status Register 3 - Lane B (LNBTCSR3)	32	RW	0400_0000h
8C0h	General Control Register 0 - Lane D (LNDGCR0)	32	RW	1104_0000h
8C4h	General Control Register 1 - Lane D (LNDGCR1)	32	RW	004C_4011h
8CCh	Speed Switch Control Register 0 - Lane D (LNDSSCR0)	32	RW	9800_2B00h
8D0h	Receive Equalization Control Register 0 - Lane D (LNDRECR0)	32	RW	0000_001Fh
8D4h	Receive Equalization Control Register 1- Lane D (LNDRECR1)	32	RO	0000_0008h
8D8h	Transmit Equalization Control Register 0 - Lane D (LNDTECR0)	32	RW	1028_3000h
8DCh	Speed Switch Control Register 1- Lane 0 (LNDSSCR1)	32	RW	0000_0000h
8E0h	TTL Control Register 0 - Lane D (LNDTTLCR0)	32	RW	0000_0400h
8FCh	Test Control/Status Register 3 - Lane D (LNDTCSR3)	32	RW	0400_0000h
1000h	PEXA Protocol Control Register 0 (PEXACR0)	32	RW	2000_0000h
1804h	SGMIIA Protocol Control Register 1 (SGMIIACR1)	32	RW	0000_0000h
180Ch	SGMIIA Protocol Control Register 3 (SGMIIACR3)	32	RO	0000_0000h
1814h	SGMIIB Protocol Control Register 1 (SGMIIBCR1)	32	RW	0000_0000h
181Ch	SGMIIB Protocol Control Register 3 (SGMIIBCR3)	32	RO	0000_0000h

28.5.2 SerDes PLLa Reset Control Register (PLL1RSTCTL - PLL2 RSTCTL)

28.5.2.1 Offset

For a = 1 to 2:

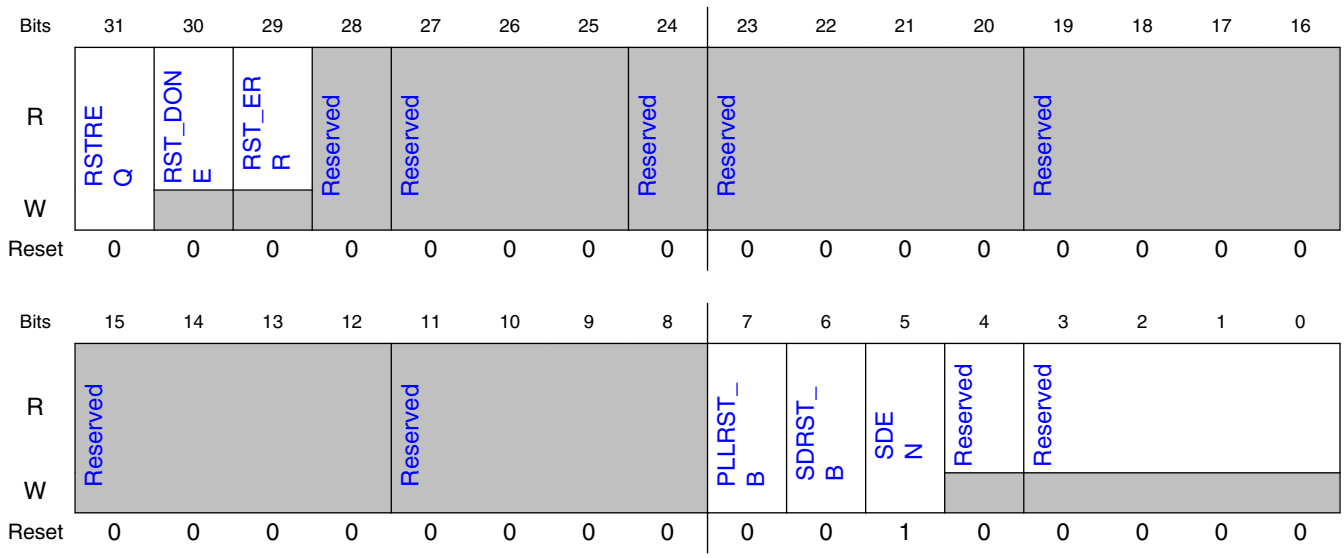
Register	Offset
PLLaRSTCTL	-20h + (a × 20h)

28.5.2.2 Function

PLL*n*RSTCTL contains control and status bits for the SerDes PLL *n* reset state machine.

This register should not be modified unless (RST_ERR=1), 1000 (RST_DONE=1), or 1111 (waiting for RSTREQ).

28.5.2.3 Diagram



28.5.2.4 Fields

Field	Function
31 RSTREQ	Reset request PLL Reset Request - write 1 self-clearing. Software setting of 1 initiates a soft reset of SerDes PLL *n*, along with all lanes using PLL *n*. Hardware automatically clears this bit before reset is done. Note: This field must be 0 if PLL2 does not have a reference clock. Clearing this bit during the reset sequence has no effect. 0b - Not requesting PLL soft reset. 1b - Requesting PLL soft reset.
30 RST_DONE	Reset done PLL Reset Done from Control Block State Machine 0b - PLL reset sequence in progress. 1b - PLL reset sequence complete.
29 RST_ERR	Reset Error No PLL Lock before counter time_out 0b - Normal function 1b - PLL lock didn't happen in the expected time period
28 —	Reserved
27-25 —	Reserved

Table continues on the next page...

Field	Function
24 —	Reserved
23-20 —	Reserved
19-16 —	Reserved
15-12 —	Reserved
11-8 —	Reserved
7 PLLRST_B	PLL reset. Resets PLLn PLL reset. Resets PLLn . Default value: 0 The SerDes reset state machine overrides this field to 0 in some states. Setting PLLRST_B=0 when RST_DONE=0 forces a SerDes PLL reset regardless of the reset state machine state, and may cause the reset state machine to time out and set RST_ERR=1. 0b - Force PLL reset 1b - Application Mode, unless RSTREQ=1
6 SDRST_B	SRDS group reset. Resets all lanes operating from PLLn The SerDes reset state machine overrides this field to 0 in some states. 0b - Force SerDes group reset 1b - Application Mode, unless RSTREQ=1
5 SDEN	SerDes enable. Enable the section of the SerDes module clocked by PLLn SerDes enable. Enable the section of the SerDes module clocked by PLLn. Default value: 1 The SerDes reset state machine overrides this field to 0 in some states. Setting SDEN=0 when RST_DONE=0 forces a SerDes powerdown regardless of the reset state machine state, and may cause the reset state machine to time out and set RST_ERR=1. Note: This field must be 0 if PLL2 does not have a reference clock. 0b - Force SerDes power down 1b - Application Mode, unless RSTREQ=1
4 —	Reserved
3-0 —	Reserved

28.5.3 SerDes PLLa Control Register 0 (PLL1CR0 - PLL2CR0)

28.5.3.1 Offset

For a = 1 to 2:

Register	Offset
PLLaCR0	-1Ch + (a × 20h)

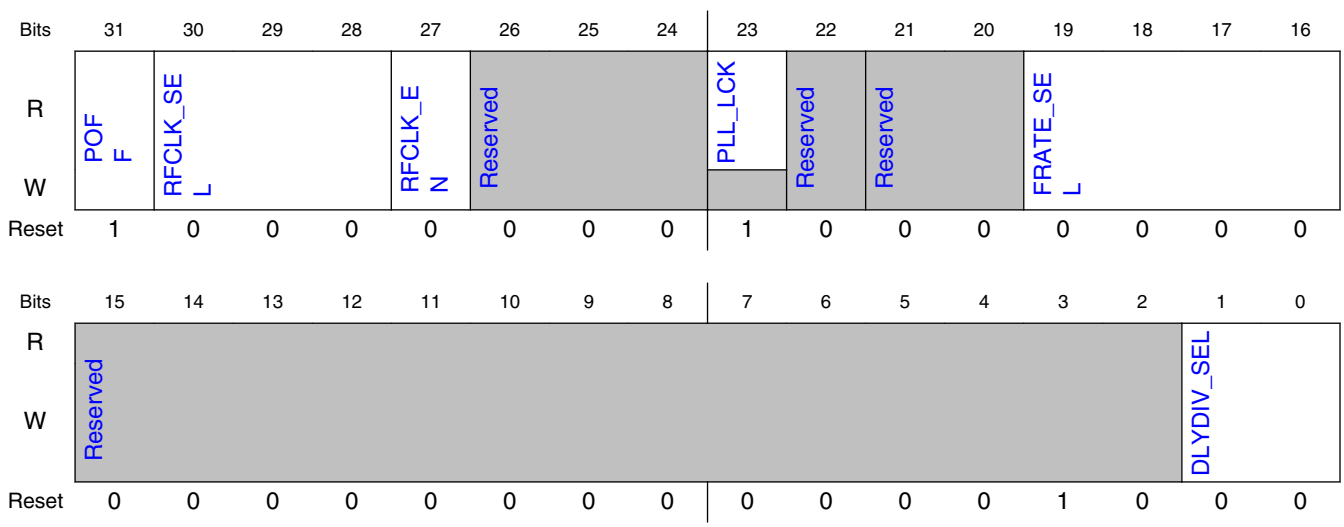
28.5.3.2 Function

PLLnCR0 contains control and status bits for SerDes PLL *n*

NOTE

This register may be automatically updated when PCI Express is active on the SerDes. Only write to this register when all associated PCI Express controllers are inactive.

28.5.3.3 Diagram



28.5.3.4 Fields

Field	Function
31 POFF	PLL off Power down an unused PLL Default value: 0

Table continues on the next page...

Field	Function
	0b - PLL on 1b - PLL off. All lanes referencing this PLL must also be disabled.
30-28 RFCLK_SEL	Reference clock frequency select All others reserved Default value set by RCW configuration. Recommended setting per protocol: SATA: 000, 001 or 011 PCIe: 000 or 001 SGMII @ 3.125: 001 or 010 SGMII @ 1.25: 000 or 001 000b - 100 MHz 001b - 125 MHz 010b - 011b - 150 MHz
27 RFCLK_EN	Reference clock observe enable Default value: 0 Recommended setting: 0 0b - Disable reference clock output 1b - Enable buffered version of SD_REF_CLKn
26-24 —	Reserved
23 PLL_LCK	PLL lock Indicates PLL(n) has calibrated and locked 0b - PLL is not locked 1b - PLL is locked
22 —	Reserved
21-20 —	Reserved
19-16 FRATE_SEL	Select frequency of PLL VCO. All lanes within a group must operate at a multiple of this frequency and within specified limits of the protocol(s). Default value set by RCW configuration and Recommended settings per protocol: All others are reserved 0000b - 5.00 GHz ---------- [SGMII, SATA, PCIe] 0111b - 4.00 GHz ---------- 1001b - 3.125 GHz -------- [SGMII, 1010b - 3.00 GHz ---------- [SATA]
15-2 —	Reserved
1-0 DLYDIV_SEL	Select PLLn_ex_dly_clk divider value: All other values reserved This feature is used for1000Base-KX. When those modes are not active, they should be off. 00 - PLLn_ex_dly_clk off

28.5.4 SerDes PLLa Control Register 1 (PLL1CR1 - PLL2CR1)

28.5.4.1 Offset

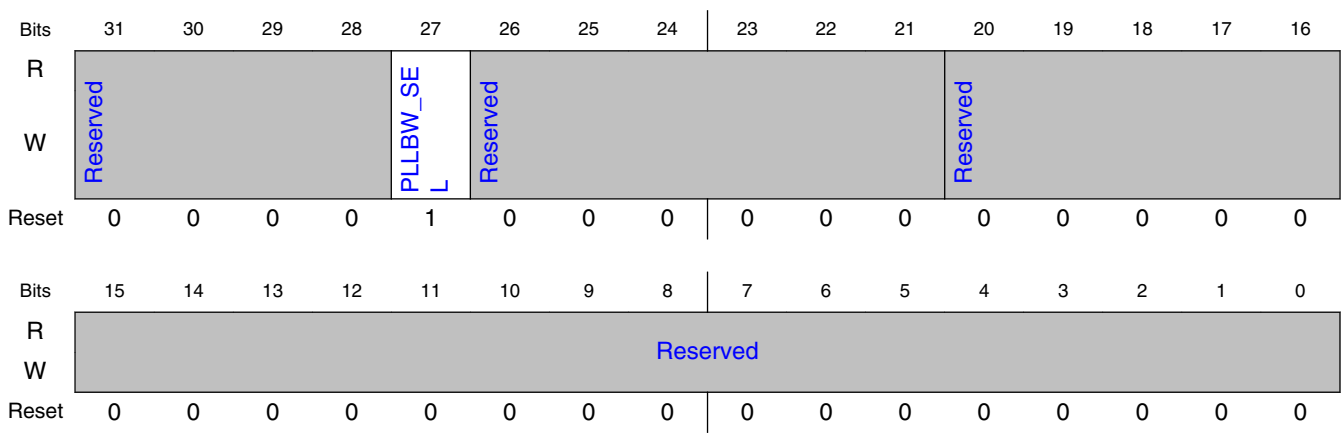
For a = 1 to 2:

Register	Offset
PLLaCR1	-18h + (a × 20h)

28.5.4.2 Function

PLLnCR1 contains control bits for SerDes PLL *n*.

28.5.4.3 Diagram



28.5.4.4 Fields

Field	Function
31-28 —	Reserved
27 PLLBW_SEL	Select Higher PLL(n) Bandwidth Recommended setting per PLL: 1

Table continues on the next page...

Field	Function
	0b - Nominal PLL Bandwidth 1b - PLL Bandwidth Setting
26-21 —	Reserved
20-0 —	Reserved

28.5.5 SerDes PLLa Control Register 5 (PLL1CR5 - PLL2CR5)

28.5.5.1 Offset

For a = 1 to 2:

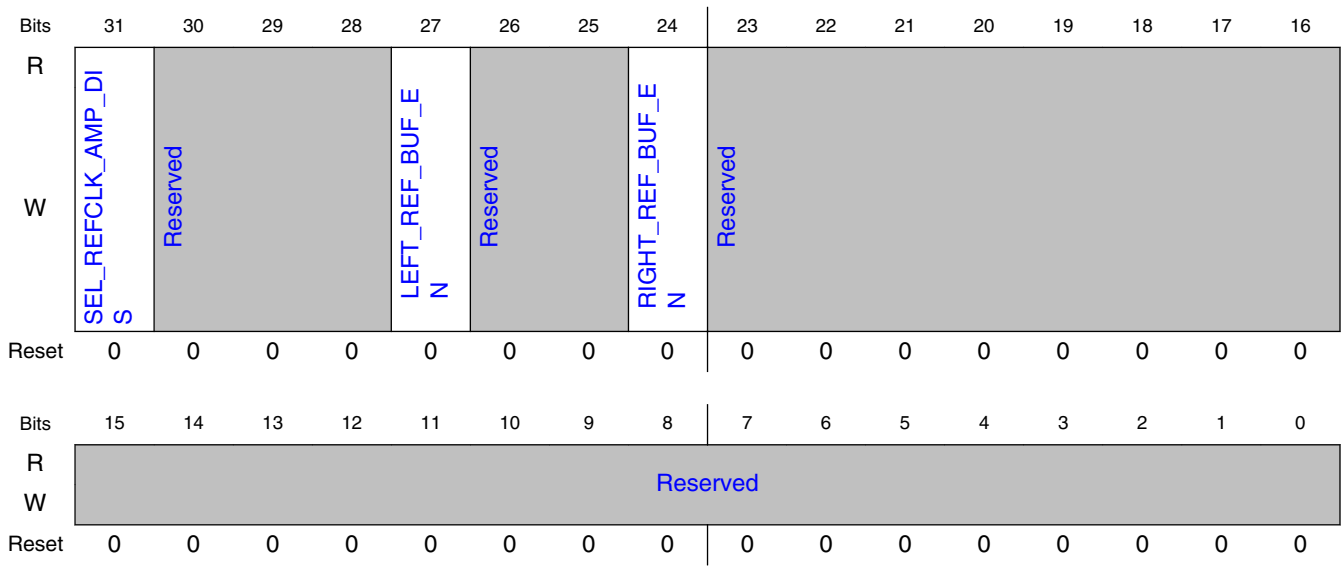
Register	Offset
PLLaCR5	-8h + (a × 20h)

28.5.5.2 Function

PLLnCR5 contains control bits for the SerDes x PLL *n*.

This register may only be configured by RCW.

28.5.5.3 Diagram



28.5.5.4 Fields

Field	Function
31 SEL_REFCLK_AMP_DIS	Disables the reference clock amplifier and selects usage of the reference clock coming from the left, either from SoC or from another 10G Serdes PLL Default value set by RCW
30-28 —	Reserved
27 LEFT_REF_BUF_EN	Enable for left directed reference clock buffer
26-25 —	Reserved
24 RIGHT_REF_BUF_EN	Enable for right directed reference clock buffer
23-0 —	Reserved

28.5.6 SerDes Transmit Calibration Control Register (TCALCR)

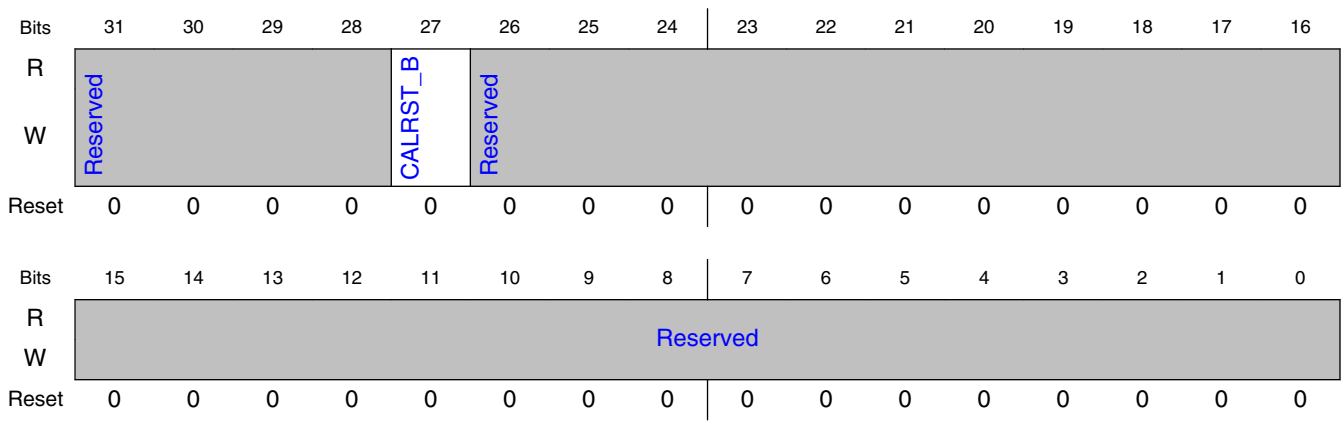
28.5.6.1 Offset

Register	Offset
TCALCR	90h

28.5.6.2 Function

TCALCR contains the control bits used for Transmit Calibration.

28.5.6.3 Diagram



28.5.6.4 Fields

Field	Function
31-28 —	Reserved
27 CALRST_B	Reset the transmit calibration During POR (warm reset sequence) it is active and is released when the first PLL comes out of reset. 0b - Reset 1b - Application Mode
26-0 —	Reserved

28.5.7 SerDes Transmit Calibration Control Register 1 (TCALCR1)

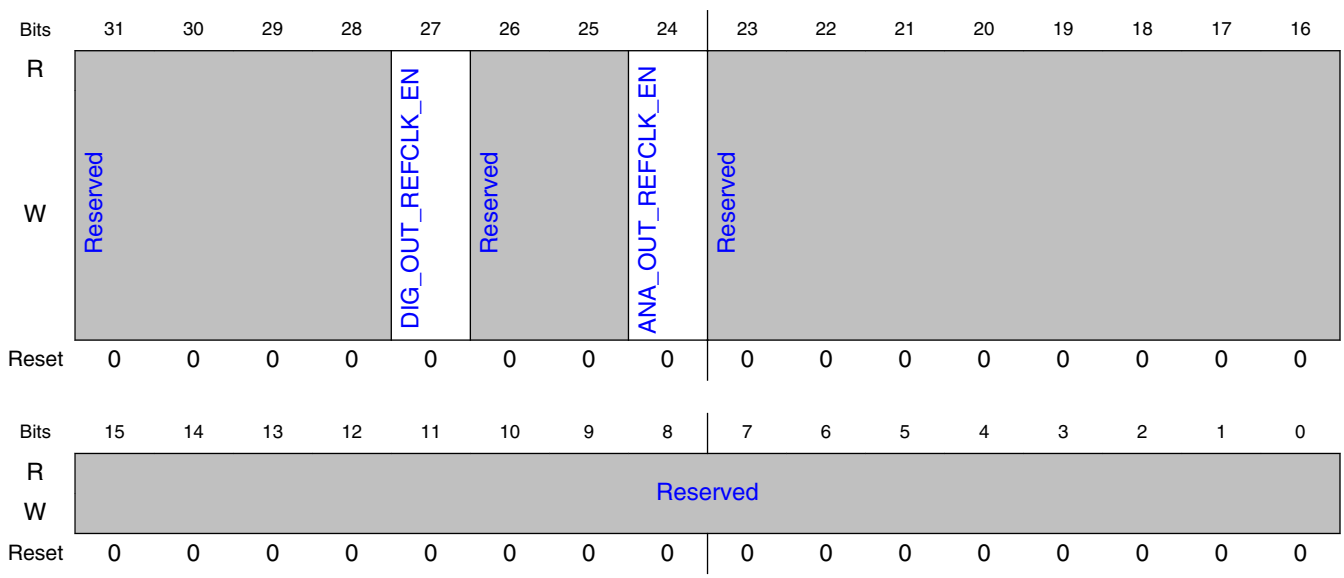
28.5.7.1 Offset

Register	Offset
TCALCR1	94h

28.5.7.2 Function

TCALCR1 contains the bits used for reference clock controls in the right half-lane.

28.5.7.3 Diagram



28.5.7.4 Fields

Field	Function
31-28 —	Reserved

Table continues on the next page...

Field	Function
27 DIG_OUT_REF CLK_EN	Enable CMOS digital buffered version of selected ref_clk to the right. Used in conjunction with PLLn_CR5[RIGHT_REF_BUF_EN]
26-25 —	Reserved
24 ANA_OUT_REF CLK_EN	Enable CML analog buffered version of selected ref_clk to the left. Used in conjunction with PLLn_CR5[RIGHT_REF_BUF_EN].
23-0 —	Reserved

28.5.8 Receive Calibration Control Register (RCALCR)

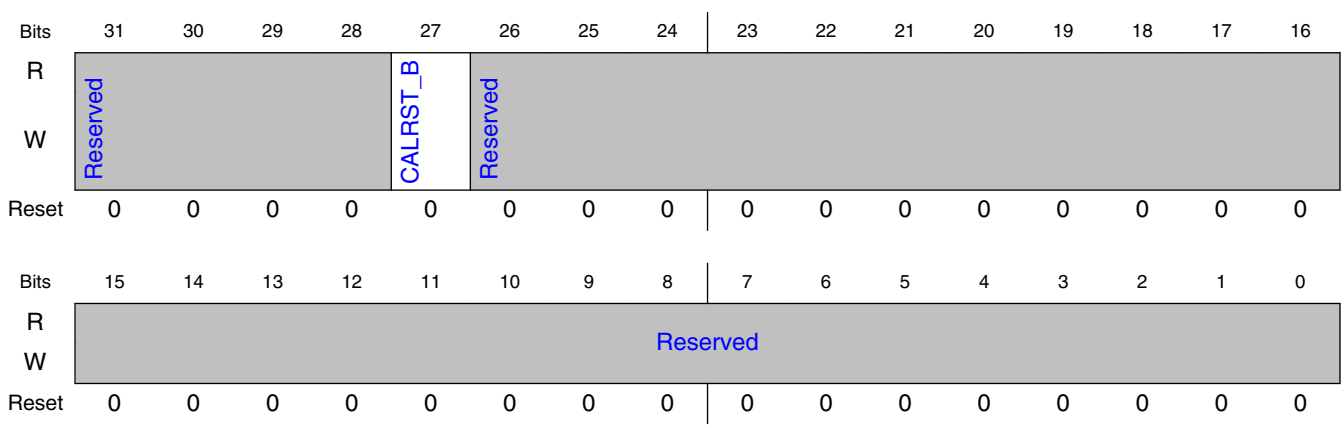
28.5.8.1 Offset

Register	Offset
RCALCR	A0h

28.5.8.2 Function

RCALCR contains the control bits for Receive Calibration.

28.5.8.3 Diagram



28.5.8.4 Fields

Field	Function
31-28 —	Reserved
27 CALRST_B	Reset the receiver calibration During POR (warm reset sequence) it is active and is released when the first PLL comes out of reset 0b - Reset 1b - Application Mode
26-0 —	Reserved

28.5.9 SerDes Receive Calibration Control Register 1 (RCALCR1)

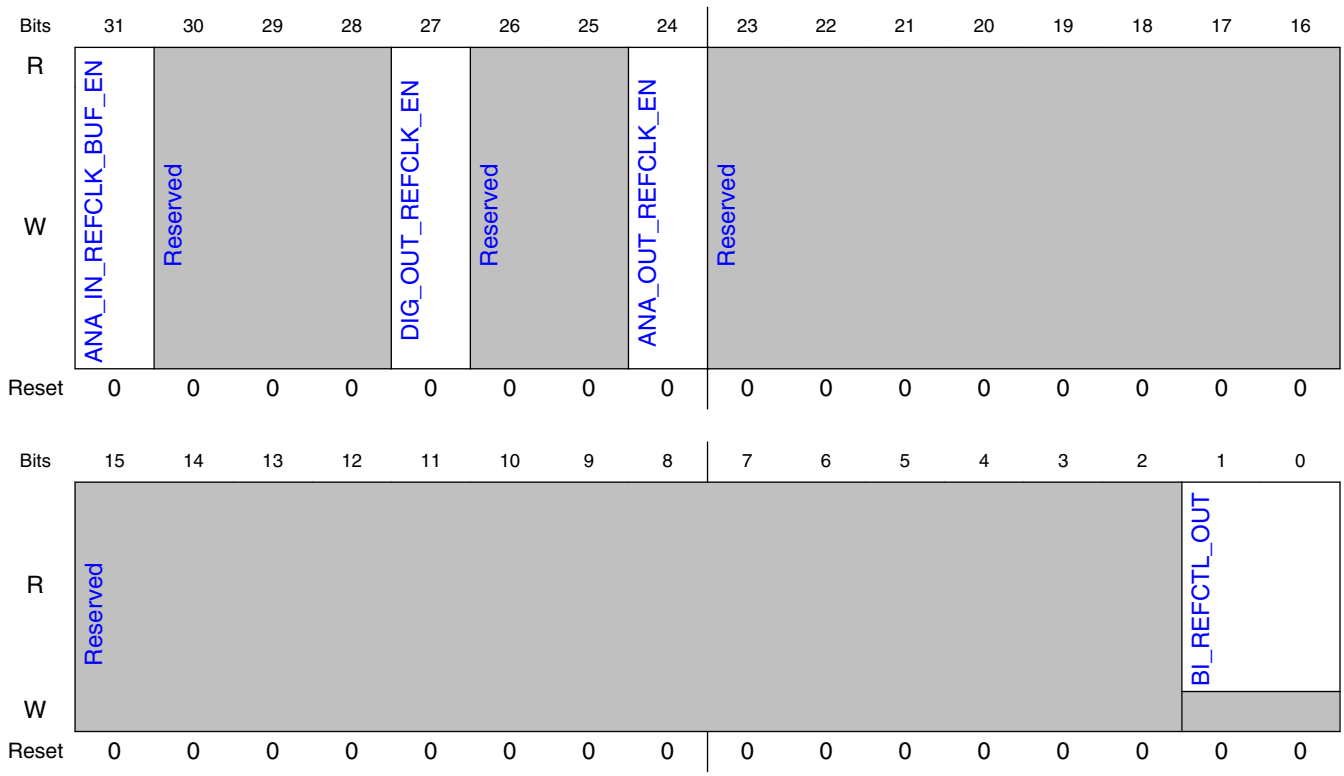
28.5.9.1 Offset

Register	Offset
RCALCR1	A4h

28.5.9.2 Function

RCALCR1 contains the bits for reference clock controls on the left half-lane.

28.5.9.3 Diagram



28.5.9.4 Fields

Field	Function
31 ANA_IN_REFCLK_BUF_EN	Enable CML analog buffer used to transition SoC reference clock to 10G Serdes
30-28 —	Reserved
27 DIG_OUT_REFCLK_EN	Enable CMOS digital buffered version of selected ref_clk to the left. Used in conjunction with PLLn_CR5[LEFT_REF_BUF_EN].
26-25 —	Reserved
24 ANA_OUT_REFCLK_EN	Enable CML analog buffered version of selected ref_clk to the left. Used in conjunction with PLLn_CR5[LEFT_REF_BUF_EN].
23-2 —	Reserved

Table continues on the next page...

Field	Function
1-0 BI_REFCTL_OUT	Output COP DFT/BurnIn/JTAG reference clock controls to wrapper

28.5.10 General Control Register 0 (GR0)

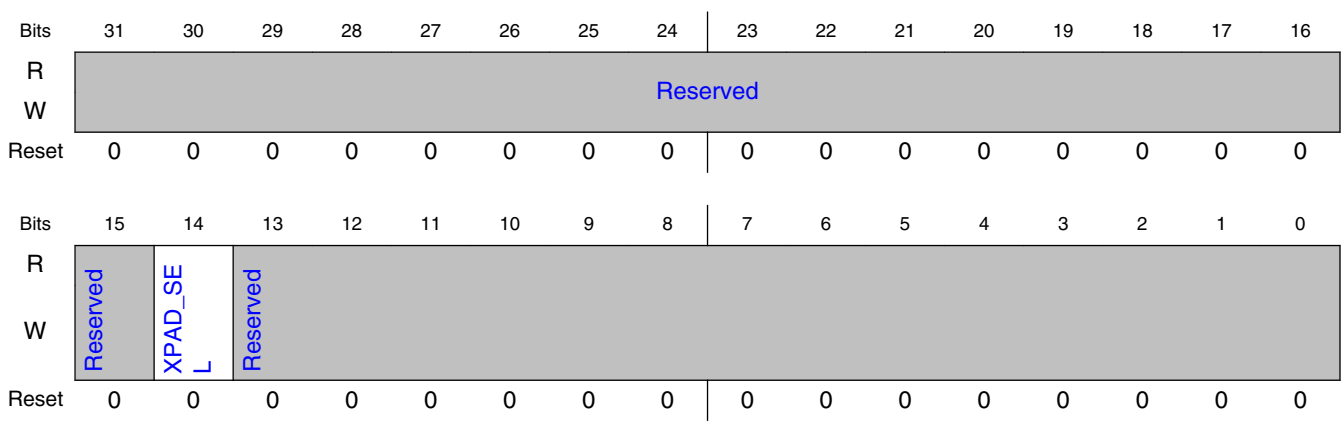
28.5.10.1 Offset

Register	Offset
GR0	B0h

28.5.10.2 Function

GR0 contains general control/status bits for the SerDes.

28.5.10.3 Diagram



28.5.10.4 Fields

Field	Function
31-15	Reserved

Table continues on the next page...

Field	Function
—	
14 XPAD_SEL	Describes to SerDes module the value of the power supply being used by the Serdes I/Os: Full SerDes reset required after changing this value. **Warning**: setting XPAD_SEL to low xpadvdd supply when the supply is > 1.35V+5% may cause irreparable damage to the device. 0b - High xpadvdd supply (nominal 1.5V - see H/W spec for details) 1b - Low xpadvdd supply (nominal 1.35V - see H/W spec for details)
13-0 —	Reserved

28.5.11 Protocol Configuration Register 0 (PCCR0)

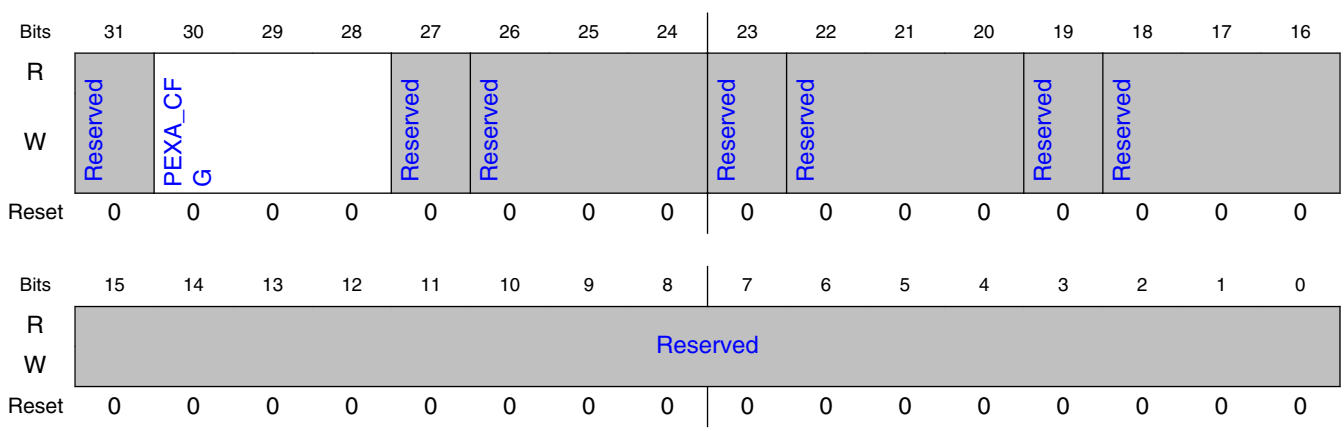
28.5.11.1 Offset

Register	Offset
PCCR0	200h

28.5.11.2 Function

PCCR0 contains the protocol configuration for PCIe.

28.5.11.3 Diagram



28.5.11.4 Fields

Field	Function
31 —	Reserved
30-28 PEXA_CFG	PEXa Configuration All others reserved Default value set by RCW configuration. 000b - disabled 001b - x1 on Lane 1 010b - x1 on Lane 2 011b - disabled 100b - disabled
27 —	Reserved
26-24 —	Reserved
23 —	Reserved
22-20 —	Reserved
19 —	Reserved
18-16 —	Reserved
15-0 —	Reserved

28.5.12 Protocol Configuration Register 2 (PCCR2)

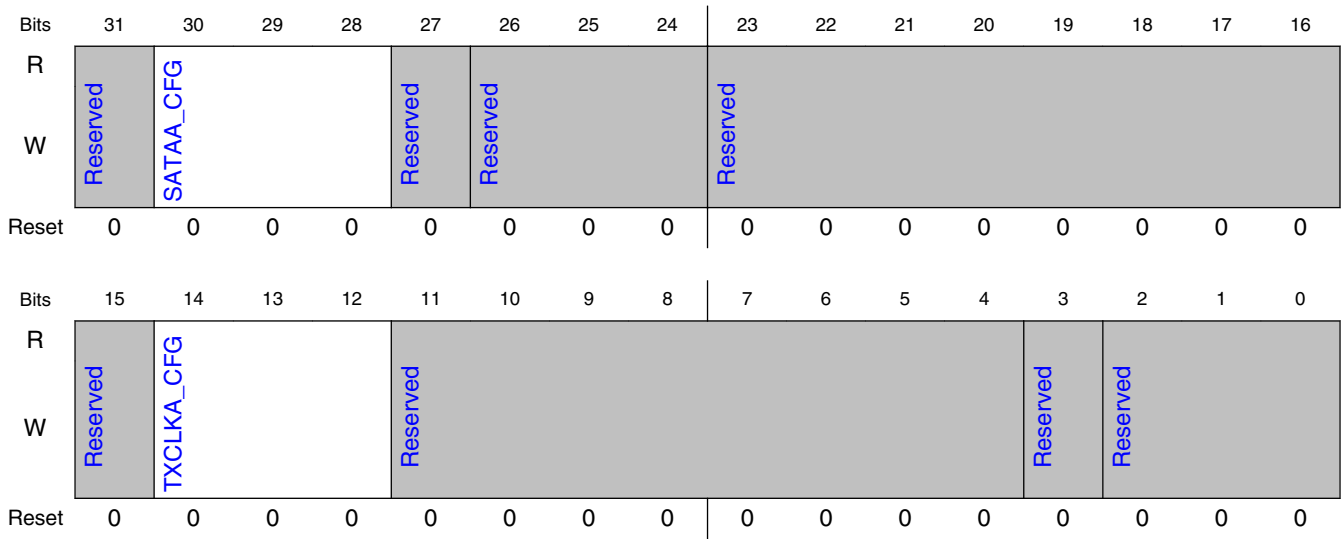
28.5.12.1 Offset

Register	Offset
PCCR2	208h

28.5.12.2 Function

PCCR2 contains the protocol configuration for SATA and Aurora.

28.5.12.3 Diagram



28.5.12.4 Fields

Field	Function
31 —	Reserved
30-28 SATAA_CFG	SATAa Configuration All others reserved 000b - Disabled 001b - x1 on Lane 2
27 —	Reserved
26-24 —	Reserved
23-15 —	Reserved
14-12 TXCLKA_CFG	TXCLKa Configuration All others reserved 000b - Disabled

Table continues on the next page...

Field	Function
	001b - Enabled on Lane 0 010b - Enabled on Lane 1
11-4 —	Reserved
3 —	Reserved
2-0 —	Reserved

28.5.13 Protocol Configuration Register 8 (PCCR8)

28.5.13.1 Offset

Register	Offset
PCCR8	220h

28.5.13.2 Function

PCCR8 contains the protocol configuration for SGMII /1000Base-KX.

28.5.13.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SGMIIA_KX	SGMIIA_CFG			SGMIIB_KX	SGMIIB_CFG			Reserved	Reserved			Reserved	Reserved		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

28.5.13.4 Fields

Field	Function
31 SGMIIA_KX	SGMIIb 1000Base-KX Configuration: Note: this configuration bit must be set before performing any MDIO initialization of the PCS. 0b - SGMII mode 1b - 1000Base-KX mode
30-28 SGMIIA_CFG	SGMIIa Configuration All others reserved Default value set by RCW configuration. 000b - disabled 001b - x1 on Lane 0 to WRIOP Mac 9
27 SGMIIB_KX	SGMIIc 1000Base-KX Configuration: Note: this configuration bit must be set before performing any MDIO initialization of the PCS. 0b - SGMII mode 1b - 1000Base-KX mode
26-24 SGMIIB_CFG	SGMIIb Configuration All others reserved Default value set by RCW configuration. 000b - disabled 001b - x1 on Lane 1 to WRIOP Mac 5
23 —	Reserved
22-20 —	Reserved
19 —	Reserved
18-16 —	Reserved
15 —	Reserved
14-12 —	Reserved
11 —	Reserved
10-8 —	Reserved
7 —	Reserved
6-4	Reserved

Table continues on the next page...

Field	Function
—	
3 —	Reserved
2-0 —	Reserved

28.5.14 General Control Register 0 - Lane a (LNAGCR0 - LNDG CR0)

28.5.14.1 Offset

Register	Offset
LNAGCR0	800h
LNBGCR0	840h
LNDGCR0	8C0h

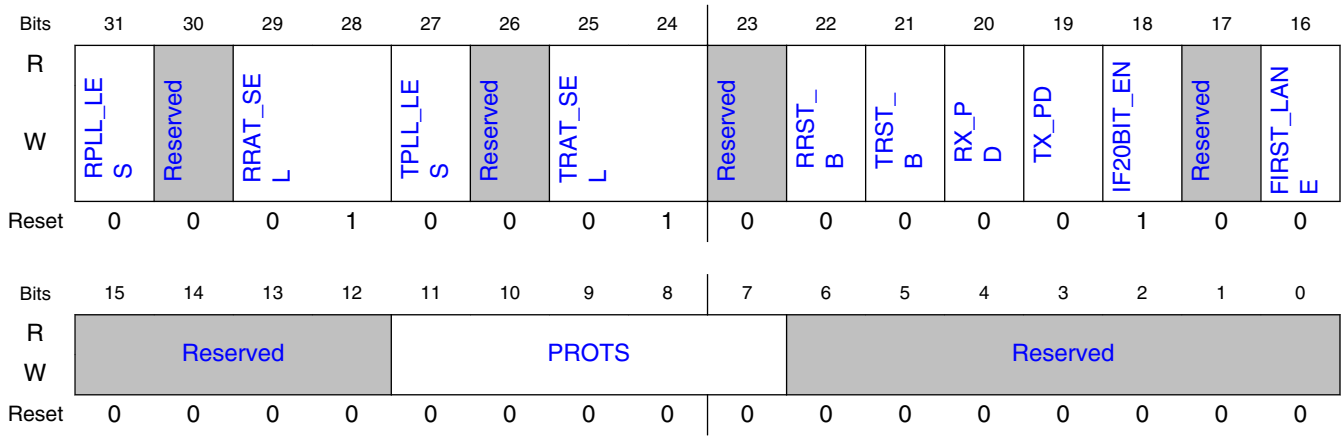
28.5.14.2 Function

LNmGCR0 contains functional control bits for SerDes lane *m*.

Special consideration must be taken when writing this register while PLLnRSTCTL[RST_DONE]=0, or if PCIe is running on this lane. See RPLL_LES, TPLL_LES, RRST_B and TRST_B fields for details.

See , for details on the sequence required to change settings.

28.5.14.3 Diagram



28.5.14.4 Fields

Field	Function
31 RPLL_LES	Directs the RX portion of lane a to use the corresponding PLL. Default value set by reset configuration. Note: must be set the same as TPLL_LES for normal function. 0b - use PLL2 1b - use PLL1
30 —	Reserved
29-28 RRAT_SEL	Receiver speed selection for lane a, relative to the corresponding PLLnCR0[FRATE_SEL], n as selected by LNaGCR0[RPLL_LES]: Default value set by reset configuration. Required setting per protocol: SGMII 10 (1.25 Gbaud) SGMII 00 (3.125 Gbaud) PCIe 01 SATA 10 This register field is read-only when running in PCIeor SATA mode. The above required values are defaults for gen1 initialization. Note: must be set the same as TRAT_SEL for normal function. 00b - Same as FRATE_SEL 01b - FRATE_SEL/2 10b - FRATE_SEL/4 11b - FRATE_SEL*2
27 TPLL_LES	Used to direct the TX portion of lane to use the corresponding PLL. Default value set by reset configuration.

Table continues on the next page...

Field	Function
	Note: must be set the same as RPLL_LES for normal function. See for details on the sequence required to change this setting. 0b - use PLL2 1b - use PLL1
26 —	Reserved
25-24 TRAT_SEL	Transmitter speed selection for lane a, relative to the corresponding PLLnCR0[FRATE_SEL], n as selected by LNaGCR0[TPLL_LES]: Default value set by reset configuration. Required setting per protocol: see RRAT_SEL This register field is read-only when running in PCIeor SATA mode. Note: must be set the same as RRAT_SEL for normal function. 00b - Same as FRATE_SEL 01b - FRATE_SEL/2 10b - FRATE_SEL/4 11b - FRATE_SEL*2
23 —	Reserved
22 RRST_B	Resets receiver for lane a Recommended setting per protocol: 1, unless PLLnRSTCTL[RST_DONE]=0, then 0. This register value is read-only during SerDes reset (POR or PLLnRSTCTL[RSTREQ]). This register field is read-only when running in PCIeor SATA mode. In SATA mode, this field may read as 1 while a controller-driven lane reset is in progress. Used in lane reset and reconfiguring procedures. See Lane Reset and Reconfiguration, for details. 0b - Reset 1b - Application Mode
21 TRST_B	Resets transmitter for lane a Recommended setting per protocol: 1 This register value is read-only during SerDes reset (POR or PLLnRSTCTL[RSTREQ]). This register field is read-only when running in PCIeor SATA mode. Used in lane reset and reconfiguring procedures. See Lane Reset and Reconfiguration, for details. 0b - Reset 1b - Application Mode
20 RX_PD	Lane powerdown for receiver on lane a Default value set by reset configuration. This register field read-only during SerDes reset (POR or PLLnRSTCTL[RSTREQ]). See Lane Enable After Powerdown for details on re-enabling powered down lanes. 0b - Lane receiver active 1b - Lane receiver powered down
19 TX_PD	Lane powerdown for transmitter on lane a Default value set by reset configuration. This register value is read-only during SerDes reset (POR or PLLnRSTCTL[RSTREQ]).

Table continues on the next page...

Field	Function
	Note: the user must not assert TX_PD for the master clock lane (TX_CLK_1STLANE=1) of a multi-lane link during normal function. See Lane Enable After Powerdown for details on e-enabling powered down lanes. 0b - Lane transmitter active 1b - Lane transmitter powered down
18 IF20BIT_EN	20-bit interface enable Default value set by reset configuration. Required value per protocol: PCIe 1 SGMII 0 SATA 1 0b - 10-bit interface 1b - 20-bit interface
17 —	Reserved
16 FIRST_LANE	Indicates this lane is the first (lane 0) of a group of lanes Default/recommended value set by reset configuration 0b - Lane a is not lane 0 of the link 1b - Lane a is lane 0 of the link or unused lane
15-12 —	Reserved
11-7 PROTS	Lane Protocol Select All others reserved 00000b - PCI EXP 00001b - SGMII/1000Base-KX , , SGMII 2.5x 00010b - SATA
6-0 —	Reserved

28.5.15 General Control Register 1 - Lane a (LNAGCR1 - LNDG CR1)

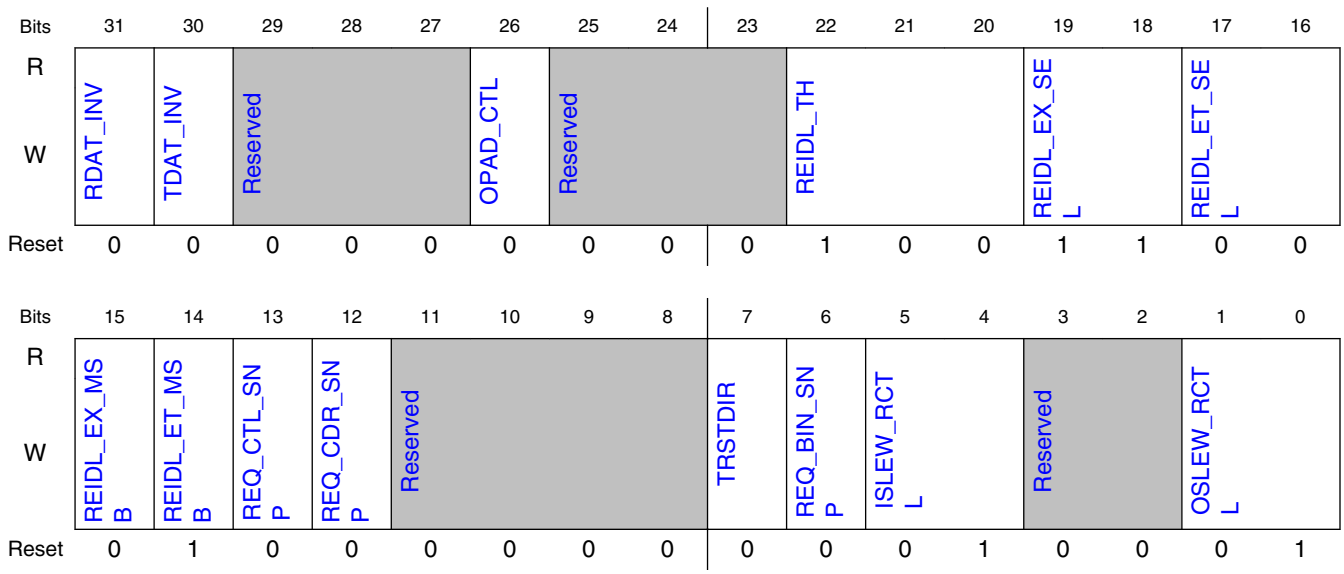
28.5.15.1 Offset

Register	Offset
LNAGCR1	804h
LNBGCR1	844h
LNDGCR1	8C4h

28.5.15.2 Function

LNmGCR1 contains functional control bits for SerDes lane a.

28.5.15.3 Diagram



28.5.15.4 Fields

Field	Function
31 RDAT_INV	Invert Rx data. Has the same effect as swapping SD_RX[m]_P and SD_RX[m]_N. Note: this field is read-only when this lane is operating as PCIe. 0b - Rx data is not inverted 1b - Rx data is inverted before it is decoded
30 TDAT_INV	Invert Tx data. Has the same effect as swapping SD_TX[m]_P and SD_TX[m]_N. 0b - Tx data is not inverted 1b - Tx data is inverted before it is transmitted
29-27 —	Reserved
26 OPAD_CTL	TX Output pad control signal for common mode Note: this field is read-only when this lane is operating as PCIe, SATA, SGMII. 0b - Transmitter Enabled 1b - Force Transmitter Output to Common Mode
25-23	Reserved

Table continues on the next page...

Field	Function
—	
22-20 REIDL_TH	Receiver electrical idle detection threshold control Default value set by RCW configuration Recommended value per protocol: PCIe: 100 (2.5 Gbaud) SGMII: 001 (1.25 Gbaud) SGMII: 000 (3.125 Gbaud) SATA: 100 (1.5 Gbaud) Others: 000 Note: SATA 3 Gbaud and PCIe 5 Gbaud setting taken from LNaSSCR0[REIDL_TH_0). Note: SATA 6 Gbaud setting taken from LNaSSCR1[REIDL_TH_1).
19-18 REIDL_EX_SEL	Exit electrical idle filter select = {REIDL_EX_MSB,REIDL_EX_SEL} Default value set by RCW configuration Recommended value per protocol: PCIe: 011 (2.5 Gbaud) SGMII: 011 (1.25 Gbaud) SGMII: 000 (3.125 Gbaud) 1GKX: 000 SATA: 001 (1.5 Gbaud) Others: 000 Note: SATA 3 Gbaud and PCIe 5 Gbaud setting taken from LNaSSCR0[REIDL_EX_SEL_0). Note: SATA 6 Gbaud setting taken from LNaSSCR1[REIDL_EX_SEL_1).
17-16 REIDL_ET_SEL	Enter idle filter select = {REIDL_ET_MSB,REIDL_ET_SEL}: Recommended setting per protocol: PCIe: 111 (2.5 Gbaud) SGMII: 100 (1.25 Gbaud) SGMII: 000 (3.125 Gbaud) 1GKX: 000 SATA: 001 (1.5 Gbaud) Others: 000 Note: SATA 3 Gbaud and PCIe 5 Gbaud setting taken from LNaSSCR0[REIDL_ET_SEL_0). Note: SATA 6 Gbaud setting taken from LNaSSCR1[REIDL_ET_SEL_1).
15 REIDL_EX_MSB	Exit idle filter select MSB. See REIDL_EX_SEL for settings. Note: SATA 3 Gbaud and PCIe 5 Gbaud setting taken from LNaSSCR0[REIDL_EX_MSB_0). Note: SATA 6 Gbaud setting taken from LNaSSCR1[REIDL_EX_MSB_1).
14 REIDL_ET_MSB	Enter idle filter select MSB. See REIDL_ET_SEL for settings. Note: SATA 3 Gbaud and PCIe 5 Gbaud setting taken from LNaSSCR0[REIDL_ET_MSB_0). Note: SATA 6 Gbaud setting taken from LNaSSCR1[REIDL_ET_MSB_1).
13	Initiate snapshot of RX Equalization Control Gaink2, Gaink3, and Offset Registers

Table continues on the next page...

Field	Function
REQ_CTL_SNP	Recommended Setting per protocol: 0
12 REQ_CDR_SNP	nitiate snapshot of RX Clock/Data Recovery (CDR) Registers Recommended Setting per protocol: 0
11-8 —	Reserved
7 TRSTDIR	Multi-lane protocol Tx clock synchronization control Default value set by RCW configuration. Recommended setting per protocol: PCIe: Others: 0
6 REQ_BIN_SNP	Initiate snapshot of RX Equalization Control Binning Registers Note: this field is read-only when this lane is operating as PCIe
5-4 ISLEW_RCTL	Slew control for Quadrature Generator Default value set by RCS configuration. Recommended setting per protocol: PCIe: 01 (2.5 Gbaud) SGMII: 01 (1.25 Gbaud) SGMII: 10 (3.125 Gbaud) SATA: 10 (1.5 Gbaud) Note: SATA 3 Gbaud and PCIe 5 Gbaud setting taken from LNaSSCR0[ISLEW_RCTL_0). Note: SATA 6 Gbaud setting taken from LNaSSCR1[ISLEW_RCTL_1).
3-2 —	Reserved
1-0 OSLEW_RCTL	Phase Interpolator Output clock edge rate control Default value set by RCW configuration. Recommended setting per protocol: PCIe: 01 (2.5 Gbaud) SGMII: 01 (1.25 Gbaud) SGMII: 10 (3.125 Gbaud) SATA: 10 (1.5 Gbaud) Note: SATA 3 Gbaud and PCIe 5 Gbaud setting taken from LNaSSCR0[OSLEW_RCTL_0). Note: SATA 6 Gbaud setting taken from LNaSSCR1[OSLEW_RCTL_1).

28.5.16 Speed Switch Control Register 0 - Lane a (LNASSCR0 - LNDSSCR0)

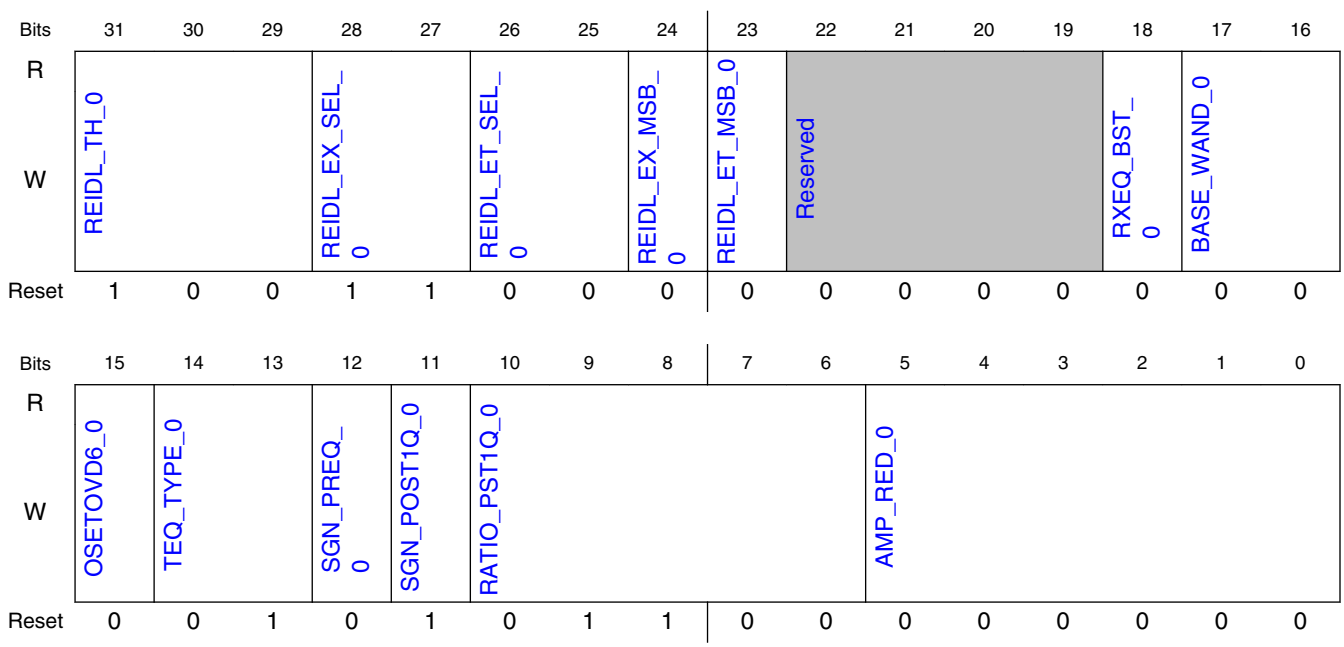
28.5.16.1 Offset

Register	Offset
LNASSCR0	80Ch
LNBSSCR0	84Ch
LNDSSCR0	8CCh

28.5.16.2 Function

LNmSSCR0 contains control bits for modifying the PCI Express 5 Gbaud and SATA 3 Gbaud behavior of SerDes lane a.

28.5.16.3 Diagram



28.5.16.4 Fields

Field	Function
31-29 REIDL_TH_0	Receiver electrical idle detection threshold control Default settings set per RCW configuration. Recommended settings per protocol:

Table continues on the next page...

Field	Function
	PCIe: 100 (5 Gbaud) SATA: 101 (3 Gbaud) Others: 000 For PCIe 2.5 Gbaud and SATA 1.5 Gbaud and protocols other than PCIe or SATA, see LNaGCR1[REIDL_TH]. For SATA 6 Gbaud, see LNaSSCR1[REIDL_TH_1].
28-27 REIDL_EX_SEL_0	Exit electrical idle filter select = {REIDL_EX_MSB_0,REIDL_EX_SEL_0} for PCIe 5 Gbaud and SATA 3 Gbaud Default settings set per RCW configuration. Recommended settings per protocol: PCIe: 011 (5 Gbaud) SATA: 001 (3 Gbaud) Others: 000 For PCIe 2.5 Gbaud and SATA 1.5 Gbaud and protocols other than PCIe or SATA, see LNaGCR1[REIDL_EX_SEL]. For SATA 6 Gbaud, see LNaSSCR1[REIDL_EX_SEL_1].
26-25 REIDL_ET_SEL_0	Enter idle filter select = {REIDL_ET_MSB_0,REIDL_ET_SEL_0} for PCIe 5 Gbaud and SATA 3 Gbaud: Default settings set per RCW configuration. Recommended settings per protocol: PCIe: 000 (5 Gbaud) SATA: 001 (3 Gbaud) Others: 000 For PCIe 2.5 Gbaud and SATA 1.5 Gbaud and protocols other than PCIe or SATA, see LNaGCR1[REIDL_ET_SEL]. For SATA 6 Gbaud, see LNaSSCR1[REIDL_ET_SEL_1].
24 REIDL_EX_MSB_0	Exit idle filter select MSB. See REIDL_EX_SEL_0 for settings. For PCIe 2.5 Gbaud and SATA 1.5 Gbaud and protocols other than PCIe or SATA, see LNaGCR1[REIDL_EX_MSB]. For SATA 6 Gbaud, see LNaSSCR1[REIDL_EX_MSB_1].
23 REIDL_ET_MSB_0	Enter idle filter select MSB. See REIDL_ET_SEL_0 for settings. For PCIe 2.5 Gbaud and SATA 1.5 Gbaud and protocols other than PCIe or SATA, see LNaGCR1[REIDL_ET_MSB]. For SATA 6 Gbaud, see LNaSSCR1[REIDL_ET_MSB_1].
22-19 —	Reserved
18 RXEQ_BST_0	Rx Equalization Boost Default value set by RCW configuration Recommended value per protocol: 0 For PCIe 2.5 Gbaud and SATA 1.5 Gbaud, see LNaRECR0[RXEQ_BST]. For SATA 6 Gbaud, see LNaSSCR1[RXEQ_BST_1].
17-16	Baseline Wander Control Select

Table continues on the next page...

Field	Function
BASE_WAND_0	Default settings set per RCW configuration. Recommended settings per protocol: 00 For PCIe 2.5 Gbaud and SATA 1.5 Gbaud, see LNaRECR0[BASE_WAND]. For SATA 6 Gbaud, see LNaSSCR1[BASE_WAND_1]. 00b - off (8b10b data) 01b - default BinBLW threshold 10b - alternate BinBLW sign 11b - Use RX Eq offset as GainBLW override
15 OSETOVD6_0	Binary Decode of Lane Adaptive Equalization offset intitialization or override value. [6] = 1: Double Imposed Offset Default value set by RCW configuration Recommended setting per protocol: PCIe: 0 (5 Gbaud) SATA: 0 (3 Gbaud) Others: 0 For PCIe 2.5 Gbaud and SATA 1.5 Gbaud and protocols other than PCIe or SATA, see LNaRECR0[OSETOVD]. For SATA 6 Gbaud, see LNaSSCR1[OSETOVD6_1].
14-13 TEQ_TYPE_0	Lane transmit equalization. Default value set by RCW configuration Recommended setting per protocol: PCIe: 01 (5 Gbaud) SATA: 01 (3 Gbaud) Others: 00 For PCIe 2.5 Gbaud and SATA 1.5 Gbaud and protocols other than PCIe or SATA, see LNaTECR0[TEQ_TYPE]. For SATA 6 Gbaud, see LNaSSCR1[TEQ_TYPE_1]. 00b - No TX Equalization 01b - 2 Levels of TX Equalization (+1 post cursor) 10b - 3 Levels of TX Equalization (+1 pre-cursor and +1 post-cursor) 11b - Reserved
12 SGN_PREQ_0	Precursor sign Default value set by RCW configuration Recommended setting per protocol: PCIe: 0 (5 Gbaud) SATA: 0 (3 Gbaud) Others: 0 For PCIe 2.5 Gbaud and SATA 1.5 Gbaud and protocols other than PCIe or SATA, see LNaTECR0[SGN_PREQ]. For SATA 6 Gbaud, see LNaSSCR1[SGN_PREQ_1]. 0b - Negative Sign (close eye) 1b - Positive Sign (open eye)

Table continues on the next page...

Field	Function
11 SGN_POST1Q_0	Post1q sign Default value set by RCW configuration Recommended setting per protocol: PCIe: 1 (5 Gbaud) SATA: 1 (3 Gbaud) Others: 0 For PCIe 2.5 Gbaud and SATA 1.5 Gbaud and protocols other than PCIe or SATA, see LNaTECR0[SGN_POST1Q]. For SATA 6 Gbaud, see LNaSSCR1[SGN_POST1Q_1] 0b - Negative Sign (close eye) 1b - Positive Sign (open eye)
10-6 RATIO_PST1Q_0	Ratio of full swing transition bit to first post-cursor. Default value set by RCW configuration Recommended setting per protocol: PCIe: 0_1100 (5 Gbaud) SATA: 0_0010 (3 Gbaud) Others: 0_0000 For PCIe 2.5 Gbaud and SATA 1.5 Gbaud and protocols other than PCIe or SATA, see LNaTECR0[RATIO_PST1Q]. For SATA 6 Gbaud, see LNaSSCR1[RATIO_PST1Q_1]. Note: this field is read-only when this lane is operating as PCIe
5-0 AMP_RED_0	Overall TX Amplitude Reduction Default value set by RCW configuration Recommended setting per protocol: PCIe: 00_0000 (5 Gbaud) SATA: 00_0111 (3 Gbaud) Others: 00_0000 For PCIe 2.5 Gbaud and SATA 1.5 Gbaud and protocols other than PCIe or SATA, see LNaTECR0[AMP_RED]. For SATA 6 Gbaud, see LNaSSCR1[AMP_RED_1]. Note: this field is read-only when this lane is operating as PCIe

28.5.17 Receive Equalization Control Register 0 - Lane a (LNAR ECR0 - LNDRECR0)

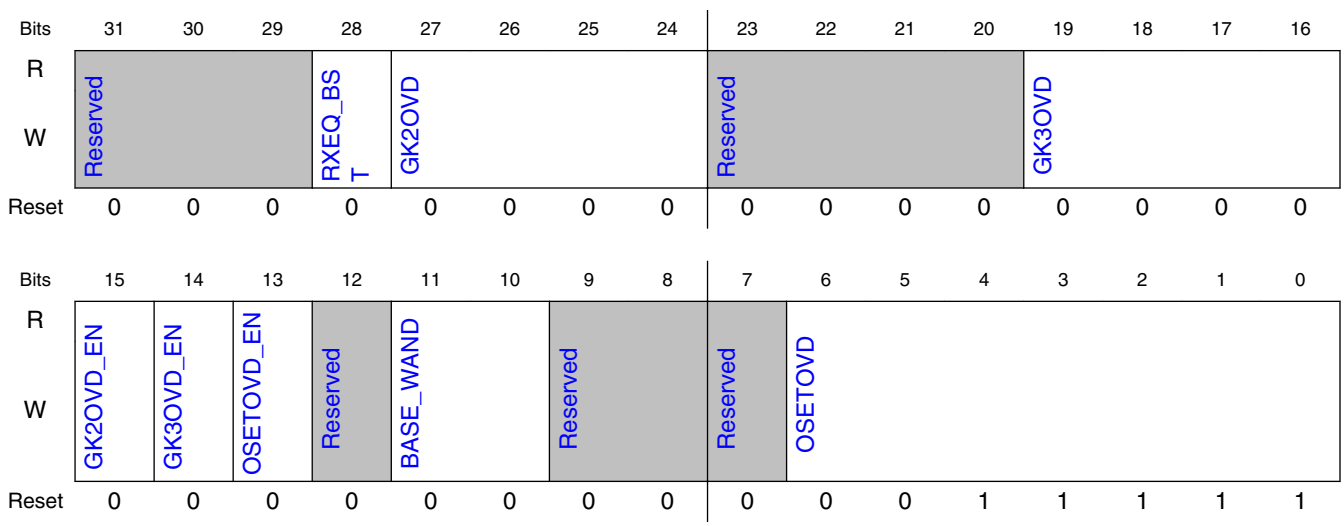
28.5.17.1 Offset

Register	Offset
LNARECR0	810h
LNBRECR0	850h
LNDRECR0	8D0h

28.5.17.2 Function

LNmRECR0 contains functional control bits for SerDes lane a

28.5.17.3 Diagram



28.5.17.4 Fields

Field	Function
31-29 —	Reserved
28 RXEQ_BST	Rx Equalization Boost Default value set by RCW configuration Recommended value per protocol: 0 Note: SATA 3 Gbaud and PCIe 5 Gbaud setting taken from LNaSSCR0[RXEQ_BST_0).

Table continues on the next page...

Field	Function
	Note: SATA 6 Gbaud setting taken from LNaSSCR1[RXEQ_BST_1).
27-24 GK2OVD	Binary decode of lane Adaptive Equalization gaink2 initialization or override value. Default value set by RCW configuration. Recommended settings per protocol: PCIe: 0000 (2.5 Gbaud) SGMII: 1111 (1.25 Gbaud) SGMII: 0000 (3.125 Gbaud) SATA: 0000 (1.5 Gbaud) Note: SATA 3 Gbaud and PCIe 5 Gbaud setting taken from LNaSSCR0[GK2OVD_0). Note: SATA 6 Gbaud setting taken from LNaSSCR1[GK2OVD_1).
23-20 —	Reserved
19-16 GK3OVD	Binary decode of lane Adaptive Equalization gaink3 intitialization or override value. Default value set by RCW config. Recommended settings per protocol: PCIe: 0000 (2.5 Gbaud) SGMII: 1111 (1.25 Gbaud) SGMII: 0000 (3.125 Gbaud) SATA: 0000 (1.5 Gbaud) Note: SATA 3 Gbaud and PCIe 5 Gbaud setting taken from LNaSSCR0[GK3OVD_0). Note: SATA 6 Gbaud setting taken from LNaSSCR1[GK3OVD_1).
15 GK2OVD_EN	Controls source of rx equalization "gaink2" setting. Recommended settings per protocol: SGMII: 1 (1.25 Gbaud) SGMII: 0 (3.125 Gbaud) Others: 0 Note: SATA 3 Gbaud and PCIe 5 Gbaud setting taken from LNaSSCR0[GK2OVD_EN_0). Note: SATA 6 Gbaud setting taken from LNaSSCR1[GK2OVD_EN_1). 0b - Use rxeq adaption derived gaink2 1b - Fix gaink2 according to GK2OVD
14 GK3OVD_EN	Controls source of rx equalization "gaink3" setting. Recommended settings per protocol: SGMII: 1 (1.25 Gbaud) SGMII: 0 (3.125 Gbaud) Others: 0 Note: SATA 3 Gbaud and PCIe 5 Gbaud setting taken from LNaSSCR0[GK3OVD_EN_0). Note: SATA 6 Gbaud setting taken from LNaSSCR1[GK3OVD_EN_1). 0 Use rxeq adaption derived gaink3 1 Fix gaink3 according to GK3OVD

Table continues on the next page...

Field	Function
13 OSETOVD_EN	Controls source of rx equalization "offset" setting. Default value set by RCW config Recommended setting per protocol: 0 0b - On release of srds_reset_b, initial rx eq offset is LNaRECR0[OSETOVD] and rx eq loop will begin adjustment at this value 1b - rx eq offset is fixed at LNaRECR0[OSETOVD]
12 —	Reserved
11-10 BASE_WAND	Baseline Wander Control Select Default value set by RCW config Recommended setting per protocol: Others: 00 Note: SATA 3 Gbaud and PCIe 5 Gbaud setting taken from LNaSSCR0[BASE_WAND_0). Note: SATA 6 Gbaud setting taken from LNaSSCR1[BASE_WAND_1). 00b - off (8b10b data) 01b - default BinBLW threshold 10b - alternate BinBLW sign 11b - Use OSETOVD[4:0] as GainBLW override
9-8 —	Reserved
7 —	Reserved
6-0 OSETOVD	Binary Decode of Lane Adaptive Equalization offset intitialization or override value. [6] 1: Double Imposed Offset [5:0] b00 0000: + Maximum Imposed Offset [5:0] b01 1111: No imposed offset [5:0] b11 1111: - Maximum Imposed Offset Note: If BASE_WAND= 11, then use OSETOVD[4:0] as GainBLW Override Default value set by RCW config Recommended setting per protocol: Others: 001_1111 Note: SATA 3 Gbaud and PCIe 5 Gbaud setting taken from LNaSSCR0[OSETOVD_0). Note: SATA 6 Gbaud setting taken from LNaSSCR1[OSETOVD_1)

28.5.18 Receive Equalization Control Register 1- Lane a (LNARECR1 - LNDRECR1)

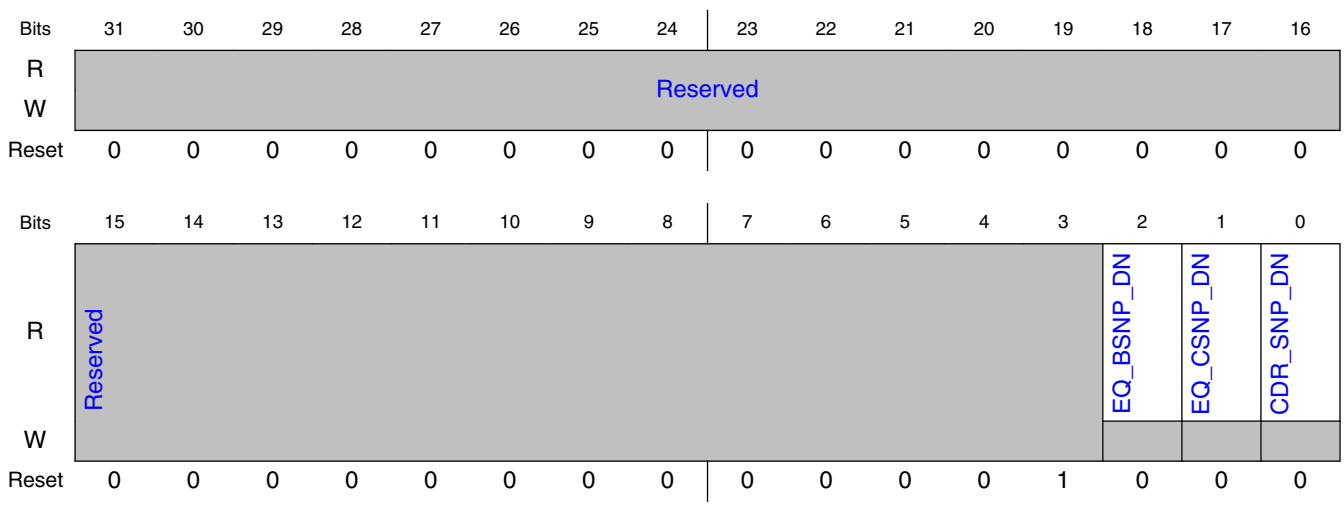
28.5.18.1 Offset

Register	Offset
LNARECR1	814h
LNBRECR1	854h
LNDRECR1	8D4h

28.5.18.2 Function

LNmRECR1 contains control and status bits for receiver equalization on lane a.

28.5.18.3 Diagram



28.5.18.4 Fields

Field	Function
31-3 —	Reserved
2	Snapshot of RX EQ Bin Complete

Table continues on the next page...

Field	Function
EQ_BSNP_DN	
1 EQ_CSNP_DN	Snapshot of RX EQ Ctrl Complete
0 CDR_SNP_DN	Snapshot of CDR Loop Complete

28.5.19 Transmit Equalization Control Register 0 - Lane a (LNATECR0 - LNDTECR0)

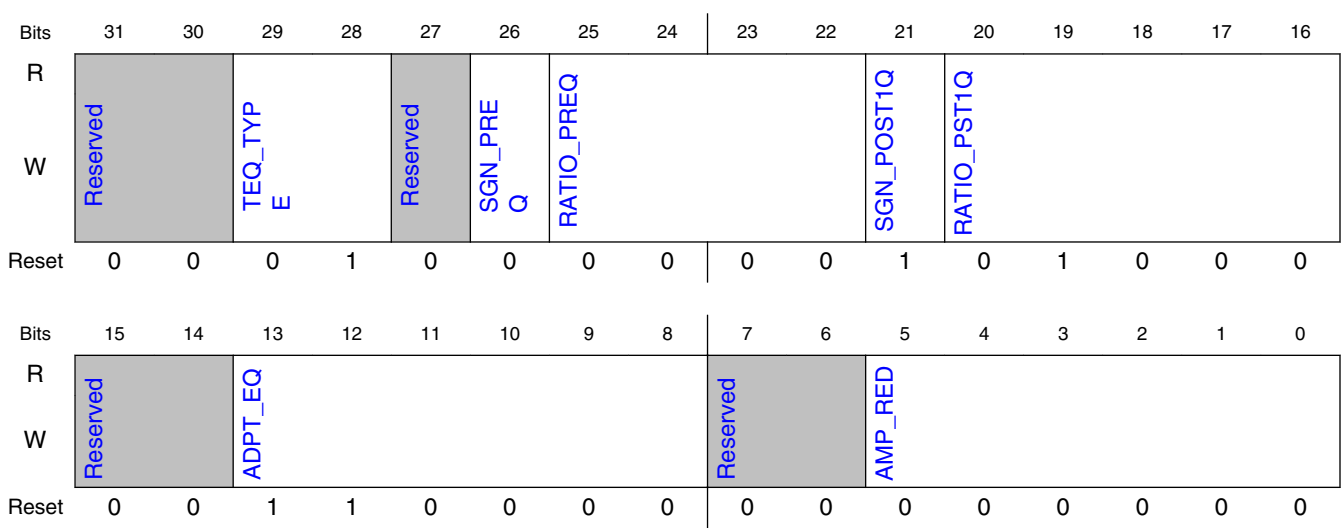
28.5.19.1 Offset

Register	Offset
LNATECR0	818h
LNBTECR0	858h
LNDTECR0	8D8h

28.5.19.2 Function

LNmTECR0 contains Tx equalization control bits for SerDes lane a.

28.5.19.3 Diagram



28.5.19.4 Fields

Field	Function
31-30 —	Reserved
29-28 TEQ_TYPE	Selects amount/type of Transmit Equalization Default value set by RCW configuration. Recommended settings per protocol: SGMII: 00 (1.25 Gbaud) SGMII: 01 (3.125 Gbaud) Others: 01 Note: SATA 3 Gbaud and PCIe 5 Gbaud setting taken from LNaSSCR0[TEQ_TYPE_0). Note: SATA 6 Gbaud setting taken from LNaSSCR1[TEQ_TYPE_1) 00b - No TX Equalization 01b - 2 Levels of TX Equalization (+1 post cursor) 10b - 3 Levels of TX Equalization (+1 pre-cursor and +1 post-cursor) 11b - Reserved
27 —	Reserved
26 SGN_PREQ	Precursor sign Default value set by RCW configuration. Recommended settings per protocol: Others: 0 Note: SATA 3 Gbaud and PCIe 5 Gbaud setting taken from LNaSSCR0[SGN_PREQ_0). Note: SATA 6 Gbaud setting taken from LNaSSCR1[SGN_PREQ_1) 0b - Negative Sign(close eye) 1b - Positive Sign (open eye)
25-22 RATIO_PREQ	Ratio of full swing transition bit to pre-cursor Default value set by RCW configuration. Recommended settings per protocol: Others: 0000 Note: this field is read-only when this lane is operating as PCIe
21 SGN_POST1Q	Post q Sign Default value set by RCW configuration. Recommended settings per protocol: SGMII: 0 (1.25 Gbaud) SGMII: 1 (3.125 Gbaud) Others: 1 Note: SATA 3 Gbaud and PCIe 5 Gbaud setting taken from LNaSSCR0[SGN_POST1Q_0)

Table continues on the next page...

Field	Function
	Note: SATA 6 Gbaud setting taken from LNaSSCR1[SGN_POST1Q_1). 0b - Negative Sign (close eye) 1b - Positive Sign (open eye)
20-16 RATIO_PST1Q	Ratio of full swing transition bit to first post-cursor. Default value set by RCW configuration. Recommended settings per protocol: PCIe: 0_1000 (2.5 Gbaud) SGMII: 0_0000 (1.25 Gbaud) SGMII: 0_0110 (3.125 Gbaud) SATA: 0_0010 (1.5 Gbaud) Others: 0_0110 Note: SATA 3 Gbaud and PCIe 5 Gbaud setting taken from LNaSSCR0[RATIO_PST1Q_0). Note: SATA 6 Gbaud setting taken from LNaSSCR1[RATIO_PST1Q_1). Note: this field is read-only when this lane is operating as PCIe
15-14 —	Reserved
13-8 ADPT_EQ	Transmitter Adjustments for 8G/10G Default value set by RCW configuration. Recommended settings per protocol: Others: 11_0000 Note: this field is read-only when this lane is operating as PCIe
7-6 —	Reserved
5-0 AMP_RED	Overall TX Amplitude Reduction Default value set by RCW configuration. Recommended settings per protocol: SGMII: 00_0110 (1.25 Gbaud) SGMII: 00_0000 (3.125 Gbaud) 1GKX: 000 SATA: 01_0000 (1.5 Gbaud) Others: 00_0000 Note: SATA 3 Gbaud and PCIe 5 Gbaud setting taken from LNaSSCR0[AMP_RED_0). Note: SATA 6 Gbaud setting taken from LNaSSCR1[AMP_RED_1) Note: this field is read-only when this lane is operating as PCIe

28.5.20 Speed Switch Control Register 1- Lane 0 (LNASSCR1 - LNDSSCR1)

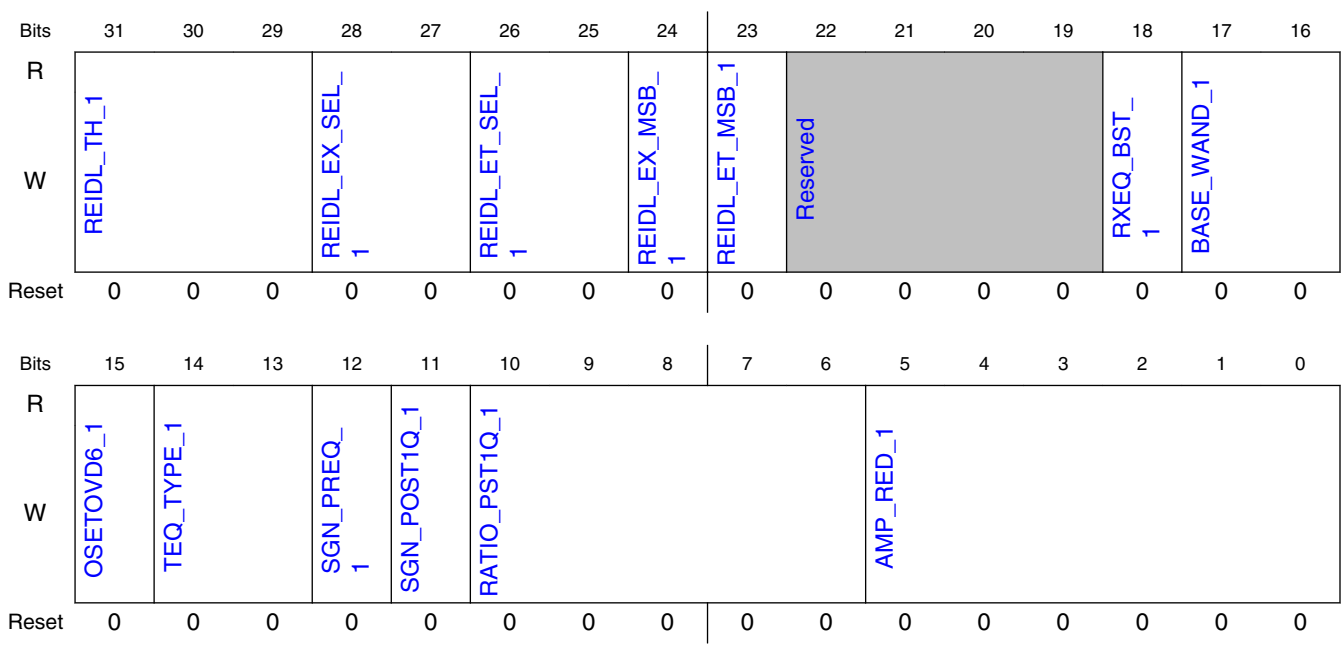
28.5.20.1 Offset

Register	Offset
LNASSCR1	81Ch
LNBSSCR1	85Ch
LNDSSCR1	8DCh

28.5.20.2 Function

LNaSSCR1 contains control bits for modifying SATA 6 Gbaud behavior of SerDes lane a.

28.5.20.3 Diagram



28.5.20.4 Fields

Field	Function
31-29 REIDL_TH_1	Receiver electrical idle detection threshold control Default set by RCW configuration. Recommended settings per protocol: SATA: 000 (6 Gbaud) Others: 000 For PCIe 2.5 Gbaud and SATA 1.5 Gbaud and protocols other than PCIe or SATA, see LNaGCR1[REIDL_TH]. For PCIe 5 Gbaud and SATA 3 Gbaud, see LNaSSCR0[REIDL_TH_0].
28-27 REIDL_EX_SEL_1	Exit electrical idle filter select = REIDL_EX_MSB_1||REIDL_EX_SEL_1 Default set by RCW configuration. Recommended settings per protocol: SATA: 000 (6 Gbaud) Others: 000 For PCIe 2.5 Gbaud and SATA 1.5 Gbaud and protocols other than PCIe or SATA, see LNaGCR1[REIDL_EX_SEL]. For PCIe 5 Gbaud and SATA 3 Gbaud, see LNaSSCR0[REIDL_EX_SEL_0].
26-25 REIDL_ET_SEL_1	Enter idle filter select = REIDL_ET_MSB_1||REIDL_ET_SEL_1: Default set by RCW configuration. Recommended setting per protocol: SATA: 000 (6 Gbaud) Others: 000 For PCIe 2.5 Gbaud and SATA 1.5 Gbaud and protocols other than PCIe or SATA, see LNaGCR1[REIDL_ET_SEL]. For PCIe 5 Gbaud and SATA 3 Gbaud, see LNaSSCR0[REIDL_ET_SEL_0].
24 REIDL_EX_MSB_1	Exit idle filter select MSB. See REIDL_EX_SEL_1 for settings. Default set by RCW configuration. For PCIe 2.5 Gbaud and SATA 1.5 Gbaud and protocols other than PCIe or SATA, see LNaGCR1[REIDL_EX_MSB]. For PCIe 5 Gbaud and SATA 3 Gbaud, see LNaSSCR0[REIDL_EX_MSB_0].
23 REIDL_ET_MSB_1	Enter idle filter select MSB. See REIDL_ET_SEL_1 for settings. For PCIe 2.5 Gbaud and SATA 1.5 Gbaud and protocols other than PCIe or SATA, see LNaGCR1[REIDL_ET_MSB]. For PCIe 5 Gbaud and SATA 3 Gbaud, see LNaSSCR0[REIDL_ET_MSB_0].
22-19 —	Reserved
18 RXEQ_BST_1	Rx Equalization Boost Default value set by RCW configuration Recommended value per protocol: 0

Table continues on the next page...

Field	Function
	For PCIe 2.5 Gbaud and SATA 1.5 Gbaud settings, see LNaRECR0[RXEQ_BST]. For PCIe 5 Gbaud and SATA 3 Gbaud settings, see LNaSSCR0[RXEQ_BST_0].
17-16 BASE_WAND_1	Baseline Wander Control Select Default value set by RCW config Recommended setting per protocol: SATA: 00 (6 Gbaud) Others: 00 For PCIe 2.5 Gbaud and SATA 1.5 Gbaud settings, see LNaRECR0[BASE_WAND]. For PCIe 5 Gbaud and SATA 3 Gbaud settings, see LNaSSCR0[BASE_WAND_0]. 00b - off (8b10b data) 01b - default BinBLW threshold 10b - alternate BinBLW sign 11b - Use OSETOVD[4:0] as GainBLW override
15 OSETOVD6_1	Binary Decode of Lane Adaptive Equalization offset intitialization or override value. [6] = 1: Double Imposed Offset Default value set by RCW configuration Recommended setting per protocol: SATA: 0 (6 Gbaud) Others: 0 For PCIe 2.5 Gbaud and SATA 1.5 Gbaud and protocols other than PCIe or SATA, see LNaRECR0[OSETOVD]. For PCIe 5 Gbaud and SATA 3 Gbaud, see LNaSSCR0[OSETOVD6_0].
14-13 TEQ_TYPE_1	Lane transmit equalization. Default value set by RCW configuration Recommended setting per protocol: SATA: 01 (6 Gbaud) Others: 00 For PCIe 2.5 Gbaud and SATA 1.5 Gbaud and protocols other than PCIe or SATA, see LNaTECR0[TEQ_TYPE]. For PCIe 5 Gbaud and SATA 3 Gbaud, see LNaSSCR0[TEQ_TYPE_0]. 00b - No TX Equalization 01b - 2 Levels of TX Equalization (+1 post cursor) 10b - 3 Levels of TX Equalization (+1 pre-cursor and +1 post-cursor) 11b - Reserved
12 SGN_PREQ_1	Precursor sign Default value set by RCW configuration Recommended setting per protocol: SATA: 0 (6 Gbaud) Others: 0 For PCIe 2.5 Gbaud and SATA 1.5 Gbaud and protocols other than PCIe or SATA, see LNaTECR0[SGN_PREQ]. For PCIe 5 Gbaud and SATA 3 Gbaud, see LNaSSCR0[SGN_PREQ_0].

Table continues on the next page...

Field	Function
	0b - Negative Sign (close eye) 1b - Positive Sign (open eye)
11 SGN_POST1Q_1	Post1q sign Default value set by RCW configuration Recommended setting per protocol: SATA: 1 (6 Gbaud) Others: 0 For PCIe 2.5 Gbaud and SATA 1.5 Gbaud and protocols other than PCIe or SATA, see LNaTECR0[SGN_POST1Q]. For PCIe 5 Gbaud and SATA 3 Gbaud, see LNaSSCR0[SGN_POST1Q_0]. 0b - Negative Sign (close eye) 1b - Positive Sign (open eye)
10-6 RATIO_PST1Q_1	Ratio of full swing transition bit to first post-cursor. Default value set by RCW configuration Recommended setting per protocol: SATA: 0_0010 (6 Gbaud) Others: 0_0000 For PCIe 2.5 Gbaud and SATA 1.5 Gbaud and protocols other than PCIe or SATA, see LNaTECR0[AMP_RED]. For PCIe 5 Gbaud and SATA 3 Gbaud, see LNaSSCR0[AMP_RED_0]. Note: this field is read-only when this lane is operating as PCIe
5-0 AMP_RED_1	Overall TX Amplitude Reduction Default value set by RCW configuration Recommended setting per protocol: SATA: 00_0000 (6 Gbaud) Others: 00_0000 For PCIe 2.5 Gbaud and SATA 1.5 Gbaud and protocols other than PCIe or SATA, see LNaTECR0[AMP_RED]. For PCIe 5 Gbaud and SATA 3 Gbaud, see LNaSSCR0[AMP_RED_0]. Note: this field is read-only when this lane is operating as PCIe

28.5.21 TTL Control Register 0 - Lane a (LNATTLCR0 - LNDTTLCR0)

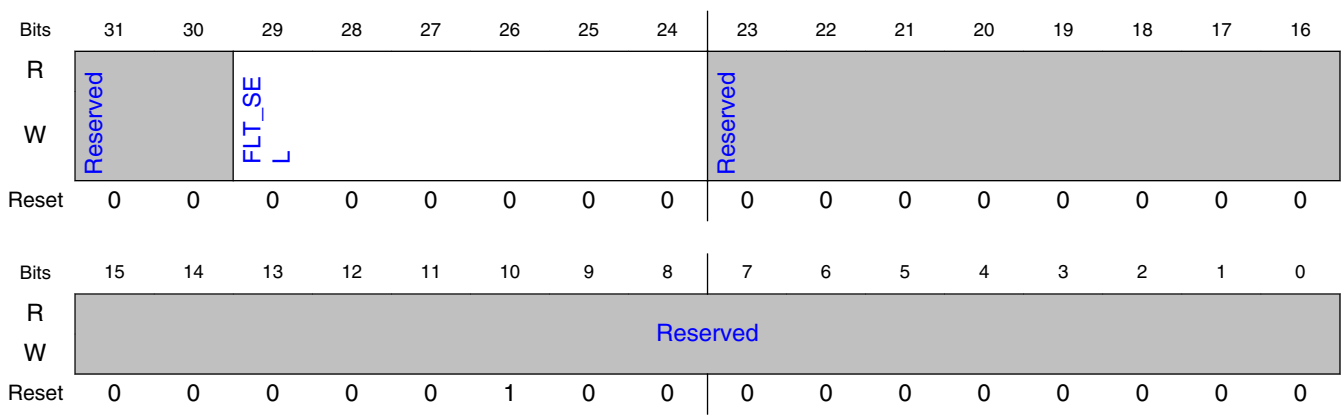
28.5.21.1 Offset

Register	Offset
LNATTLCR0	820h
LNBTTLCR0	860h
LNDTTLCR0	8E0h

28.5.21.2 Function

LNmTTLCR0 contains control bits for the Transition Tracking Loop (TTL) on SerDes lane a.

28.5.21.3 Diagram



28.5.21.4 Fields

Field	Function
31-30 —	Reserved
29-24 FLT_SEL	Selects the gain 'Kfr', 'Kph' and TTL Edge Counting Window Widths in the CDR Loop for the Lane. Default value set by RCW configuration. Recommended values per protocol: SGMII: 11_1001 (1.25 Gbaud) SGMII: 00_0000 (3.125 Gbaud)

Table continues on the next page...

Field	Function
	Others: 00_0000
23-0 —	Reserved

28.5.22 Test Control/Status Register 3 - Lane a (LNATCSR3 - LNDTCSR3)

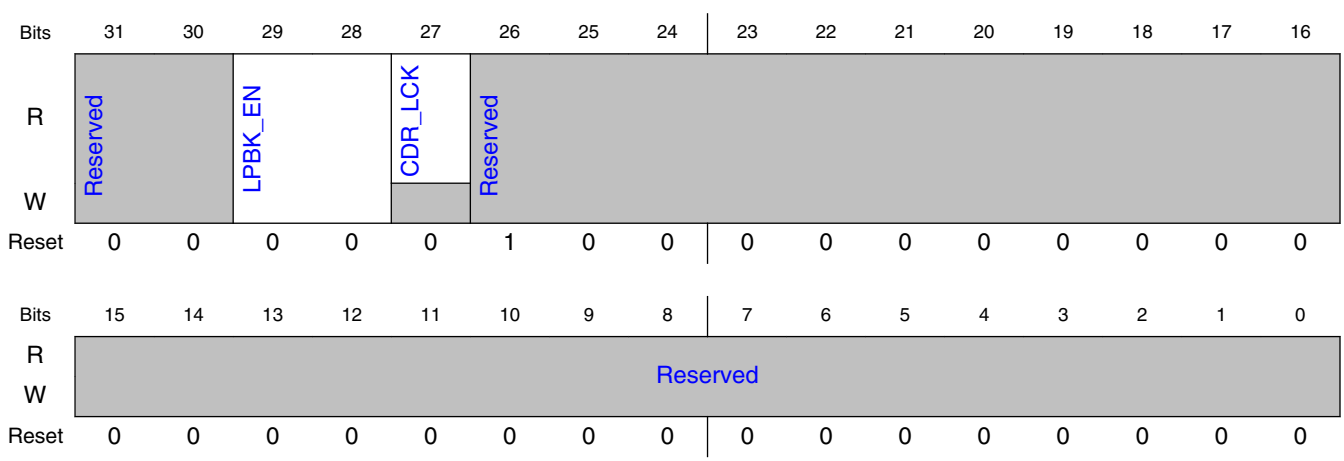
28.5.22.1 Offset

Register	Offset
LNATCSR3	83Ch
LNBTCSR3	87Ch
LNDTCSR3	8FCh

28.5.22.2 Function

LNmTCSR3 contains test control and status bits.

28.5.22.3 Diagram



28.5.22.4 Fields

Field	Function
31-30 —	Reserved
29-28 LPBK_EN	Loopback data from TX to RX All others reserved Note: Loopback using PCI-Express requires Root Complex configuration. PCI Express End point loopback is not supported. 00b - Application Mode 01b - Loopback Mode
27 CDR_LCK	When asserted, CDR loop has acquired a valid Rx clock
26-0 —	Reserved

28.5.23 PEXA Protocol Control Register 0 (PEXACR0)

28.5.23.1 Offset

Register	Offset
PEXACR0	1000h

28.5.23.2 Function

The PCIe Protocol Control and Status Registers support the control and status bits related to the PCIe PCS layers and PHY control.

PEXnCR0 contains control bits used for the PEXn protocol.

28.5.23.3 Diagram

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

Bits	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

28.5.23.4 Fields

Field	Function
31-28 —	Reserved
27 RD_SW	Reserved
26-0 —	Reserved

28.5.24 SGMIIa Protocol Control Register 1 (SGMIIACR1 - SGMIIBCR1)

28.5.24.1 Offset

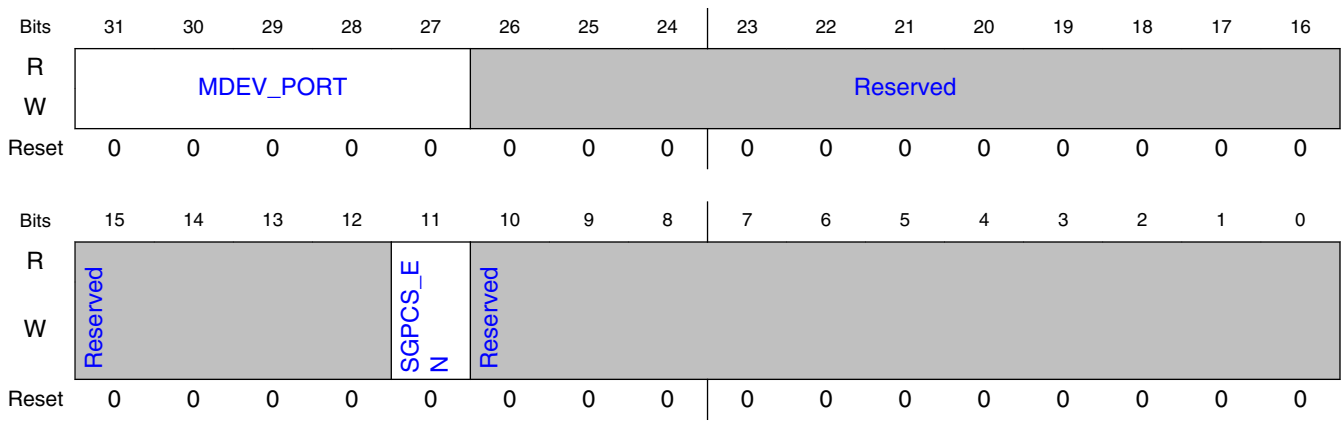
For a = A to B (0 to 1):

Register	Offset
SGMIIaCR1	1804h + (a × 10h)

28.5.24.2 Function

SGMIInCR1 contains control bits used for the SGMIIn protocol.

28.5.24.3 Diagram



28.5.24.4 Fields

Field	Function
31-27 MDEV_PORT	MDIO bus port address. • When accessing using Clause 22, this field is compared to MDIO_CTL[PHY_ADDR] to accept a command. • When accessing SGMII using clause 45, the MDIO Ethernet Management Interface register MDIO_CTL[DEV_ADDR], must be 03h and this field is compared to MDIO_CTL[PORT_ADDR] to accept a command. The register address (starting at 8000h) is specified in MDIO_ADDR. Default value: 0 Note: software must wait at least 3 platform clocks after changing this value before performing any MDIO accesses to the SGMIIa PCS.
26-12 —	Reserved
11 SGPCS_EN	SGMII PCS enable Recommended value: 1 if SGMIIa is enabled, else 0 Default value is set by RCW Note: must be set to 1 as part of econfiguring a port from XAUI or XFI to SGMII or 1000Base-KX operation
10-0 —	Reserved

28.5.25 SGMIIa Protocol Control Register 3 (SGMIIACR3 - SGMIIBCR3)

28.5.25.1 Offset

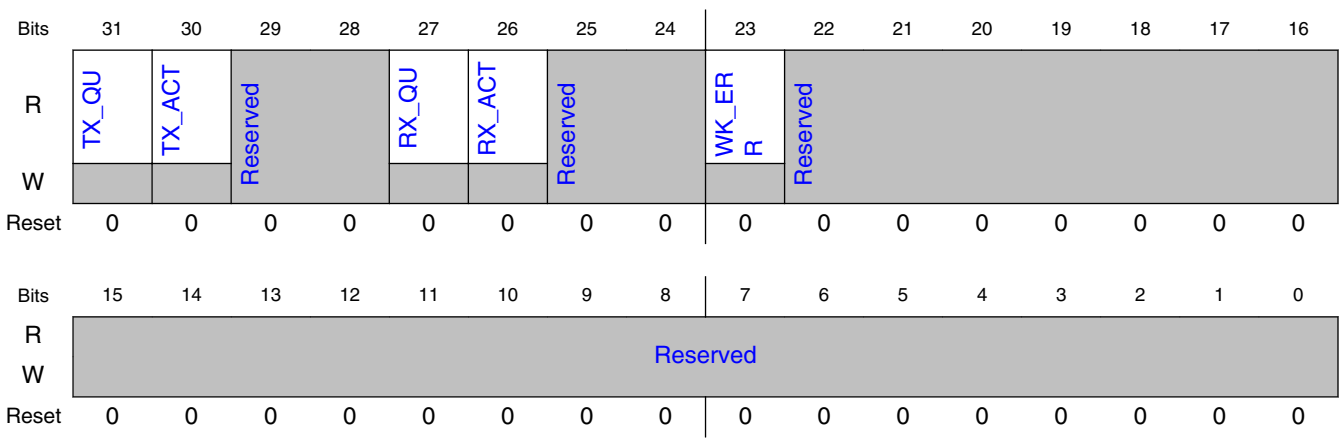
For a = A to B (0 to 1):

Register	Offset
SGMIIaCR3	180Ch + (a × 10h)

28.5.25.2 Function

SGMIInCR3 contains control bits used for the SGMIIn protocol.

28.5.25.3 Diagram



28.5.25.4 Fields

Field	Function
31 TX_QU	Tx quiet LPI state (read-only). Tx is in electrical idle.
30	Tx active LPI state (read-only)

Table continues on the next page...

Field	Function
TX_ACT	
29-28 —	reserved
27 RX_QU	Rx quiet LPI state (read-only). Rx is in electrical idle
26 RX_ACT	Rx active LPI state (read-only)
25-24 —	reserved
23 WK_ERR	Error on LPI wake (read-only)
22-0 —	Reserved

28.6 MDIO register spaces

The SerDes module also contains MDIO slave ports containing registers for each PCS module.

The PCS MDIO registers are accessed through an indirect method from the MDIO Ethernet management interface registers in the Ethernet MAC controllers. There are two methods used to access the PCS MDIO registers—dubbed Clause 45 and Clause 22 (from the IEEE 802.3 sub clauses that define them), depending on the protocol.

The Clause 45 protocols are subdivided into separate spaces selected by the device address. The following table shows the Clause 45 register spaces:

Table 28-5. Clause 45 register spaces

1000Base-KX MDIO registers			
Register space	**Device address**	**Notes**	**See**
1000Base-KX PCS registers	03h	Includes vendor-specific alias of Clause 22 SGMII registers starting at register address 8000h	MDIO_KX_PCS register descriptions
1000Base-KX Auto-Negotiation registers	07h		MDIO_KX_AN register descriptions
1000Base-KX Vendor-Specific 1 registers	1Dh		MDIO_KX_VENDOR_SPEC register descriptions

The following table shows the Clause 22 register spaces:

Table 28-6. Clause 22 registers

Registers	Notes	See
SGMII		MDIO_SGMII register descriptions

NOTE

Throughout the MDIO register sections:

- Each MDIO register is 16 bits.
- Register offsets are 16-bit offsets, not byte offsets.
- Addresses not listed are reserved.
- Read accesses to reserved addresses return 0x0000.

28.6.1 MDIO_KX_PCS register descriptions

The 1000Base-KX PCS register space is selected when the associated SGMIInCR1[MDEV_PORT] matches the Ethernet MAC port address (MDIO_CTL[PORT_ADDR]) and the device address (MDIO_CTL[DEV_ADDR]) is 03h.

This register space is also used for (vendor-specific) Clause 45 access to SGMII register aliases (starting at register address 8000h).

28.6.1.1 MDIO_KX_PCS Memory map

MDIO_KX_PCS base address: 0h

Offset	Register	Width (In bits)	Access	Reset value
0h	KX PCS Control (KX_PCS_CR)	16	RW	1140h
1h	KX PCS Status (KX_PCS_SR)	16	RO	0009h
2h	KX PCS Device Identifier Upper (KX_PCS_DEV_ID)	16	RO	0083h
3h	KX PCS Device Identifier Lower (KX_PCS_DEV_ID_L)	16	RO	E400h
5h	KX PCS Devices In Package 0 (KX_PCS_DEV_PRES0)	16	RO	0000h
6h	KX PCS Devices In Package 1 (KX_PCS_DEV_PRES1)	16	RO	0088h
Eh	KX PCS Package Identifier Upper (KX_PCS_PKG_ID_U)	16	RO	0083h
Fh	KX PCS Package Identifier Lower (KX_PCS_PKG_ID_L)	16	RO	E400h
8000h	SGMII Control (C45_SGMII_CR)	16	RW	1140h
8001h	SGMII Status (C45_SGMII_SR)	16	RO	0009h

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
8002h	SGMII PHY Identifier Upper (C45_SGMII_PHY_ID_H)	16	RO	0083h
8003h	SGMII PHY Identifier Lower (C45_SGMII_PHY_ID_L)	16	RO	E400h
8004h	SGMII Device Ability for 1000Base-X (C45_SGMII_DEV_ABIL_1 KBX)	16	RW	01A0h
8004h	SGMII Device Ability for SGMII (C45_SGMII_DEV_ABIL_SGMII)	16	RW	01A0h
8005h	SGMII Partner Ability for 1000Base-X (C45_SGMII_LP_DEV_ABIL_ 1KBX)	16	RO	0000h
8005h	SGMII Partner Ability for SGMII (C45_SGMII_LP_DEV_ABIL_SGMII)	16	RO	0000h
8006h	SGMII AN Expansion (C45_SGMII_AN_EXP)	16	RO	0004h
8007h	SGMII Next Page Transmit (C45_SGMII_NP_TX)	16	RW	0000h
8008h	SGMII LP Next Page Receive (C45_SGMII_NP_RX)	16	RO	0000h
800Fh	SGMII Extended Status (C45_SGMII_XTND_STAT)	16	RU	0000h
8010h	SGMII Scratch (C45_SGMII_SCRATCH)	16	RW	0000h
8011h	SGMII Design Revision (C45_SGMII_REV)	16	RO	0001h
8012h	SGMII Link Timer Lower (C45_SGMII_LINK_TMR_L)	16	RW	12D0h
8013h	SGMII Link Timer Upper (C45_SGMII_LINK_TMR_H)	16	RW	0013h
8014h	SGMII IF Mode (C45_SGMII_IF_MODE)	16	RW	0000h

28.6.1.2 KX PCS Control (KX_PCS_CR)

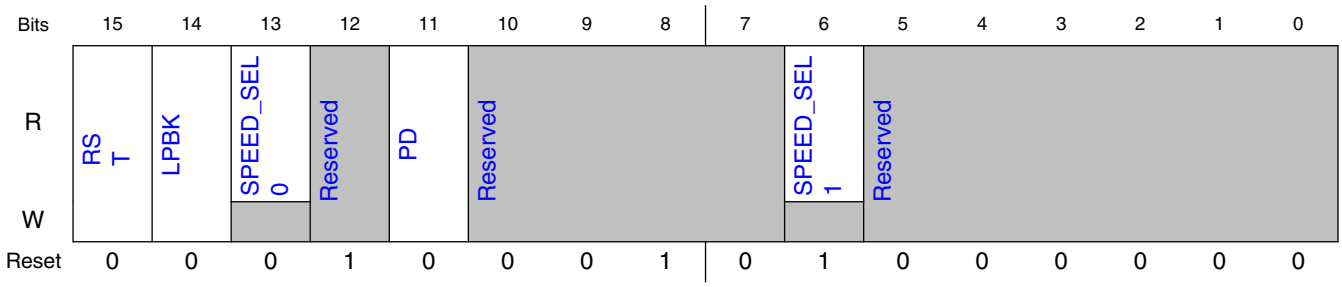
28.6.1.2.1 Offset

Register	Offset
KX_PCS_CR	0h

28.6.1.2.2 Function

The 1000Base-KX PCS Control Register contains control bits used to enable/disable functions in the PCS, and to initiate commands such as reset.

28.6.1.2.3 Diagram



28.6.1.2.4 Fields

Field	Function
15 RST	Reset Self-Clearing Read / Write Reset Command Register. When set to 1, a synchronous reset pulse is generated which resets all the PCS state machines, the Comma detection function and the 8b/10b coder / decoder. Should be set to 0 (default) for normal operation.
14 LPBK	Loopback Loopback Command. 0b - Normal operation 1b - Reserved
13 SPEED_SEL0	Speed selection (LSB) MSB,LSB 1,0: 1000 Mb/s All others reserved Note: SGMII speed is controlled with the IF Mode register
12 —	Reserved
11 PD	Power down 0b - Normal operation 1b - Power down PCS. The PCS is held in internal soft reset until the bit is cleared again.
10-7 —	Reserved
6 SPEED_SEL1	Speed selection (MSB) See SPEED_SEL0
5-0 —	Reserved

28.6.1.3 KX PCS Status (KX_PCS_SR)

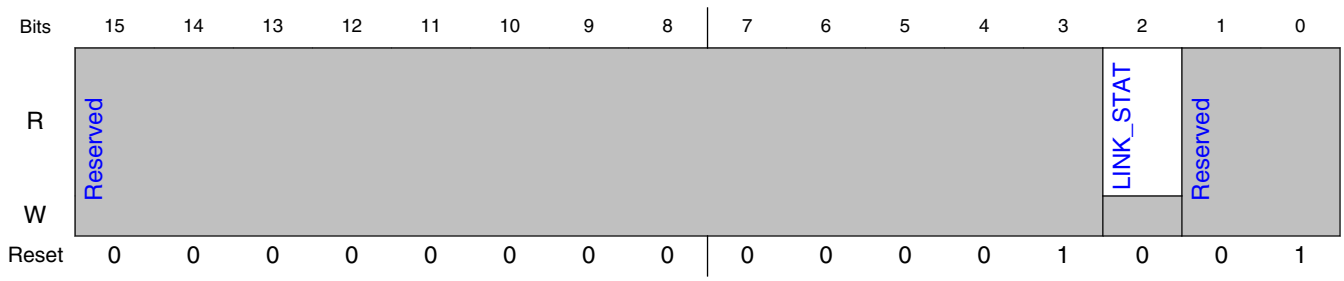
28.6.1.3.1 Offset

Register	Offset
KX_PCS_SR	1h

28.6.1.3.2 Function

The 1000Base-KX PCS Status Register contains status bits on the operation of the PCS.

28.6.1.3.3 Diagram



28.6.1.3.4 Fields

Field	Function
15-3 —	Reserved
2 LINK_STAT	Link status Link Status 0b - The link is not valid. If the link synchronization is lost a 0 is latched which is cleared only after a register read access 1b - This link is valid.
1-0 —	Reserved

28.6.1.4 KX PCS Device Identifier Upper (KX_PCS_DEV_ID)

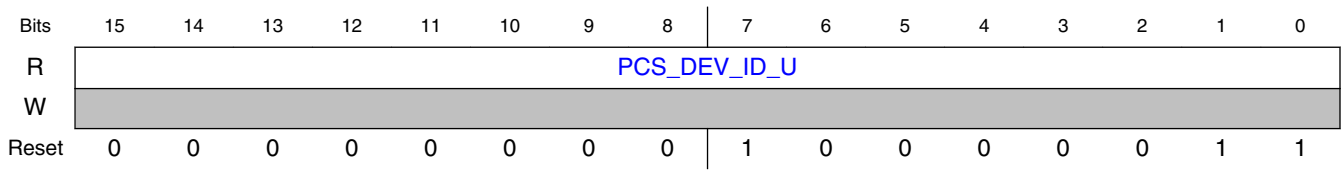
28.6.1.4.1 Offset

Register	Offset
KX_PCS_DEV_ID	2h

28.6.1.4.2 Function

The 1000Base-KX PCS Device Identifier Upper Register contains the upper half of the 32-bit Device Identifier.

28.6.1.4.3 Diagram



28.6.1.4.4 Fields

Field	Function
15-0 PCS_DEV_ID_U	PCS Device Identifier Upper PCS Device Identifier Upper: OUI[3:18]

28.6.1.5 KX PCS Device Identifier Lower (KX_PCS_DEV_ID_L)

28.6.1.5.1 Offset

Register	Offset
KX_PCS_DEV_ID_L	3h

28.6.1.5.2 Function

The 1000Base-KX PCS Identifier Lower Register contains the lower half of the 32-bit Device Identifier.

28.6.1.5.3 Diagram

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

28.6.1.5.4 Fields

Field	Function
15-0 PCS_DEV_ID_L	PCS Device Identifier Lower PCS Device Identifier Lower: 15:10 OUI[19:24] 9:4 Manufacturer's Model Number 3:0 Revision Number

28.6.1.6 KX PCS Devices In Package 0 (KX_PCS_DEV_PRES0)

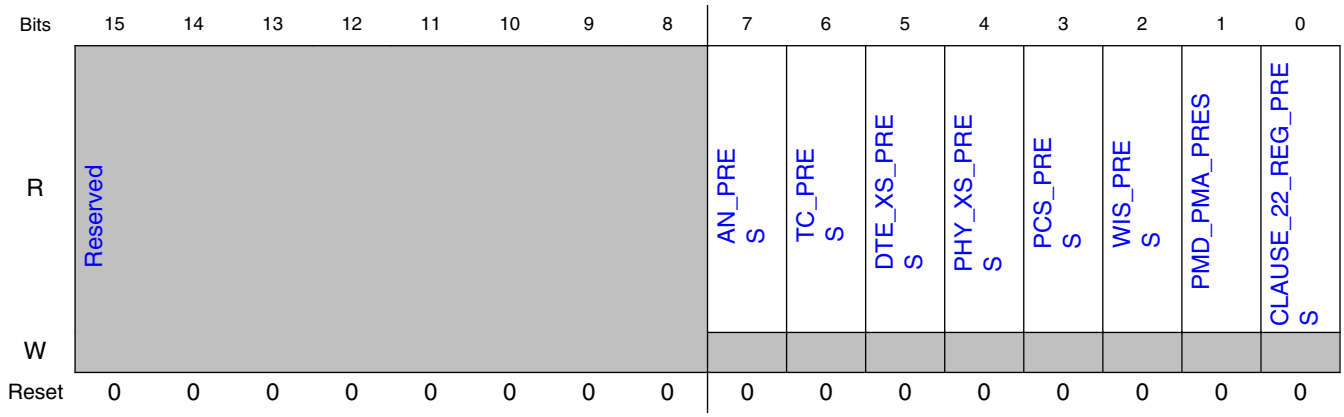
28.6.1.6.1 Offset

Register	Offset
KX_PCS_DEV_PRES0	5h

28.6.1.6.2 Function

The 1000Base-KX PCS Devices in Package 0 Register contains half of the PCS devices in package status.

28.6.1.6.3 Diagram



28.6.1.6.4 Fields

Field	Function
15-8 —	Reserved
7 AN_PRES	AN present 0b - Auto-Negotiation not present in package 1b - Auto-Negotiation present in package
6 TC_PRES	TC present 0b - TC not present in package 1b - TC present in package
5 DTE_XS_PRES	DTE XS present 0b - DTE XS not present in package 1b - DTE XS present in package
4 PHY_XS_PRES	PHY XS present 0b - PHY XS not present in package 1b - PHY XS present in package
3 PCS_PRES	PCS present 0b - PCS not present in package 1b - PCS present in package
2 WIS_PRES	WIS present 0b - WIS not present in package 1b - WIS present in package
1 PMD_PMA_PRES	PMD/PMA present 0b - PMA/PMD not present in package 1b - PMA/PMD present in package
0 CLAUSE_22_REG_PRES	Clause 22 registers present 0b - Clause 22 registers not present in package 1b - Clause 22 registers present in package

28.6.1.7 KX PCS Devices In Package 1 (KX_PCS_DEV_PRES1)

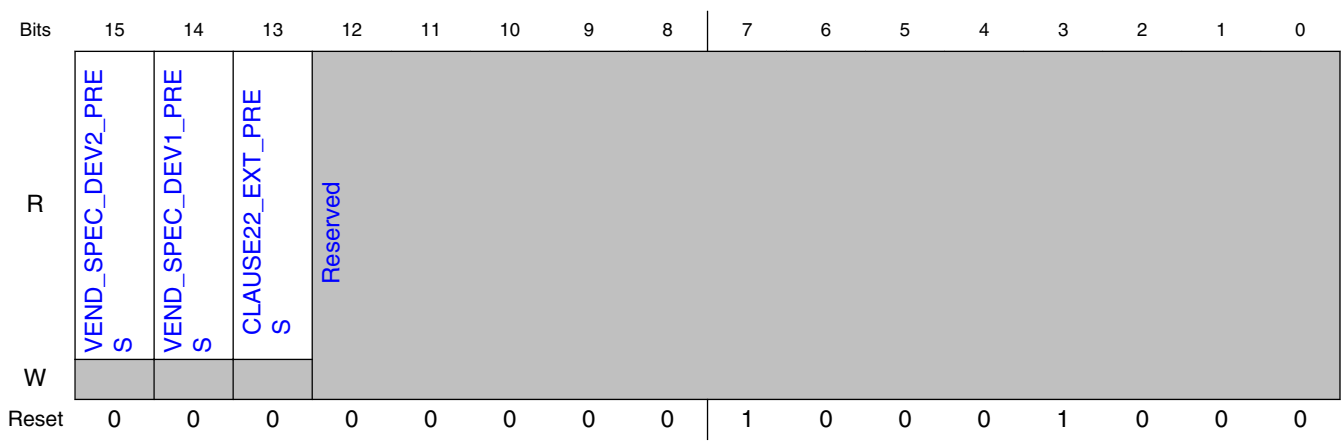
28.6.1.7.1 Offset

Register	Offset
KX_PCS_DEV_PRES1	6h

28.6.1.7.2 Function

The 1000Base-KX PCS Devices in Package 1 Register contains half of the PCS devices in package status.

28.6.1.7.3 Diagram



28.6.1.7.4 Fields

Field	Function
15 VEND_SPEC_DEV2_PRES	Vendor specific device 2 present 0b - Vendor specific device 2 not present in package 1b - Vendor specific device 2 present in package
14 VEND_SPEC_DEV1_PRES	Vendor specific device 1 present 0b - Vendor specific device 1 not present in package 1b - Vendor specific device 1 present in package
13 CLAUSE22_EXT_PRES	Clause 22 extension present 0b - Clause 22 extension not present in package 1b - Clause 22 extension present in package
12-0 —	Reserved

28.6.1.8 KX PCS Package Identifier Upper (KX_PCS_PKG_ID_U)

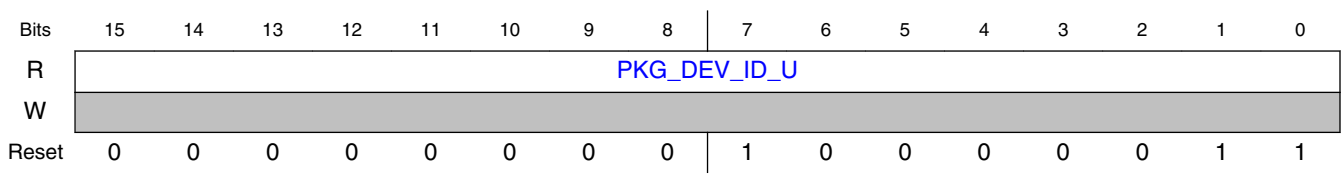
28.6.1.8.1 Offset

Register	Offset
KX_PCS_PKG_ID_U	Eh

28.6.1.8.2 Function

The 1000Base-KX PCS Package Device Identifier Upper Register contains the upper half of the 32-bit Package Identifier.

28.6.1.8.3 Diagram



28.6.1.8.4 Fields

Field	Function
15-0 PKG_DEV_ID_U	Package Device Identifier Upper Package Device Identifier Upper: OUI[3:18]

28.6.1.9 KX PCS Package Identifier Lower (KX_PCS_PKG_ID_L)

28.6.1.9.1 Offset

Register	Offset
KX_PCS_PKG_ID_L	Fh

28.6.1.9.2 Function

The KX PCS Package Identifier Lower Register contains the lower half of the 32-bit Package Identifier.

28.6.1.9.3 Diagram

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

28.6.1.9.4 Fields

Field	Function
15-0 PKG_DEV_ID_L	Package Device Identifier Lower Package Device Identifier Lower: 15:10 OUI[19:24] 9:4 Manufacturer's Model Number 3:0 Revision Number

28.6.1.10 SGMII Control (C45_SGMII_CR)

28.6.1.10.1 Offset

Register	Offset
C45_SGMII_CR	8000h

28.6.1.10.2 Function

The SGMII Control Register contains control bits used to enable/disable functions in the PCS, and to initiate commands such as reset.

28.6.1.10.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RST	LPBK	SPEED_SEL 0	AN_EN	PD	ISOLATE	RESTART_AN	FD	COL_TEST	SPEED_SEL 1	Reserved					
Reset	0	0	0	1	0	0	0	1	0	1	0	0	0	0	0	0

28.6.1.10.4 Fields

Field	Function
15 RST	Reset Self-Clearing Read / Write Reset Command Register. When set to 1, a synchronous reset pulse is generated which resets all the PCS state machines, the Comma detection function and the 8b/10b coder / decoder. Should be set to 0 (default) for normal operation.
14 LPBK	Loopback Loopback Command. 0b - Normal operation 1b - Reserved
13 SPEED_SEL0	Speed selection (LSB) Speed selection (LSB) All others reserved Note: SGMII speed is controlled with the IF Mode register MSB,LSB 1,0: 1000 Mb/s All others reserved
12 AN_EN	AN enable Auto-negotiation enable 0b - Disable auto-negotiation 1b - Enable auto-negotiation
11 PD	Power down Power down 0b - Normal operation 1b - Power down PCS. The PCS is held in internal soft reset until the bit is cleared again.
10 ISOLATE	Isolate Isolate 0b - Normal operation 1b - Reserved
9 RESTART_AN	Restart AN Restart auto-negotiation

Table continues on the next page...

Field	Function
	This bit is self-clearing 0b - Normal operation 1b - Restart auto-negotiation
8 FD	Full duplex Duplex mode. Read-only 0b - Reserved 1b - Full duplex
7 COL_TEST	Collision test Collision test. Read-only 0b - Disable COL signal test 1b - Reserved
6 SPEED_SEL1	Speed selection (MSB) Speed selection (MSB) See SPEED_SEL0
5-0 —	Reserved

28.6.1.11 SGMII Status (C45_SGMII_SR)

28.6.1.11.1 Offset

Register	Offset
C45_SGMII_SR	8001h

28.6.1.11.2 Function

The SGMII Status Register contains status bits on the operation of the PCS.

28.6.1.11.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ST_100BASET4	FD_100BASEX	HD_100BASEX	FD_10MBPS	HD_10MBPS	FD_100BASET2	HD_100BASET2	EXT_STAT	Reserved		AN_COMP	REM_FAULT	AN_ABIL	LINK_STAT	JAB_DET	EXT_CAP
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1

28.6.1.11.4 Fields

Field	Function
15 ST_100BASET4	100Base-T4 Read Only bit set to 0 to indicate that the PCS does not support 100Base-T4 operation
14 FD_100BASEX	100Base-X Full Duplex Read Only bit set to 0 to indicate that the PCS does not support 100Base-X operation
13 HD_100BASEX	100Base-X Half Duplex Read Only bit set to 0 to indicate that the PCS does not support 100Base-X operation
12 FD_10MBPS	10Mbps Full Duplex Read Only bit set to 0 to indicate that the PCS does not support 10Mbps operation
11 HD_10MBPS	10Mbps Half Duplex Read Only bit set to 0 to indicate that the PCS does not support 10Mbps operation
10 FD_100BASET2	100BaseT2 Full Duplex Read Only bit set to 0 to indicate that the PCS does not support 100Base-T2 operation.
9 HD_100BASET2	100BaseT2 Half Duplex Read Only bit set to 0 to indicate that the PCS does not support 100Base-T2 operation.
8 EXT_STAT	Extended status Read Only bit always set to 0 to indicate that the PCS does not implement an extended status register.
7-6 —	Reserved
5 AN_COMP	AN complete Auto-negotiation complete. Read Only Bit 0b - The Auto Negotiation process is not completed or Auto Negotiation is disabled 1b - The Auto Negotiation process is completed and that the Auto Negotiation control registers are valid.
4 REM_FAULT	Remote fault Read Only Bit always set to 0. The PCS does not implement a PHY specific remote fault detection optional function.
3 AN_ABIL	AN ability Auto Negotiation Ability. Read Only Bit set to „1. to indicate that the PCS supports Auto-Negotiation.
2 LINK_STAT	Link status Link Status 0b - The link is not valid. If the link synchronization is lost a 0 is latched which is cleared only after a register read access 1b - This link is valid.
1 JAB_DET	Jabber detection Read Only bit always set to 0, the Core does not support the optional Jabber detection function
0 EXT_CAP	Extended capability Read Only bit set to 1. to indicate that the Core supports extended registers

28.6.1.12 SGMII PHY Identifier Upper (C45_SGMII_PHY_ID_H)

28.6.1.12.1 Offset

Register	Offset
C45_SGMII_PHY_ID_H	8002h

28.6.1.12.2 Function

The SGMII PHY Identifier Upper Register contains the upper half of the 32-bit PHY Identifier.

28.6.1.12.3 Diagram

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

28.6.1.12.4 Fields

Field	Function
15-0 PHY_IDENTIFIER_U	PHY Identifier Upper PHY Identifier Upper: OUI[3:18]

28.6.1.13 SGMII PHY Identifier Lower (C45_SGMII_PHY_ID_L)

28.6.1.13.1 Offset

Register	Offset
C45_SGMII_PHY_ID_L	8003h

28.6.1.13.2 Function

The SGMII PHY Identifier Lower Register contains the lower half of the 32-bit PHY Identifier.

28.6.1.13.3 Diagram

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

28.6.1.13.4 Fields

Field	Function
15-0 PHY_IDENTIFIER_L	PHY Identifier Lower PHY Identifier Lower: 15:10 OUI[19:24] 9:4 Manufacturer's Model Number 3:0 Revision Number

28.6.1.14 SGMII Device Ability for 1000Base-X (C45_SGMII_DEV_ABIL_1KBX)

28.6.1.14.1 Offset

Register	Offset
C45_SGMII_DEV_ABIL_1KBX	8004h

28.6.1.14.2 Function

The SGMII Device Ability Register contains the control bits used to advertise the device abilities to the Link Partner during auto-negotiation for 1000Base-X mode.

28.6.1.14.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	NP	ACK	RF2	RF1	Reserved			PS2	PS1	HD	FD	Reserved				
W																
Reset	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0

28.6.1.14.4 Fields

Field	Function
15 NP	Next page Next page support. Set to 1 to indicate next page can be transferred following base page exchange
14 ACK	Ack Read only bit set to 1 when device has received three consecutive matching ability values from the link partner.
13 RF2	Remote fault 2 Fault condition advertised by the device: RF1=0 / RF2=0: No error, link is OK (Reset condition). RF1=0 / RF2=1: Device advertises that it is Off Line. RF1=1 / RF2=0: Device advertises a link failure condition.. RF1=1 / RF2=1: Device advertises an Auto Negotiation error. Note: the PCS does not interpret or set these bits. It is up to the application to implement the fault functions as needed.
12 RF1	Remote Fault 1 See RF2
11-9 —	Reserved
8 PS2	Pause 2 Pause bits both set to 1 (Reset Value) to advertise that the Core supports pause on both transmit and receive. Can be set to any other value to advertise other flow control capabilities.
7 PS1	Pause 1 See PS2
6 HD	Half Duplex Half Duplex Enable. Read only bit set to 1 when the device advertises that it supports Half Duplex Mode of operation.
5 FD	Full Duplex Full Duplex Enable. Set to 1 when the device advertises that it supports Full Duplex Mode of operation
4-0 —	Reserved

28.6.1.15 SGMII Device Ability for SGMII (C45_SGMII_DEV_ABIL_SGMII)

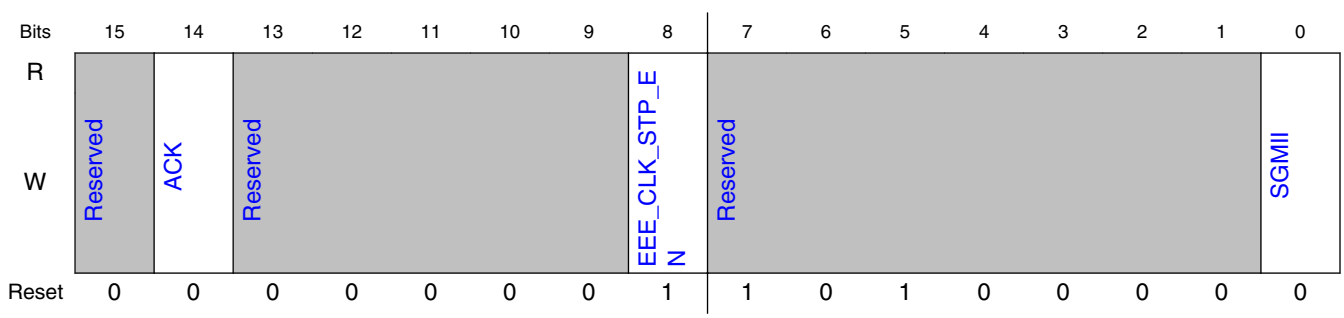
28.6.1.15.1 Offset

Register	Offset
C45_SGMII_DEV_ABIL_SGMII	8004h

28.6.1.15.2 Function

The SGMII Device Ability Register contains the control bits used to advertise the device abilities to the Link Partner during auto-negotiation for SGMII mode.

28.6.1.15.3 Diagram



28.6.1.15.4 Fields

Field	Function
15 —	Reserved
14 ACK	Ack Read only bit set to 1 when device has received three consecutive matching ability values from the link partner
13-9 —	Reserved. Should be set to 0
8 EEE_CLK_STP_EN	EEE Clock Stop Enable EEE Clock Stop Enable

Table continues on the next page...

Field	Function
	Should be set to 0 before SGMII Auto-Negotiation enable/restart, as this device does not support EEE clock stop. 0b - EEE Clock Stop Disabled 1b - EEE Clock Stop Enabled
7-1 —	Reserved. Should be set to 0
0 SGMII	SGMII SGMII mode. Should be set to 1.

28.6.1.16 SGMII Partner Ability for 1000Base-X (C45_SGMII_LP_DEV_ABIL_1KBX)

28.6.1.16.1 Offset

Register	Offset
C45_SGMII_LP_DEV_ABIL_1KBX	8005h

28.6.1.16.2 Function

The SGMII Partner Ability Register contains the bits advertised by the Link Partner during auto-negotiation for 1000Base-X mode.

28.6.1.16.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	NP	ACK	RF2	RF1	Reserved			PS2	PS1	HD	FD	Reserved				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

28.6.1.16.4 Fields

Field	Function
15 NP	Next Page Next page support.
14	Ack

Table continues on the next page...

Field	Function
ACK	Set to 1 when link partner has received three consecutive matching ability values from the device.
13 RF2	Remote Fault 2 Fault condition advertised by the link partner: RF1/RF2: 00: No error, link is OK (Reset condition). 01: Off Line. 10: Link failure 11: Auto Negotiation error.
12 RF1	Remote Fault 1 See RF2
11-9 —	Reserved
8 PS2	Pause 2 Pause capability of link partner (ASM_DIR:PAUSE) 0b00: No Pause 0b11: Symmetric Pause 0b10: Asymmetric Pause toward link partner 0b11: Both Symmetric Pause and Asymmetric Pause toward local device
7 PS1	Pause 1 See PS2
6 HD	Half Duplex Half Duplex support
5 FD	Full Duplex Full Duplex support
4-0 —	Reserved

28.6.1.17 SGMII Partner Ability for SGMII (C45_SGMII_LP_DEV_ABIL_SGMII)

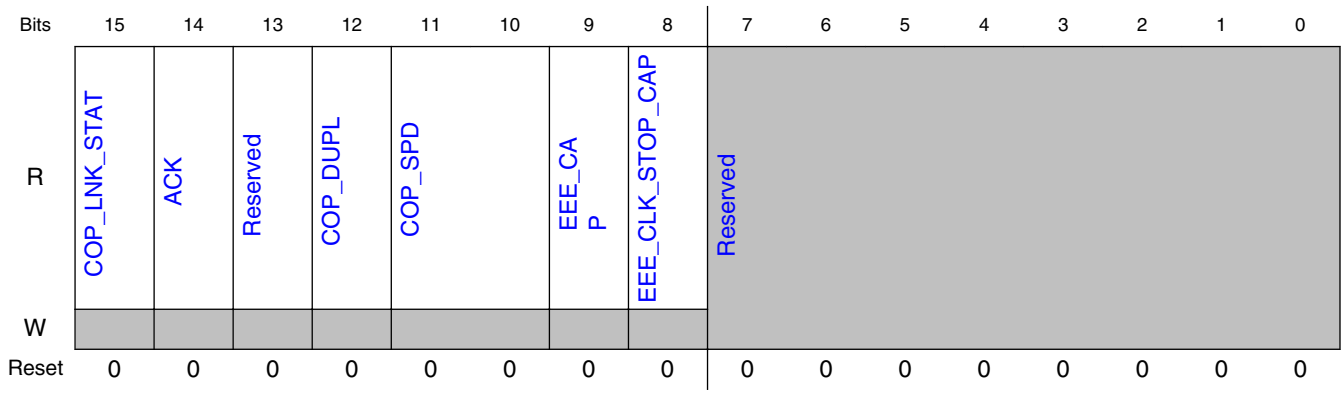
28.6.1.17.1 Offset

Register	Offset
C45_SGMII_LP_DEV_ABIL_SGMII	8005h

28.6.1.17.2 Function

The SGMII Partner Ability Register contains the capability status of the SGMII link partner.

28.6.1.17.3 Diagram



28.6.1.17.4 Fields

Field	Function
15 COP_LNK_STAT	Copper Link Status Read only bit, used by the SGMII PHY to advertise the Link Partner Copper status: 0b - Copper interface link is down 1b - Copper interface link is up
14 ACK	Ack Read only bit set to „1. when the Link Partner Copper Interface advertises that it has that received three consecutive matching ability values from the device
13 —	Always 0
12 COP_DUPL	Copper Duplex Read only bit, used by the SGMII PHY to advertise the Link Copper duplex capability: 0b - Copper Interface resolved to Half-Duplex 1b - Copper Interface resolved to Full-Duplex
11-10 COP_SPD	Copper speed Read only bits, used by the SGMII PHY to advertise the Link Partner Copper interface speed 00b - Copper Interface Speed is 10Mbps 01b - Copper Interface Speed is 100Mbps 10b - Copper Interface Speed is Gigabit 11b - Reserved
9 EEE_CAP	EEE Capability EEE Capability

Table continues on the next page...

Field	Function
	0b - EEE not supported 1b - EEE supported
8 EEE_CLK_STOP_CAP	EEE Clock Stop Capability EEE Clock Stop Capability 0b - EEE Clock Stop not supported 1b - EEE Clock Stop supported
7-0 —	Reserved

28.6.1.18 SGMII AN Expansion (C45_SGMII_AN_EXP)

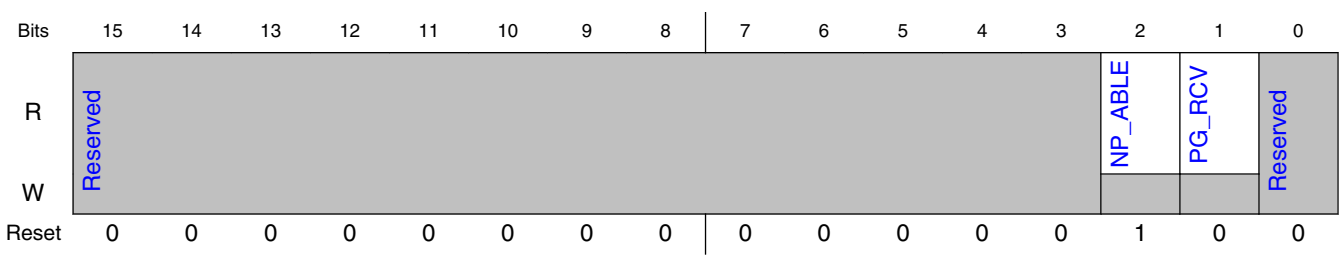
28.6.1.18.1 Offset

Register	Offset
C45_SGMII_AN_EXP	8006h

28.6.1.18.2 Function

The SGMII AN Expansion Register contains status bits indicating the PCS next page auto-negotiation capability and status.

28.6.1.18.3 Diagram



28.6.1.18.4 Fields

Field	Function
15-3 —	Reserved

Table continues on the next page...

Field	Function
2 NP_ABLE	Next Page Able Read Only bit set to 1 to indicate the PCS does support the Next Page function
1 PG_RCV	Page Received Set to 1 to indicate that a new page has been received with new partner ability available in the PCS register PARTNER_ABILITY. The bit is set to 0 (Reset value) when the system management agent performs a read access.
0 —	Reserved

28.6.1.19 SGMII Next Page Transmit (C45_SGMII_NP_TX)

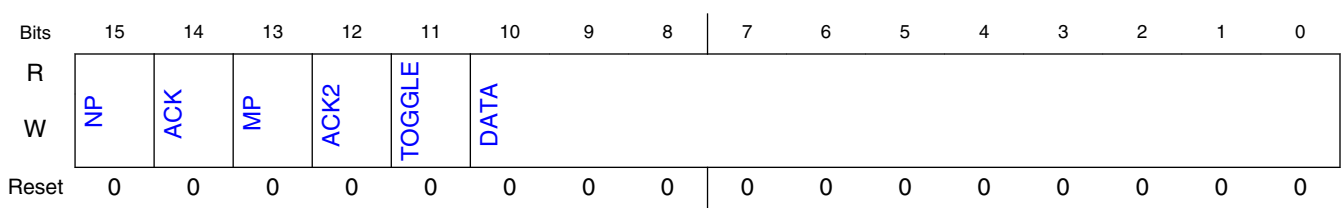
28.6.1.19.1 Offset

Register	Offset
C45_SGMII_NP_TX	8007h

28.6.1.19.2 Function

The SGMII NP TX Register contains next page data to transfer to the remote devide. Writing to this register initiates a next page exchange (sets mr_np_loaded variable).

28.6.1.19.3 Diagram



28.6.1.19.4 Fields

Field	Function
15 NP	Next page Next page indication. Indicates if additional next pages will follow (1) or this is the last page (0).
14	Ack

Table continues on the next page...

Field	Function
ACK	Acknowledge bit used by the autoneg function during message transfer. It has no meaning to the application and should be written with 0 always and ignored on read.
13 MP	Message page Message page (1) or unformatted page (0) format.
12 ACK2	Ack 2 Application level acknowledge to indicate acceptance of a message. Use is defined by application layer.
11 TOGGLE	Toggle The toggle bit is used by the auto-negotiation function to keep track of message exchange. It has no meaning to the application and should be written with 0 always and ignored on read.
10-0 DATA	Data 11 bits of data which can either be a message-id (if MP bit 13=1) or unformatted data (if MP bit 13=0). See IEEE 802.3 Annex 28C for message page encodings

28.6.1.20 SGMII LP Next Page Receive (C45_SGMII_NP_RX)

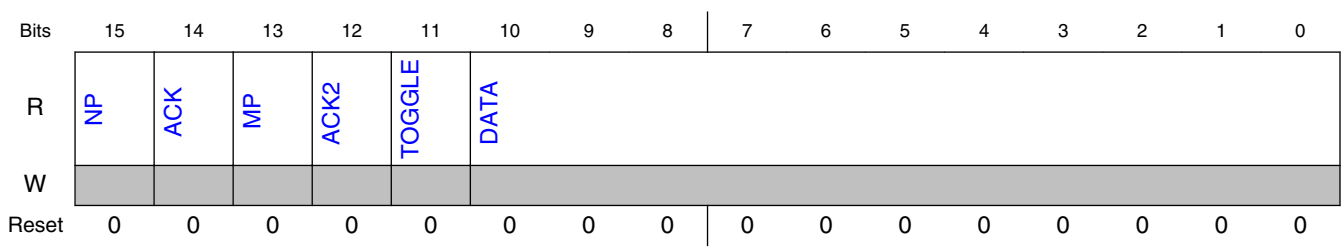
28.6.1.20.1 Offset

Register	Offset
C45_SGMII_NP_RX	8008h

28.6.1.20.2 Function

The SGMII NP RX Register contains the next page data received from the remote device during the latest next page exchange. The value is overridden with every new next page. Next page transfers are controlled by the application.

28.6.1.20.3 Diagram



28.6.1.20.4 Fields

Field	Function
15 NP	Next page Next page indication. Indicates if additional next pages will follow (1) or this is the last page (0).
14 ACK	Ack Acknowledge bit used by the autoneg function during message transfer. It has no meaning to the application and should be written with 0 always and ignored on read.
13 MP	Message page Message page (1) or unformatted page (0) format.
12 ACK2	Ack 2 Application level acknowledge to indicate acceptance of a message. Use is defined by application layer.
11 TOGGLE	Toggle The toggle bit is used by the auto-negotiation function to keep track of message exchange. It has no meaning to the application and should be written with 0 always and ignored on read.
10-0 DATA	Data 11 bits of data which can either be a message-id (if MP bit 13=1) or unformatted data (if MP bit 13=0). See IEEE 802.3 Annex 28C for message page encodings

28.6.1.21 SGMII Extended Status (C45_SGMII_XTND_STAT)

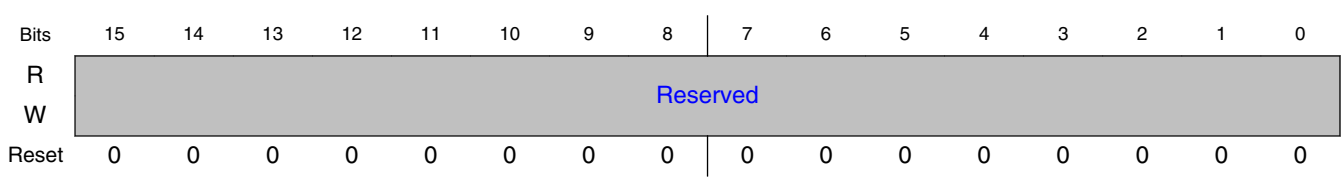
28.6.1.21.1 Offset

Register	Offset
C45_SGMII_XTND_STAT	800Fh

28.6.1.21.2 Function

The SGMII Extended Status Register is reserved, as this device does not implement the optional extended status registers.

28.6.1.21.3 Diagram



28.6.1.21.4 Fields

Field	Function
15-0 —	Reserved. Always 0

28.6.1.22 SGMII Scratch (C45_SGMII_SCRATCH)

28.6.1.22.1 Offset

Register	Offset
C45_SGMII_SCRATCH	8010h

28.6.1.22.2 Function

The SGMII Scratch Register provides a memory location available to test register read and write operations.

28.6.1.22.3 Diagram

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

28.6.1.22.4 Fields

Field	Function
15-0 SCRATCH	Scratch Scratch field.

28.6.1.23 SGMII Design Revision (C45_SGMII_REV)

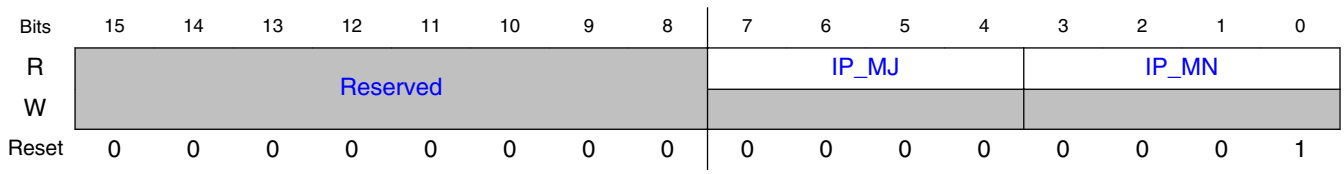
28.6.1.23.1 Offset

Register	Offset
C45_SGMII_REV	8011h

28.6.1.23.2 Function

The SGMII Revision Register contains revision information for the PCS.

28.6.1.23.3 Diagram



28.6.1.23.4 Fields

Field	Function
15-8 —	Reserved
7-4 IP_MJ	Major revision Major revision
3-0 IP_MN	Minor revision Minor revision

28.6.1.24 SGMII Link Timer Lower (C45_SGMII_LINK_TMR_L)

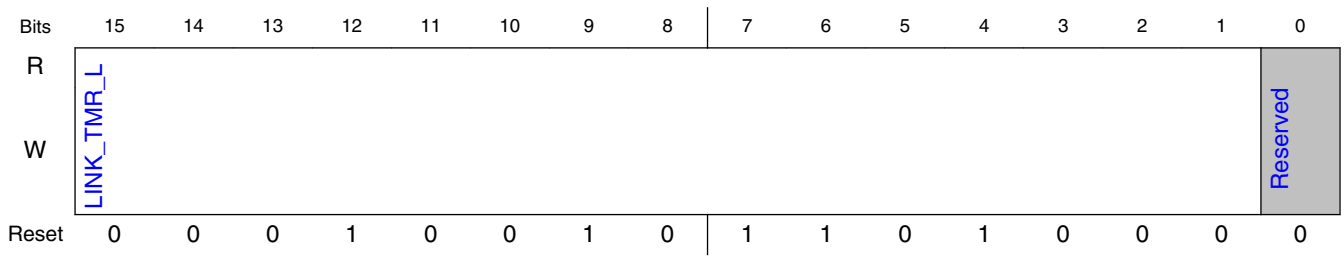
28.6.1.24.1 Offset

Register	Offset
C45_SGMII_LINK_TMR_L	8012h

28.6.1.24.2 Function

The SGMII Link Timer Register Lower contains bits 15:1 of the link timer value.

28.6.1.24.3 Diagram



28.6.1.24.4 Fields

Field	Function
15-1 LINK_TMR_L	Link Timer Lower Link timer[15:1]
0 —	Reserved

28.6.1.25 SGMII Link Timer Upper (C45_SGMII_LINK_TMR_H)

28.6.1.25.1 Offset

Register	Offset
C45_SGMII_LINK_TMR_H	8013h

28.6.1.25.2 Function

The SGMII Link Timer Register Upper contains bits 20:16 of the link timer value.

28.6.1.25.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											LINK_TMR_U				
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1

28.6.1.25.4 Fields

Field	Function
15-5 —	Reserved
4-0 LINK_TMR_U	Link Timer Upper Link timer[20:16]

28.6.1.26 SGMII IF Mode (C45_SGMII_IF_MODE)

28.6.1.26.1 Offset

Register	Offset
C45_SGMII_IF_MODE	8014h

28.6.1.26.2 Function

The SGMII IF Mode Register contains control bits to set the interface mode.

28.6.1.26.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SGMII_DUPLEX	SGMII_SPEED		USE_SGMII_AN	SGMII_EN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

28.6.1.26.4 Fields

Field	Function
15-5 —	Reserved
4 SGMII_DUPLEX	SGMII Duplex SGMII Duplex Mode: Bit ignored when SGMII_EN is set to 0 or USE_SGMII_AN is set to 1. 0b - Full duplex enabled (default)
3-2 SGMII_SPEED	SGMII Speed SGMII Speed. When the PCS operates in SGMII mode (SGMII_EN set to 1) and is programmed not to be automatically configured (USE_SGMII_AN set to 0), sets the PCS speed of operation: Bits ignored when SGMII_EN is set to 0 or USE_SGMII_AN is set to 1. 00b - 10Mbps 01b - 100Mbps 10b - Gigabit (1G or 2.5G) 11b - Reserved
1 USE_SGMII_AN	Use SGMII AN Use the SGMII Auto-Negotiation Results to Program the PCS Speed. When set to 0 (Reset Value), the PCS operation should be programmed with the register bit SGMII_SPEED and SGMII_DUPLEX. When set to 1, the PCS operation is automatically programmed with the Partner abilities advertised during Auto-Negotiation. Ignored when SGMII_EN is set to 0.
0 SGMII_EN	SGMII Enable SGMII Mode Enable. When set to '0' (Reset Value), the PCS operates in standard 1000Base-X Gigabit mode, when set to '1', the PCS operates in SGMII Mode

28.6.2 MDIO_KX_AN register descriptions

The 1000Base-KX auto-negotiation register space is selected when the associated SGMIInCR1[MDEV_PORT] matches the Ethernet MAC port address (MDIO_CTL[PORT_ADDR]) and the device address (MDIO_CTL[DEV_ADDR]) is 07h.

28.6.2.1 MDIO_KX_AN Memory map

MDIO_KX_AN base address: 0h

Offset	Register	Width (In bits)	Access	Reset value
0h	KX AN Control (KX_AN_CR)	16	RW	0000h
1h	KX AN Status (KX_AN_SR)	16	RO	0040h

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
2h	KX AN Device Identifier Upper (KX_AN_DEV_ID_H)	16	RO	0083h
3h	KX AN Device Identifier Lower (KX_AN_DEV_ID_L)	16	RO	E400h
5h	KX AN Devices in Package 0 (KX_AN_DEV_PRES0)	16	RO	0088h
6h	KX AN Devices in Package 1 (KX_AN_DEV_PRES1)	16	RO	0000h
Eh	KX AN Package Identifier Upper (KX_AN_PKG_ID_H)	16	RO	0083h
Fh	KX AN Package Identifier Lower (KX_AN_PKG_ID_L)	16	RO	E400h
10h	KX AN Advertisement 0 (KX_AN_ADVERT0)	16	RW	0C01h
11h	KX AN Advertisement 1 (KX_AN_ADVERT1)	16	RW	003Fh
12h	KX AN Advertisement 2 (KX_AN_ADVERT2)	16	RW	0000h
13h	KX AN LP Base Page Ability 0 (KX_AN_LP_BASE_PG_ABIL0)	16	RO	0000h
14h	KX AN LP Base Page Ability 1 (KX_AN_LP_BASE_PG_ABIL1)	16	RO	0000h
15h	KX AN LP Base Page Ability 2 (KX_AN_LP_BASE_PG_ABIL2)	16	RO	0000h
16h	KX AN XNP Transmit 0 (KX_AN_XNP_TX0)	16	RO	2001h
17h	KX AN XNP Transmit 1 (KX_AN_XNP_TX1)	16	RO	0000h
18h	KX AN XNP Transmit 2 (KX_AN_XNP_TX2)	16	RO	0000h
19h	KX AN LP XNP Ability 0 (KX_AN_LP_XNP_ABIL0)	16	RO	0000h
1Ah	KX AN LP XNP Ability 1 (KX_AN_LP_XNP_ABIL1)	16	RO	0000h
1Bh	KX AN LP XNP Ability 2 (KX_AN_LP_XNP_ABIL2)	16	RO	0000h
30h	KX Backplane Ethernet Status (KX_BP_STAT)	16	RO	0001h
8000h	KX Millisecond Count (KX_MS_CNT)	16	RW	9896h

28.6.2.2 KX AN Control (KX_AN_CR)

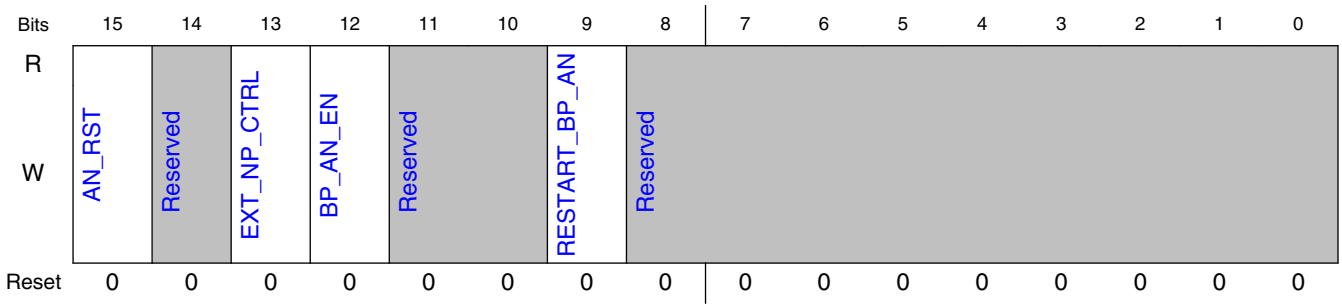
28.6.2.2.1 Offset

Register	Offset
KX_AN_CR	0h

28.6.2.2.2 Function

The 1000Base-KX AN Control Register contains control bits used to enable/disable functions in the PCS, and to initiate commands such as reset.

28.6.2.2.3 Diagram



28.6.2.2.4 Fields

Field	Function
15 AN_RST	AN Reset AN reset This bit is self-clearing 0b - AN normal operation 1b - AN reset
14 —	Reserved
13 EXT_NP_CTRL	Extended Next Page Control Extended Next Page Control When enabled (1) transmission of next page with non-null code field is possible. The next page registers should be initialized and must be set (handshaking) every time a next page is received. When disabled (0) only null next page is transmitted in response to received next pages from link partner. Note: the next page data registers (AN XNP) are writeable only if this bit is set. 0b - Extended next pages are disabled 1b - Extended next pages are enabled
12 BP_AN_EN	Backplane AN Enable Backplane auto-negotiation enable 0b - Disable auto-negotiation 1b - Enable auto-negotiation
11-10 —	Reserved
9 RESTART_BP_AN	Restart backplane AN Restart backplane auto-negotiation This bit is self-clearing 0b - Auto-Negotiation in process, disabled, or not supported 1b - Restart Auto-Negotiation process
8-0 —	Reserved

28.6.2.3 KX AN Status (KX_AN_SR)

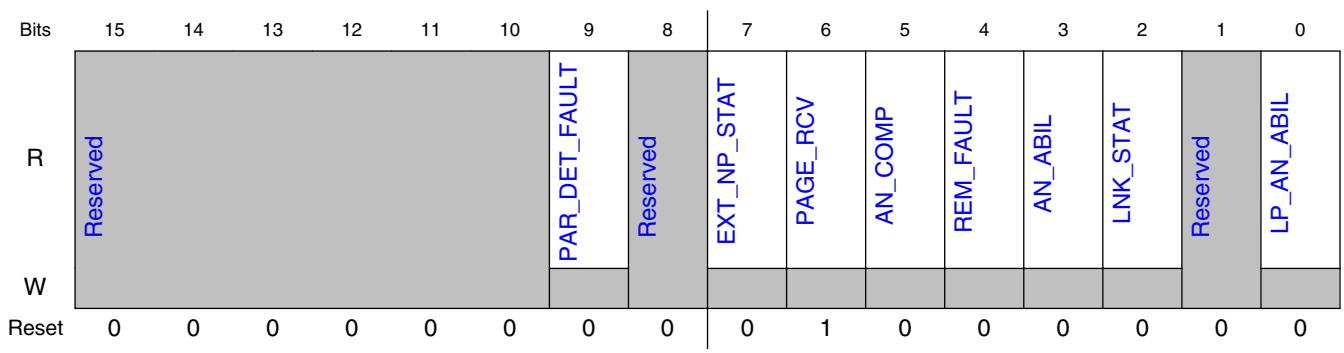
28.6.2.3.1 Offset

Register	Offset
KX_AN_SR	1h

28.6.2.3.2 Function

The 1000Base-KX AN Status Register contains status bits for the 1000Base-KX auto-negotiate function.

28.6.2.3.3 Diagram



28.6.2.3.4 Fields

Field	Function
15-10 —	Reserved
9 PAR_DET_FAULT	Parallel Detection Fault Parallel Detection Fault Once set, stays set until the register is read. 0b - A fault has not been detected via the parallel detection function. 1b - A fault has been detected via the parallel detection function.
8 —	Reserved
7	Extended Next Page Status

Table continues on the next page...

Field	Function
EXT_NP_STAT	Extended Next Page Status This bit is a copy of AN Control[EXT_NP_CTRL] 0b - Extended next pages disabled 1b - Extended next pages enabled
6 PAGE_RCV	Page received Page received Once set, stays set until the register is read 0b - Page has not been received 1b - Page has been received
5 AN_COMP	AN Complete Auto-negotiation complete 0b - Auto-negotiation process not completed 1b - Auto-negotiation process completed
4 REM_FAULT	Remote Fault Remote Fault Once set, stays set until the register is read 0b - No remote fault condition detected 1b - Remote fault condition detected
3 AN_ABIL	AN Ability Auto-negotiation Ability Always set to 1 to indicate that PCS is able to perform auto-negotiation
2 LNK_STAT	Link Status Link Status Once cleared, stays cleared until the register is read 0b - Link is down 1b - Link is up
1 —	Reserved
0 LP_AN_ABIL	Link Partner AN Ability Link Partner Auto-Negotiation Ability 0b - Link Partner is not able to perform auto-negotiation, or Link Partner ability not yet captured 1b - Link Parner is able to perform auto-negotiation

28.6.2.4 KX AN Device Identifier Upper (KX_AN_DEV_ID_H)

28.6.2.4.1 Offset

Register	Offset
KX_AN_DEV_ID_H	2h

28.6.2.4.2 Function

The 1000Base-KX AN Device Identifier Upper Register contains the upper half of the 32-bit device identifier.

28.6.2.4.3 Diagram

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

28.6.2.4.4 Fields

Field	Function
15-0 AN_DEV_ID_U	AN Device Identifier Upper PCS Device Identifier Upper: OUI[3:18]

28.6.2.5 KX AN Device Identifier Lower (KX_AN_DEV_ID_L)

28.6.2.5.1 Offset

Register	Offset
KX_AN_DEV_ID_L	3h

28.6.2.5.2 Function

The 1000Base-KX AN Device Identifier Lower Register contains the lower half of the 32-bit Device Identifier.

28.6.2.5.3 Diagram

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

28.6.2.5.4 Fields

Field	Function
15-0 AN_DEV_ID_L	AN Device ID Lower PCS Device Identifier Lower: 15:10 OUI[19:24] 9:4 Manufacturer's Model Number 3:0 Revision Number

28.6.2.6 KX AN Devices in Package 0 (KX_AN_DEV_PRES0)

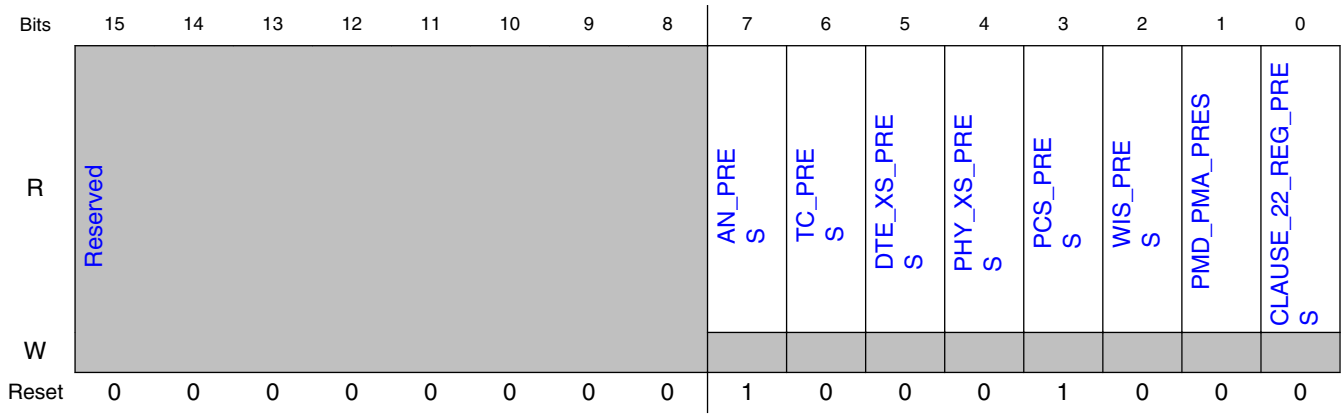
28.6.2.6.1 Offset

Register	Offset
KX_AN_DEV_PRES0	5h

28.6.2.6.2 Function

The 1000Base-KX AN Devices in Package 0 Register contains half of the AN devices in package status.

28.6.2.6.3 Diagram



28.6.2.6.4 Fields

Field	Function
15-8 —	Reserved
7 AN_PRES	AN Present 0b - Auto-Negotiation not present in package 1b - Auto-Negotiation present in package
6 TC_PRES	TC Present 0b - TC not present in package 1b - TC present in package
5 DTE_XS_PRES	DTE XS Present 0b - DTE XS not present in package 1b - DTE XS present in package
4 PHY_XS_PRES	PHY XS Present 0b - PHY XS not present in package 1b - PHY XS present in package
3 PCS_PRES	PCS Present 0b - PCS not present in package 1b - PCS present in package
2 WIS_PRES	WIS Present 0b - WIS not present in package 1b - WIS present in package
1 PMD_PMA_PRES	PMD/PMA Present 0b - PMA/PMD not present in package 1b - PMA/PMD present in package
0 CLAUSE_22_REG_PRES	Clause 22 register present 0b - Clause 22 registers not present in package 1b - Clause 22 registers present in package

28.6.2.7 KX AN Devices in Package 1 (KX_AN_DEV_PRES1)

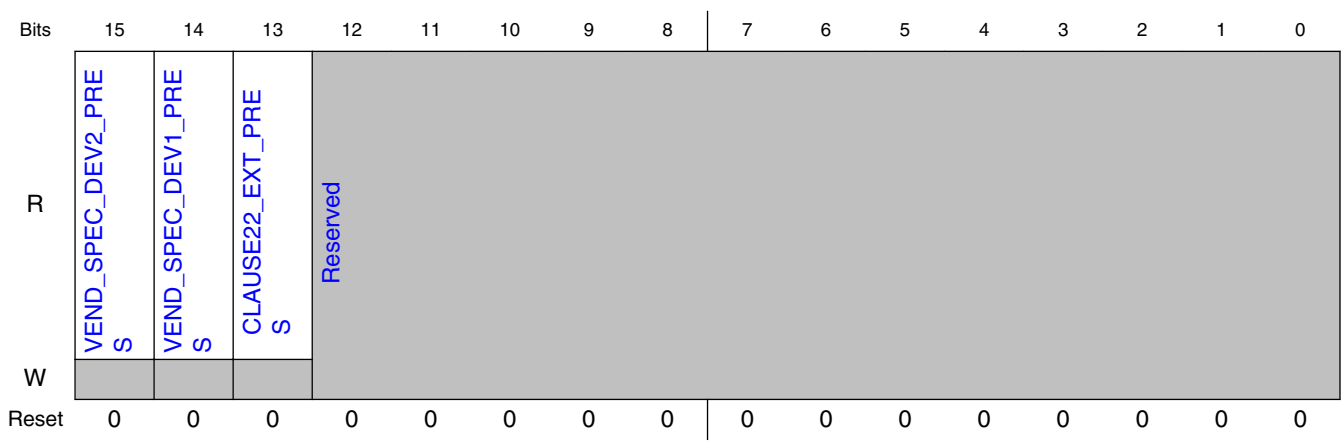
28.6.2.7.1 Offset

Register	Offset
KX_AN_DEV_PRES1	6h

28.6.2.7.2 Function

The 1000Base-KX AN Devices in Package 1 Register contains half of the AN devices in package status.

28.6.2.7.3 Diagram



28.6.2.7.4 Fields

Field	Function
15 VEND_SPEC_DEV2_PRES	Vendor Specific Device 2 Present 0b - Vendor specific device 2 not present in package 1b - Vendor specific device 2 present in package
14 VEND_SPEC_DEV1_PRES	Vendor Specific Device 1 Present 0b - Vendor specific device 1 not present in package 1b - Vendor specific device 1 present in package
13 CLAUSE22_EXT_PRES	Clause 22 Extension Present 0b - Clause 22 extension not present in package 1b - Clause 22 extension present in package
12-0 —	Reserved

28.6.2.8 KX AN Package Identifier Upper (KX_AN_PKG_ID_H)

28.6.2.8.1 Offset

Register	Offset
KX_AN_PKG_ID_H	Eh

28.6.2.8.2 Function

The 1000Base-KX AN Package Device Identifier Upper Register contains the upper half of the 32-bit Package Identifier.

28.6.2.8.3 Diagram

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

28.6.2.8.4 Fields

Field	Function
15-0 PKG_DEV_ID_U	Package Device Identifier Upper Package Device Identifier Upper: OUI[3:18]

28.6.2.9 KX AN Package Identifier Lower (KX_AN_PKG_ID_L)

28.6.2.9.1 Offset

Register	Offset
KX_AN_PKG_ID_L	Fh

28.6.2.9.2 Function

The KX AN Package Identifier Lower Register contains the lower half of the 32-bit Package Identifier.

28.6.2.9.3 Diagram

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

28.6.2.9.4 Fields

Field	Function
15-0 PKG_DEV_ID_L	Package Device Identifier Lower Package Device Identifier Lower: 15:10 OUI[19:24] 9:4 Manufacturer's Model Number 3:0 Revision Number

28.6.2.10 KX AN Advertisement 0 (KX_AN_ADVERT0)

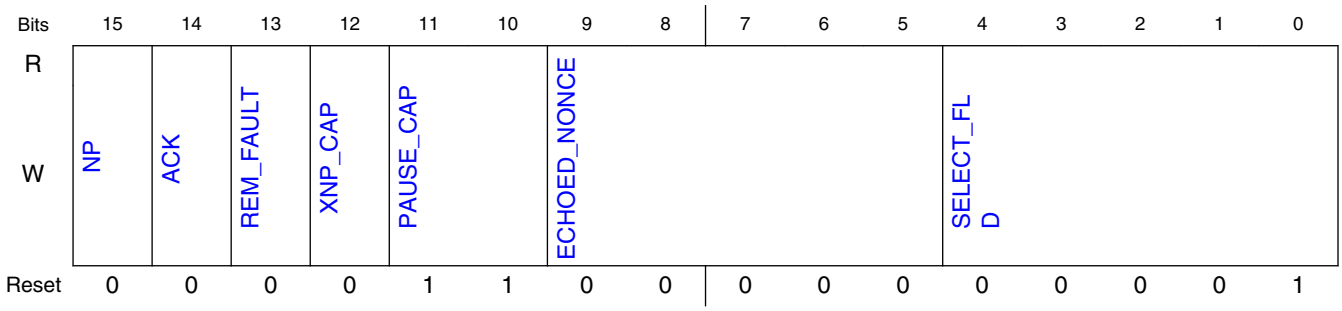
28.6.2.10.1 Offset

Register	Offset
KX_AN_ADVERT0	10h

28.6.2.10.2 Function

The 1000Base-KX AN Advertisement Register 0 contains bits 15:0 of the 48-bit page bits used during base page exchange. They should be programmed before auto-negotiation is started. The 48-bit value is distributed over registers 0, 1, 2.

28.6.2.10.3 Diagram



28.6.2.10.4 Fields

Field	Function
15 NP	Next Page Next Page 0b - The device has no next pages to send 1b - The device has next pages to send
14 ACK	Ack Acknowledge Should always be set to 0
13 REM_FAULT	Remote Fault Remote Fault 0b - No remote fault advertised 1b - Advertise remote fault
12 XNP_CAP	Extended Next Page Capabiltity Extended Next Page Capability 0b - Device does not support extended next pages
11-10 PAUSE_CAP	Pause Capability Pause Capability (ASM_DIR:PAUSE) Default is 11 00b - No PAUSE 01b - Symmetric PAUSE 10b - Asymmetric PAUSE toward link partner 11b - Both Symmetric PAUSE and Asymmetric PAUSE toward local device
9-5 ECHOED_NONCE	Echoed Nonce Echoed Nonce field (E4:0) Contains the transmitted nonce field from the link partner Only valid if ACK is set
4-0 SELECT_FLD	Selector field Selector field (S4:0) All values not shown are reserved 00001b - IEEE Std 802.3

28.6.2.11 KX AN Advertisement 1 (KX_AN_ADVERT1)

28.6.2.11.1 Offset

Register	Offset
KX_AN_ADVERT1	11h

28.6.2.11.2 Function

The 1000Base-KX AN Advertisement Register 1 contains bits 31:16 of the 48-bit page bits used during base page exchange. They should be programmed before auto-negotiation is started. The 48-bit value is distributed over registers 0, 1, 2.

28.6.2.11.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	TECH_A10_A0								TRANS_NONCE							
Reset	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

28.6.2.11.4 Fields

Field	Function
15-5	Technology A10:A0
TECH_A10_A0	Bits A10:A0 of the technology field. 5 (A0): 1000Base-KX 6 (A1): 10GBase-KX4 7 (A2): reserved 8 (A3): reserved 9 (A4): reserved 10 (A5): reserved 11 (A6): reserved 12 (A7): reserved 13 (A8): reserved 14 (A9): reserved 15 (A10): reserved

Table continues on the next page...

Field	Function
	Must set all bits except A0 to 0
4-0 TRANS_NONCE	Transmitted Nonce Transmitted Nonce Two devices must have a different nonce for auto-negotiation to operation (i.e. a loopback will not allow auto-negotiation to complete.

28.6.2.12 KX AN Advertisement 2 (KX_AN_ADVERT2)

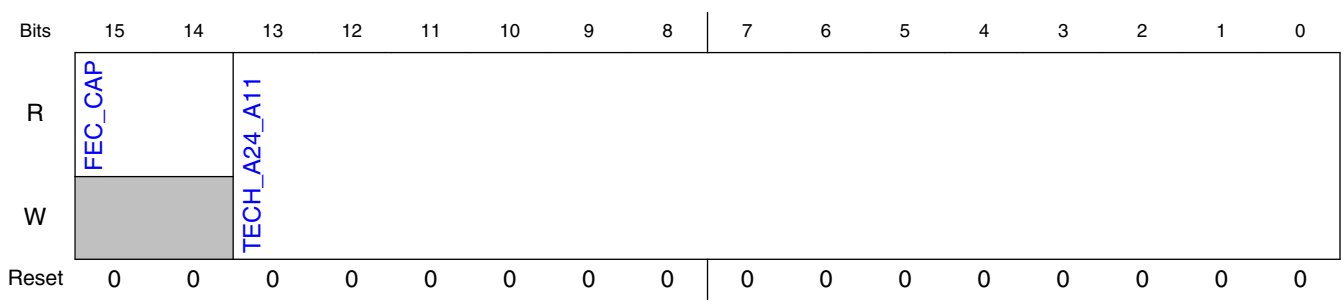
28.6.2.12.1 Offset

Register	Offset
KX_AN_ADVERT2	12h

28.6.2.12.2 Function

The 1000Base-KX AN Advertisement Register 0 contains bits 47:32 of the 48-bit page bits used during base page exchange. They should be programmed before auto-negotiation is started. The 48-bit value is distributed over registers 0, 1, 2. Register 2 must be written last.

28.6.2.12.3 Diagram



28.6.2.12.4 Fields

Field	Function
15-14 FEC_CAP	FEC Capability Capability bits (F1:0) for FEC support

Table continues on the next page...

Field	Function
	00b - PCS does not implement a FEC function
13-0 TECH_A24_A11	Technology A24:A11 Technology field A24:11 All bits reserved. Should be set to 0

28.6.2.13 KX AN LP Base Page Ability 0 (KX_AN_LP_BASE_PG_ABIL0)

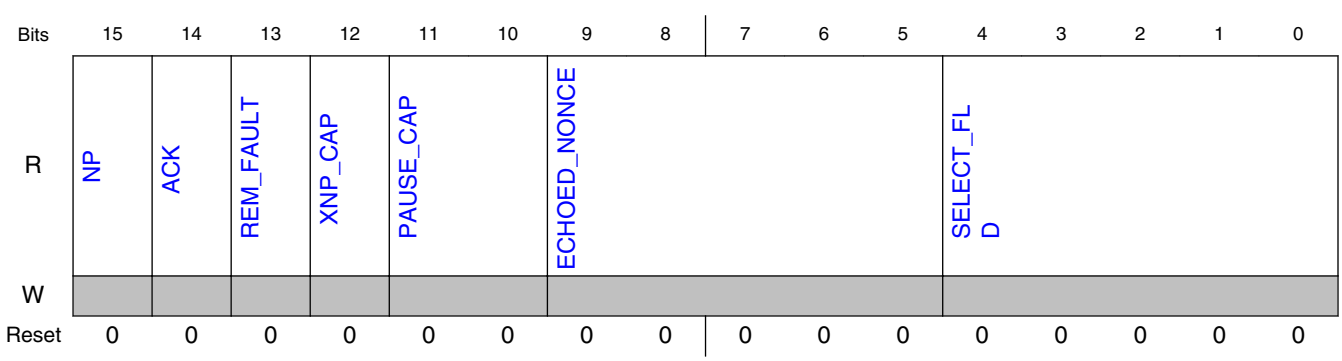
28.6.2.13.1 Offset

Register	Offset
KX_AN_LP_BASE_PG_ABIL0	13h

28.6.2.13.2 Function

The 1000Base-KX AN LP Base Page Ability Register 0 contains bits 15:0 of the link partner base storage. The 48-bit value is stored over registers 0, 1 and 2.

28.6.2.13.3 Diagram



28.6.2.13.4 Fields

Field	Function
15 NP	Next Page Next Page

Table continues on the next page...

Field	Function
	0b - The link partner has no next pages to send 1b - The link partner has next pages to send
14 ACK	Ack Acknowledge 0b - No link partner acknowledge 1b - Link partner acknowledge
13 REM_FAULT	Remote Fault Remote Fault 0b - No remote fault advertised by link partner 1b - Remote fault advertised by link partner
12 XNP_CAP	Extended Next Page Capability Extended Next Page Capability 0b - Link partner does not support extended next pages 1b - Link parnet supports extended next pages
11-10 PAUSE_CAP	Pause capability Pause Capability (ASM_DIR:PAUSE) 00b - No PAUSE 01b - Symmetric PAUSE 10b - Asymmetric PAUSE toward local device 11b - Both Symmetric PAUSE and Asymmetric PAUSE toward link partner
9-5 ECHOED_NONCE	Echoed nonce Echoed Nonce field (E4:0) Contains the transmitted nonce field from the device as seen by the link partner Only valid if ACK is set
4-0 SELECT_FLD	Selector field Selector field (S4:0) All values not shown are reserved 00001b - IEEE Std 802.3

28.6.2.14 KX AN LP Base Page Ability 1 (KX_AN_LP_BASE_PG_ABIL1)

28.6.2.14.1 Offset

Register	Offset
KX_AN_LP_BASE_PG_ABIL1	14h

28.6.2.14.2 Function

The 1000Base-KX AN LP Base Page Ability Register 1 contains bits 31:16 of the link partner base storage. The 48-bit value is distributed over registers 0, 1, 2.

28.6.2.14.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TECH_A10_A0											TRANS_NONCE				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

28.6.2.14.4 Fields

Field	Function
15-5 TECH_A10_A0	Technology A10:A0 Bits A10:A0 of the technology field. 5 (A0): 1000Base-KX 6 (A1): 10GBase-KX4 7 (A2): reserved 8 (A3): reserved 9 (A4): reserved 10 (A5): reserved 11 (A6): reserved 12 (A7): reserved 13 (A8): reserved 14 (A9): reserved 15 (A10): reserved
4-0 TRANS_NONCE	Transmitted Nonce Transmitted Nonce

28.6.2.15 KX AN LP Base Page Ability 2 (KX_AN_LP_BASE_PG_ABIL2)

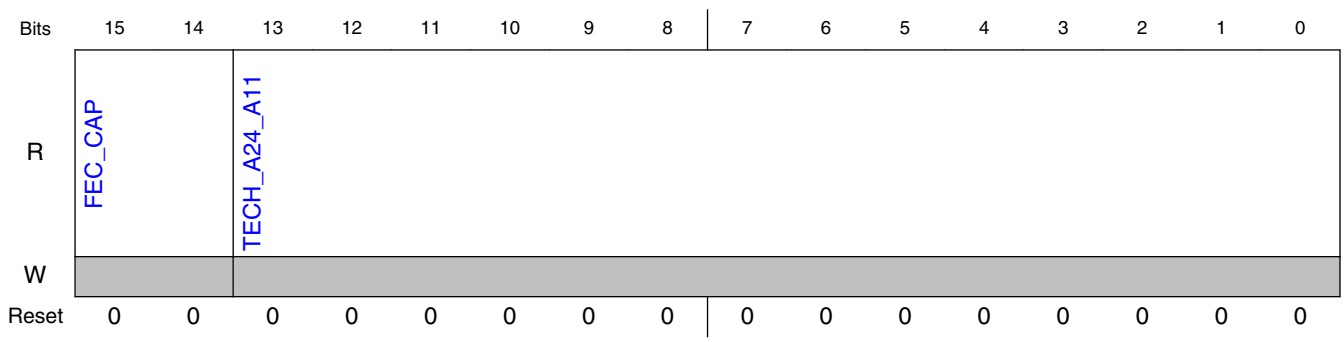
28.6.2.15.1 Offset

Register	Offset
KX_AN_LP_BASE_PG_ABIL2	15h

28.6.2.15.2 Function

The 1000Base-KX AN LP Base Page Ability Register 2 contains bits 47:32 of the link partner base storage. The 48-bit value is distributed over registers 0, 1, 2.

28.6.2.15.3 Diagram



28.6.2.15.4 Fields

Field	Function
15-14 FEC_CAP	FEC Capability Capability bits (F1:0) for FEC support F1: FEC ability F0: FEC requested
13-0 TECH_A24_A11	Technology A24:A11 Technology field A24:11 Reserved for future technology

28.6.2.16 KX AN XNP Transmit 0 (KX_AN_XNP_TX0)

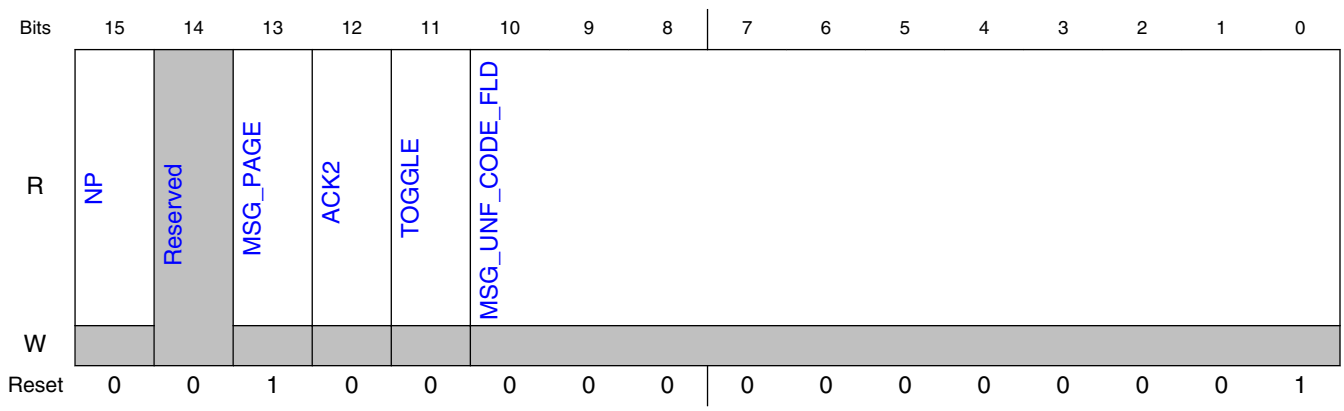
28.6.2.16.1 Offset

Register	Offset
KX_AN_XNP_TX0	16h

28.6.2.16.2 Function

The 1000Base-KX AN XNP Transmit Register 0 contains bits 15:0 of the XNP Transmit Register. The 48-bit value is distributed over registers 0, 1, 2.

28.6.2.16.3 Diagram



28.6.2.16.4 Fields

Field	Function
15 NP	Next Page Next Page 0b - Device does not have any more Next Pages to send 1b - Device has more Next Pages to Send
14 —	Reserved
13 MSG_PAGE	Message Page Message Page 0b - Next page is an unformatted page 1b - Next page is a message page
12 ACK2	Ack 2 Acknowledge 2 0b - The receiver is not able to act on the information defined in the message 1b - The receiver is able to act on the information defined in the message

Table continues on the next page...

Field	Function
11 TOGGLE	Toggle Toggle
10-0 MSG_UNF_CODE_FLD	Message/Unformatted Code Field Message/Unformatted Code Field When MSG_PAGE=1: Message code field, else unformatted code field. For the null message code, the value is 0x1

28.6.2.17 KX AN XNP Transmit 1 (KX_AN_XNP_TX1)

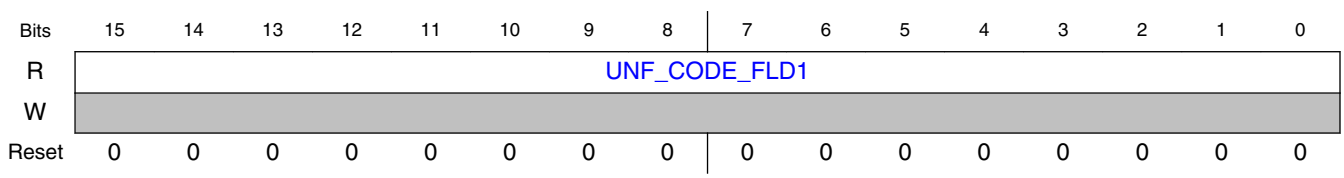
28.6.2.17.1 Offset

Register	Offset
KX_AN_XNP_TX1	17h

28.6.2.17.2 Function

The 1000Base-KX AN XNP Transmit Register 1 contains bits 31:16 of the XNP Transmit Register. The 48-bit value is distributed over registers 0, 1, 2.

28.6.2.17.3 Diagram



28.6.2.17.4 Fields

Field	Function
15-0 UNF_CODE_FLD1	Unformatted Code Field 1 Unformatted Code Field 1

28.6.2.18 KX AN XNP Transmit 2 (KX_AN_XNP_TX2)

28.6.2.18.1 Offset

Register	Offset
KX_AN_XNP_TX2	18h

28.6.2.18.2 Function

The 1000Base-KX AN XNP Transmit Register 2 contains bits 48:32 of the XNP Transmit Register. The 48-bit value is distributed over registers 0, 1, 2. Register 2 must be written last.

28.6.2.18.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	UNF_CODE_FLD2															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

28.6.2.18.4 Fields

Field	Function
15-0 UNF_CODE_FLD2	Unformatted Code Field 2 Unformatted Code Field 2

28.6.2.19 KX AN LP XNP Ability 0 (KX_AN_LP_XNP_ABIL0)

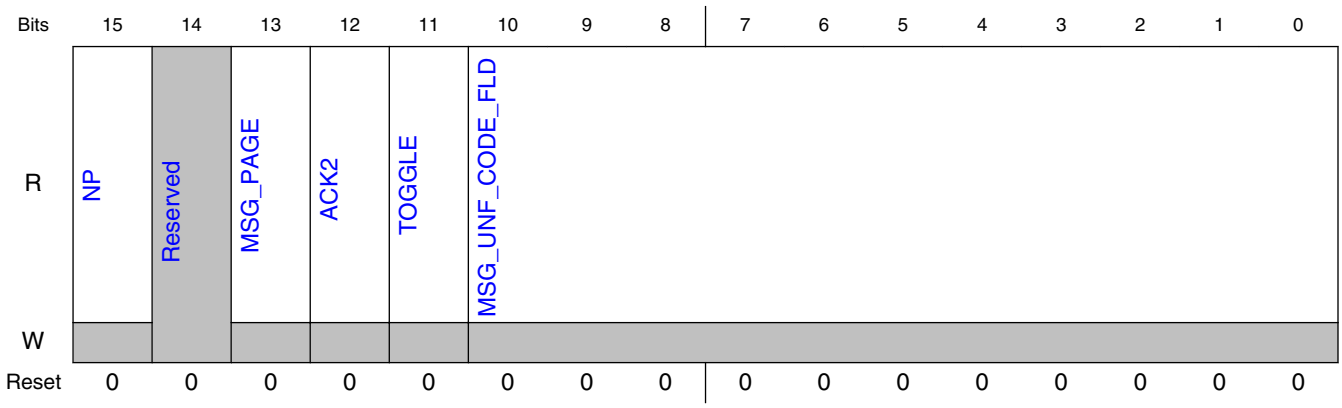
28.6.2.19.1 Offset

Register	Offset
KX_AN_LP_XNP_ABIL0	19h

28.6.2.19.2 Function

The 1000Base-KX AN LP XNP Ability Register 0 contains bits 15:0 of the LP XNP Ability Register. The 48-bit value is distributed over registers 0, 1, 2.

28.6.2.19.3 Diagram



28.6.2.19.4 Fields

Field	Function
15 NP	Next Page Next Page 0b - Device does not have any more Next Pages to send 1b - Device has more Next Pages to Send
14 —	Reserved
13 MSG_PAGE	Message Page Message Page 0b - Next page is an unformatted page 1b - Next page is a message page
12 ACK2	Ack 2 Acknowledge 2 0b - The receiver is not able to act on the information defined in the message 1b - The receiver is able to act on the information defined in the message
11 TOGGLE	Toggle Toggle
10-0 MSG_UNF_CODE_FLD	Message/Unformatted Code Field Message/Unformatted Code Field When MSG_PAGE=1: Message code field, else unformatted code field. For the null message code, the value is 0x1

28.6.2.20 KX AN LP XNP Ability 1 (KX_AN_LP_XNP_ABIL1)

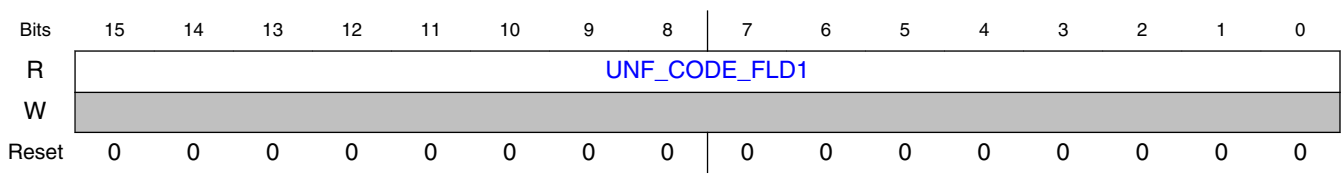
28.6.2.20.1 Offset

Register	Offset
KX_AN_LP_XNP_ABIL1	1Ah

28.6.2.20.2 Function

The 1000Base-KX AN LP XNP Ability Register 1 contains bits 31:16 of the XNP Transmit Register. The 48-bit value is distributed over registers 0, 1, 2.

28.6.2.20.3 Diagram



28.6.2.20.4 Fields

Field	Function
15-0 UNF_CODE_FLD1	Unformatted Code Field 1 Unformatted Code Field 1

28.6.2.21 KX AN LP XNP Ability 2 (KX_AN_LP_XNP_ABIL2)

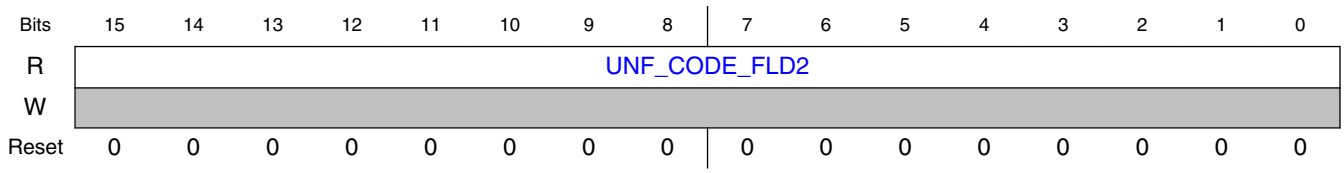
28.6.2.21.1 Offset

Register	Offset
KX_AN_LP_XNP_ABIL2	1Bh

28.6.2.21.2 Function

The 1000Base-KX AN LP XNP Ability Register 2 contains bits 48:32 of the XNP Transmit Register. The 48-bit value is distributed over registers 0, 1, 2.

28.6.2.21.3 Diagram



28.6.2.21.4 Fields

Field	Function
15-0 UNF_CODE_FLD2	Unformatted Code Field 2 Unformatted Code Field 2

28.6.2.22 KX Backplane Ethernet Status (KX_BP_STAT)

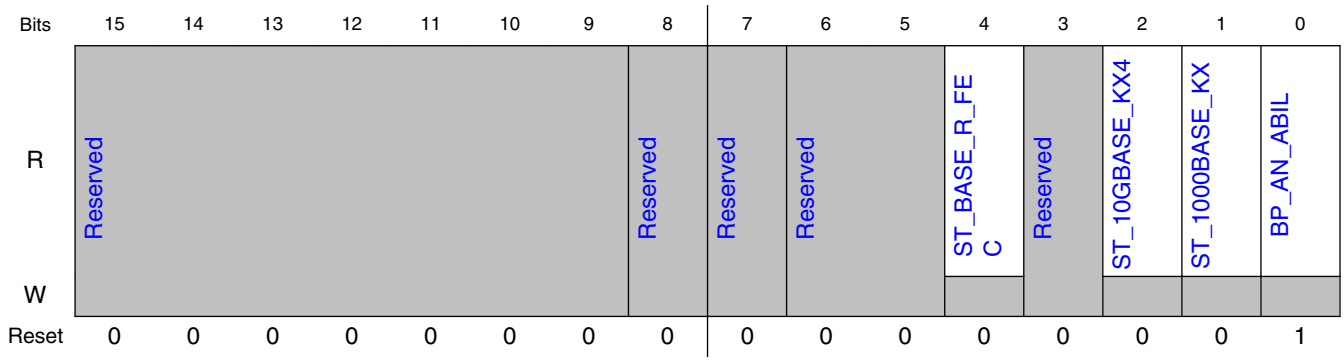
28.6.2.22.1 Offset

Register	Offset
KX_BP_STAT	30h

28.6.2.22.2 Function

The 1000Base-KX AN Backplane Ethernet Status Register provides the result after auto-negotiation DME page exchange completed.

28.6.2.22.3 Diagram



28.6.2.22.4 Fields

Field	Function
15-9 —	Reserved
8 —	Reserved
7 —	Reserved
6-5 —	Reserved
4 ST_BASE_R_FEC	Base-R FEC Always 0 after negotiation complete
3 —	Reserved
2 ST_10GBASE_KX4	10GBase-KX4 Always 0 after negotiation complete
1 ST_1000BASE_KX	1000Base-KX 1000Base-KX 0b - Not auto-negotiated to 1000Base-KX 1b - Auto-negotiated to 1000Base-KX
0 BP_AN_ABIL	Backplane AN Ability 1000Base-KX capable PHY type is implemented Always 1

28.6.2.23 KX Millisecond Count (KX_MS_CNT)

28.6.2.23.1 Offset

Register	Offset
KX_MS_CNT	8000h

28.6.2.23.2 Function

The 1000Base-KX AN Millisecond Counter Register contains the upper 16 bits of the 18-bit counter value for counting 1 ms.

28.6.2.23.3 Diagram

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

28.6.2.23.4 Fields

Field	Function
15-0 COUNT_MS	Count Milliseconds Count milliseconds Upper 16-bits value of the 18-bit counter is provided for counting 1ms. The milliseconds counter operates on 8bit samples (i.e. 6.4ns) incrementing by 4 with every sample (i.e. 0x9896*4*6.4ns=1ms)

28.6.3 MDIO_KX_VENDOR_SPEC register descriptions

The 1000Base-KX vendor-specific 1 register space is selected when the associated SGMIInCR1[MDEV_PORT] matches the Ethernet MAC port address (MDIO_CTL[PORT_ADDR]) and the device address (MDIO_CTL[DEV_ADDR]) is 1Dh.

28.6.3.1 MDIO_KX_VENDOR_SPEC Memory map

MDIO_KX_VENDOR_SPEC base address: 0h

Offset	Register	Width (In bits)	Access	Reset value
0h	KX Revision (KX_VND_REV)	16	RO	0114h
1h	KX Scratch (KX_VND_SCRATCH)	16	RW	0000h
2h	KX PCS Interrupt Event (KX_VND_PCS_INT)	16	RW	0000h
3h	KX PCS Interrupt Mask (KX_VND_INT_MSK)	16	RW	0000h
4h	KX AN Interrupt Event (KX_VND_AN_INT)	16	RW	0000h
5h	KX AN Interrupt Mask (KX_VND_AN_INT_MSK)	16	RW	0000h

28.6.3.2 KX Revision (KX_VND_REV)

28.6.3.2.1 Offset

Register	Offset
KX_VND_REV	0h

28.6.3.2.2 Function

The 1000Base-KX version register contains version information for the KX PHY.

28.6.3.2.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CFG_VER								IP_VER				IP_REV			
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	1	0	1	0	0

28.6.3.2.4 Fields

Field	Function
15-8 CFG_VER	Configuration/Integration Version Configuration/Integration Version

Table continues on the next page...

Field	Function
7-4 IP_VER	IP Version IP Version
3-0 IP_REV	IP revision IP Revision

28.6.3.3 KX Scratch (KX_VND_SCRATCH)

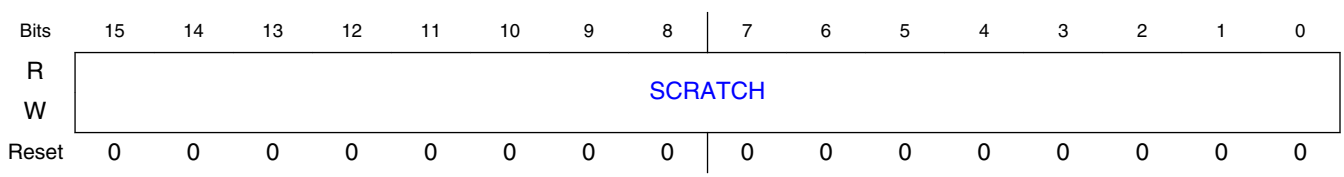
28.6.3.3.1 Offset

Register	Offset
KX_VND_SCRATCH	1h

28.6.3.3.2 Function

The 1000Base-KX scratch register may be used to test register reads and writes.

28.6.3.3.3 Diagram



28.6.3.3.4 Fields

Field	Function
15-0 SCRATCH	Scratch Scratch register

28.6.3.4 KX PCS Interrupt Event (KX_VND_PCS_INT)

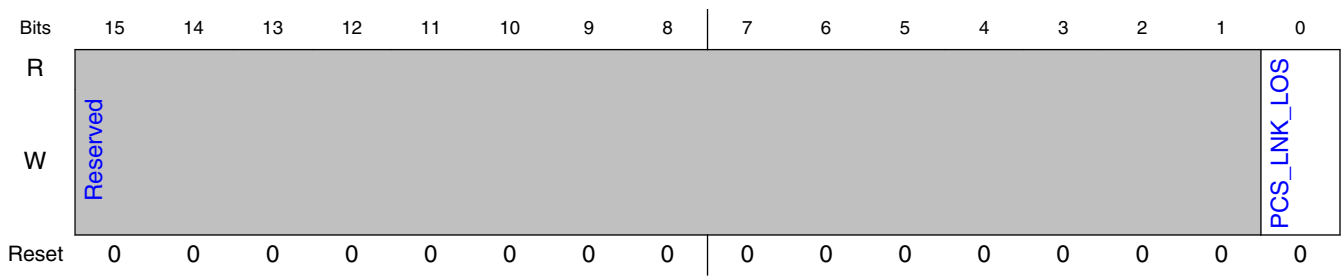
28.6.3.4.1 Offset

Register	Offset
KX_VND_PCS_INT	2h

28.6.3.4.2 Function

The 1000Base-KX PCS interrupt event register contains detect bits for KX PCS interrupt events.

28.6.3.4.3 Diagram



28.6.3.4.4 Fields

Field	Function
15-1 —	Reserved
0 PCS_LNK_LOS	PCS Link LOS PCS Link Loss Cleared on register read 0b - PCS has not detected a loss of link synchronization 1b - PCS has detected a loss of link synchronization

28.6.3.5 KX PCS Interrupt Mask (KX_VND_INT_MSK)

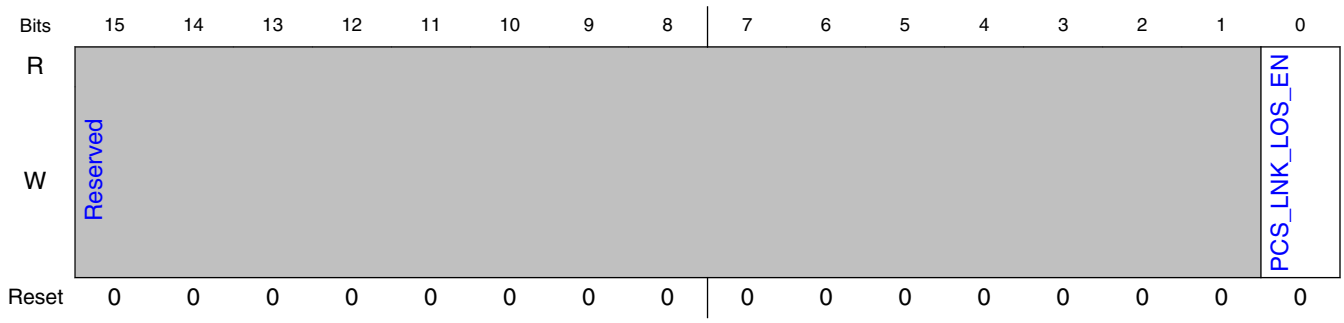
28.6.3.5.1 Offset

Register	Offset
KX_VND_INT_MSK	3h

28.6.3.5.2 Function

The 1000Base-KX PCS Interrupt Event Register contains mask bits for KX PCS interrupt events. See the Frame Manager reference manual for details on handling of PCS interrupts.

28.6.3.5.3 Diagram



28.6.3.5.4 Fields

Field	Function
15-1 —	Reserved
0 PCS_LNK_LOS_EN	PCS Link LOS Enable PCS Link Loss Event Enable 0b - A PCS loss of link synchronization will not cause an interrupt event 0b - A PCS loss of link synchronization will cause an interrupt event

28.6.3.6 KX AN Interrupt Event (KX_VND_AN_INT)

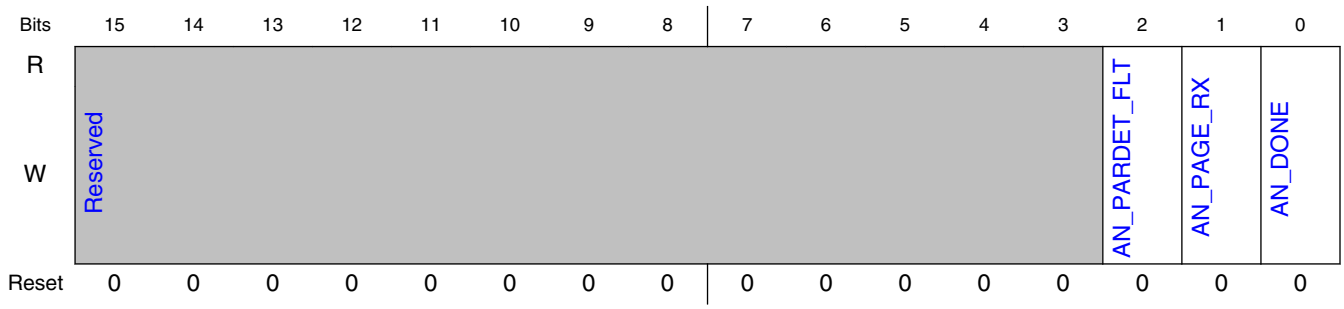
28.6.3.6.1 Offset

Register	Offset
KX_VND_AN_INT	4h

28.6.3.6.2 Function

The 1000Base-KX PCS Interrupt Event Register contains detect bits for KX auto-negotiation interrupt events.

28.6.3.6.3 Diagram



28.6.3.6.4 Fields

Field	Function
15-3 —	Reserved
2 AN_PARDET_FLT	AN Parallel Detection Fault Parallel detection fault Cleared on register read 0b - No fault detected 1b - Ambiguous link status detected while waiting on DME page receive
1 AN_PAGE_RX	AN Page Rx Auto-negotiate page receive Indicates the validity of the remote ability word (base page or next page) Cleared on register read 0b - An ability word was not received from the remote device 1b - An ability word was received from the remote device during backplane auto-negotiation
0 AN_DONE	AN Done Backplane auto-negotiation complete Note: not used for 1000Base-X/SGMII auto-negotiation Cleared on register read 0b - Backplane auto-negotiation not complete 1b - Backplane auto-negotiation complete

28.6.3.7 KX AN Interrupt Mask (KX_VND_AN_INT_MSK)

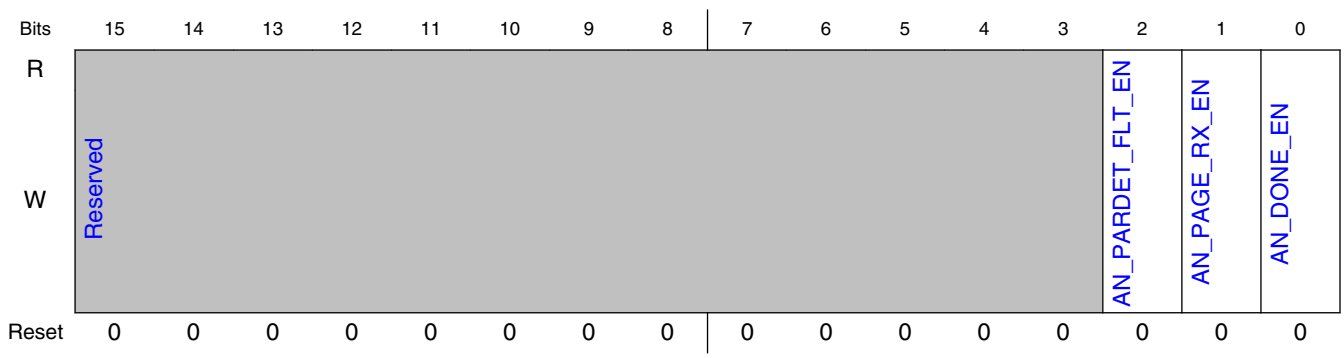
28.6.3.7.1 Offset

Register	Offset
KX_VND_AN_INT_MSK	5h

28.6.3.7.2 Function

The 1000Base-KX PCS Interrupt Event Register contains mask bits for KX AN interrupt events. See the Frame Manager reference manual for details on handling of AN interrupts.

28.6.3.7.3 Diagram



28.6.3.7.4 Fields

Field	Function
15-3 —	Reserved
2 AN_PARDET_FLT_EN	AN Parallel Fault Enable AN Parallel Detection Fault Event Enable 0b - AN Parallel Dection Fault will not cause an interrupt event 1b - AN Parallel Dection Fault will cause an interrupt event
1 AN_PAGE_RX_EN	AN Page Rx Enable AN Page Recieve Event Enable 0b - AN Page Receive will not cause an interrupt event 1b - AN Page Receive will cause an interrupt event
0	AN Done Enable

Field	Function
AN_DONE_EN	AN Done Event Enable 0b - AN done will not cause an interrupt event 1b - AN done will cause an interrupt event

28.6.4 MDIO_SGMII register descriptions

The SGMII MDIO register space is selected when the associated SGMIInCR1[MDEV_PORT] matches the Ethernet MAC PHY address (MDIO_CTL[PHY_ADDR]).

28.6.4.1 MDIO_SGMII Memory map

MDIO_SGMII base address: 0h

Offset	Register	Width (In bits)	Access	Reset value
0h	SGMII Control (SGMII_CR)	16	RW	1140h
1h	SGMII Status (SGMII_SR)	16	RO	0009h
2h	SGMII PHY Identifier Upper (SGMII_PHY_ID_H)	16	RO	0083h
3h	SGMII PHY Identifier Lower (SGMII_PHY_ID_L)	16	RO	E400h
4h	SGMII Device Ability for 1000Base-X (SGMII_DEV_ABIL_1KBX)	16	RW	01A0h
4h	SGMII Device Ability for SGMII (SGMII_DEV_ABIL_SGMII)	16	RW	01A0h
5h	SGMII Partner Ability for 1000Base-X (SGMII_LP_DEV_ABIL_1KBX)	16	RO	0000h
5h	SGMII Partner Ability for SGMII (SGMII_LP_DEV_ABIL_SGMII)	16	RO	0000h
6h	SGMII AN Expansion (SGMII_AN_EXP)	16	RO	0004h
7h	SGMII Next Page Transmit (SGMII_NP_TX)	16	RW	0000h
8h	SGMII LP Next Page Receive (SGMII_NP_RX)	16	RO	0000h
Fh	SGMII Extended Status (SGMII_XTND_STAT)	16	RU	0000h
10h	SGMII Scratch (SGMII_SCRATCH)	16	RW	0000h
11h	SGMII Design Revision (SGMII_REV)	16	RO	0001h
12h	SGMII Link Timer Lower (SGMII_LINK_TMR_L)	16	RW	12D0h
13h	SGMII Link Timer Upper (SGMII_LINK_TMR_H)	16	RW	0013h
14h	SGMII IF Mode (SGMII_IF_MODE)	16	RW	0000h

28.6.4.2 SGMII Control (SGMII_CR)

28.6.4.2.1 Offset

Register	Offset
SGMII_CR	0h

28.6.4.2.2 Function

The SGMII Control Register contains control bits used to enable/disable functions in the PCS, and to initiate commands such as reset.

28.6.4.2.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RST	LPBK	SPEED_SEL 0	AN_EN	PD	ISOLATE	RESTART_AN	FD	COL_TEST	SPEED_SEL 1	Reserved					
W																
Reset	0	0	0	1	0	0	0	1	0	1	0	0	0	0	0	0

28.6.4.2.4 Fields

Field	Function
15 RST	Reset Self-Clearing Read / Write Reset Command Register. When set to 1, a synchronous reset pulse is generated which resets all the PCS state machines, the Comma detection function and the 8b/10b coder / decoder. Should be set to 0 (default) for normal operation.
14 LPBK	Loopback Loopback Command. 0b - Normal operation 1b - Reserved
13 SPEED_SEL0	Speed selection (LSB) Speed selection (LSB) All others reserved Note: SGMII speed is controlled with the IF Mode register MSB,LSB 1,0: 1000 Mb/s

Table continues on the next page...

Field	Function
	All others reserved
12 AN_EN	AN Enable Auto-negotiation enable 0b - Disable auto-negotiation 1b - Enable auto-negotiation
11 PD	Power down Power down 0b - Normal operation 1b - Power down PCS. The PCS is held in internal soft reset until the bit is cleared again.
10 ISOLATE	Isolate Isolate 0b - Normal operation 1b - Reserved
9 RESTART_AN	Restart AN Restart auto-negotiation This bit is self-clearing 0b - Normal operation 1b - Restart auto-negotiation
8 FD	Full Duplex Duplex mode. Read-only 0b - Reserved 1b - Full duplex
7 COL_TEST	Collision test Collision test. Read-only 0b - Disable COL signal test 1b - Reserved
6 SPEED_SEL1	Speed selection (MSB) Speed selection (MSB) See SPEED_SEL0
5-0 —	Reserved

28.6.4.3 SGMII Status (SGMII_SR)

28.6.4.3.1 Offset

Register	Offset
SGMII_SR	1h

28.6.4.3.2 Function

The SGMII Status Register contains status bits on the operation of the PCS.

28.6.4.3.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ST_100BASET4	FD_100BASEX	HD_100BASEX	FD_10MBPS	HD_10MBPS	FD_100BASET2	HD_100BASET2	EXT_STAT	Reserved		AN_COMP	REM_FAULT	AN_ABIL	LINK_STAT	JAB_DET	EXT_CAP
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1

28.6.4.3.4 Fields

Field	Function
15 ST_100BASET4	100Base-T4 Read Only bit set to 0 to indicate that the PCS does not support 100Base-T4 operation
14 FD_100BASEX	100Base-X Full Duplex Read Only bit set to 0 to indicate that the PCS does not support 100Base-X operation
13 HD_100BASEX	100Base-X Half Duplex Read Only bit set to 0 to indicate that the PCS does not support 100Base-X operation
12 FD_10MBPS	10Mbps Full Duplex Read Only bit set to 0 to indicate that the PCS does not support 10Mbps operation
11 HD_10MBPS	10Mbps Half Duplex Read Only bit set to 0 to indicate that the PCS does not support 10Mbps operation
10 FD_100BASET2	100Base-T2 Full Duplex Read Only bit set to 0 to indicate that the PCS does not support 100Base-T2 operation.
9 HD_100BASET2	100Base-T2 Half Duplex Read Only bit set to 0 to indicate that the PCS does not support 100Base-T2 operation.
8 EXT_STAT	Extended Status Read Only bit always set to 0 to indicate that the PCS does not implement an extended status register.
7-6 —	Reserved
5 AN_COMP	AN Complete Auto-negotiation complete. Read Only Bit

Table continues on the next page...

Field	Function
	0b - The Auto Negotiation process is not completed or Auto Negotiation is disabled 1b - The Auto Negotiation process is completed and that the Auto Negotiation control registers are valid.
4 REM_FAULT	Remote Fault Read Only Bit always set to 0. The PCS does not implement a PHY specific remote fault detection optional function.
3 AN_ABIL	AN Ability Auto Negotiation Ability. Read Only Bit set to „1. to indicate that the PCS supports Auto-Negotiation.
2 LINK_STAT	Link Status Link Status 0b - The link is not valid. If the link synchronization is lost a 0 is latched which is cleared only after a register read access 1b - This link is valid.
1 JAB_DET	Jabber Detection Read Only bit always set to 0, the Core does not support the optional Jabber detection function
0 EXT_CAP	Extended Capability Read Only bit set to „1. to indicate that the Core supports extended registers

28.6.4.4 SGMII PHY Identifier Upper (SGMII_PHY_ID_H)

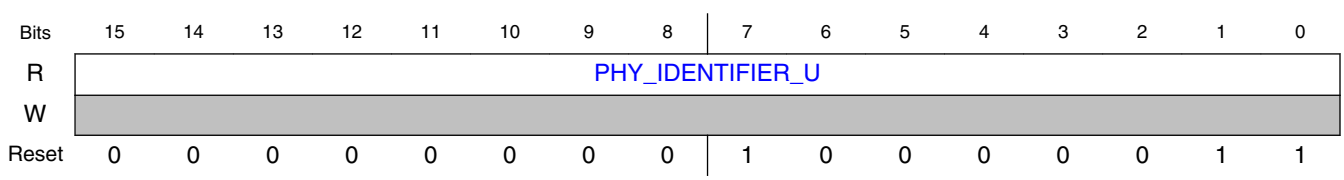
28.6.4.4.1 Offset

Register	Offset
SGMII_PHY_ID_H	2h

28.6.4.4.2 Function

The SGMII PHY Identifier Upper Register contains the upper half of the 32-bit PHY Identifier.

28.6.4.4.3 Diagram



28.6.4.4.4 Fields

Field	Function
15-0 PHY_IDENTIFIER_U	PHY Identifier Upper PHY Identifier Upper: OUI[3:18]

28.6.4.5 SGMII PHY Identifier Lower (SGMII_PHY_ID_L)

28.6.4.5.1 Offset

Register	Offset
SGMII_PHY_ID_L	3h

28.6.4.5.2 Function

The SGMII PHY Identifier Lower Register contains the lower half of the 32-bit PHY Identifier.

28.6.4.5.3 Diagram

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

28.6.4.5.4 Fields

Field	Function
15-0 PHY_IDENTIFIER_L	PHY Identifier Lower PHY Identifier Lower: 15:10 OUI[19:24] 9:4 Manufacturer's Model Number 3:0 Revision Number

28.6.4.6 SGMII Device Ability for 1000Base-X (SGMII_DEV_ABIL_1 KBX)

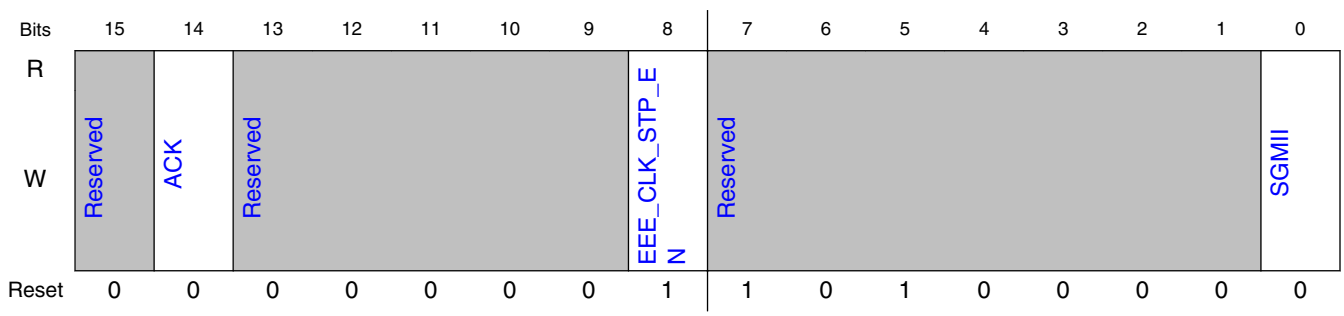
28.6.4.6.1 Offset

Register	Offset
SGMII_DEV_ABIL_1KBX	4h

28.6.4.6.2 Function

The SGMII Device Ability Register contains the control bits used to advertise the device abilities to the Link Partner during auto-negotiation for 1000Base-X mode.

28.6.4.6.3 Diagram



28.6.4.6.4 Fields

Field	Function
15 —	Reserved
14 ACK	Ack Read only bit set to 1 when device has received three consecutive matching ability values from the link partner
13-9 —	Reserved. Should be set to 0
8 EEE_CLK_STP_EN	EEE Clock Stop Enable EEE Clock Stop Enable

Table continues on the next page...

Field	Function
	Should be set to 0 before SGMII Auto-Negotiation enable/restart, as this device does not support EEE clock stop. 0b - EEE Clock Stop Disabled 1b - EEE Clock Stop Enabled
7-1 —	Reserved. Should be set to 0
0 SGMII	SGMII SGMII mode. Should be set to 1.

28.6.4.7 SGMII Device Ability for SGMII (SGMII_DEV_ABIL_SGMII)

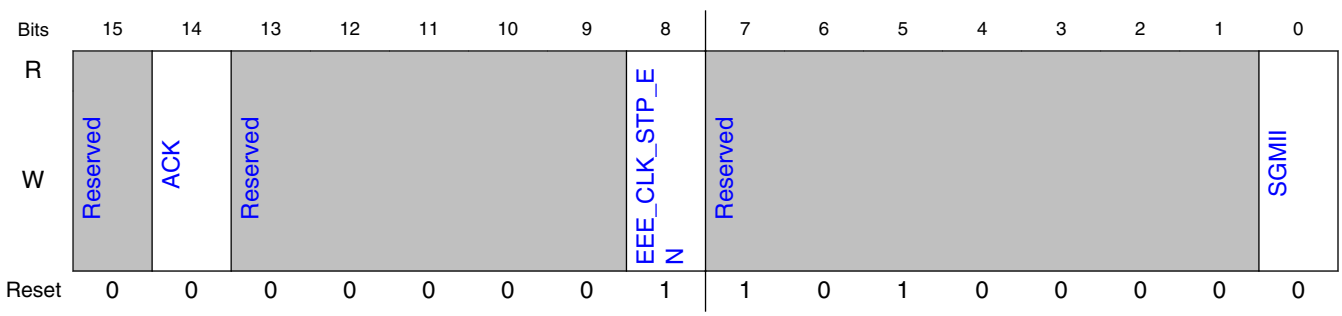
28.6.4.7.1 Offset

Register	Offset
SGMII_DEV_ABIL_SGMII	4h

28.6.4.7.2 Function

The SGMII Device Ability Register contains the control bits used to advertise the device abilities to the Link Partner during auto-negotiation for SGMII mode.

28.6.4.7.3 Diagram



28.6.4.7.4 Fields

Field	Function
15	Reserved

Table continues on the next page...

Field	Function
—	
14 ACK	Ack Read only bit set to 1 when device has received three consecutive matching ability values from the link partner
13-9 —	Reserved. Should be set to 0
8 EEE_CLK_STP_EN	EEE Clock Stop Enable EEE Clock Stop Enable Should be set to 0 before SGMII Auto-Negotiation enable/restart, as this device does not support EEE clock stop. 0b - EEE Clock Stop Disabled 1b - EEE Clock Stop Enabled
7-1 —	Reserved. Should be set to 0
0 SGMII	SGMII SGMII mode. Should be set to 1.

28.6.4.8 SGMII Partner Ability for 1000Base-X (SGMII_LP_DEV_ABIL_1KBX)

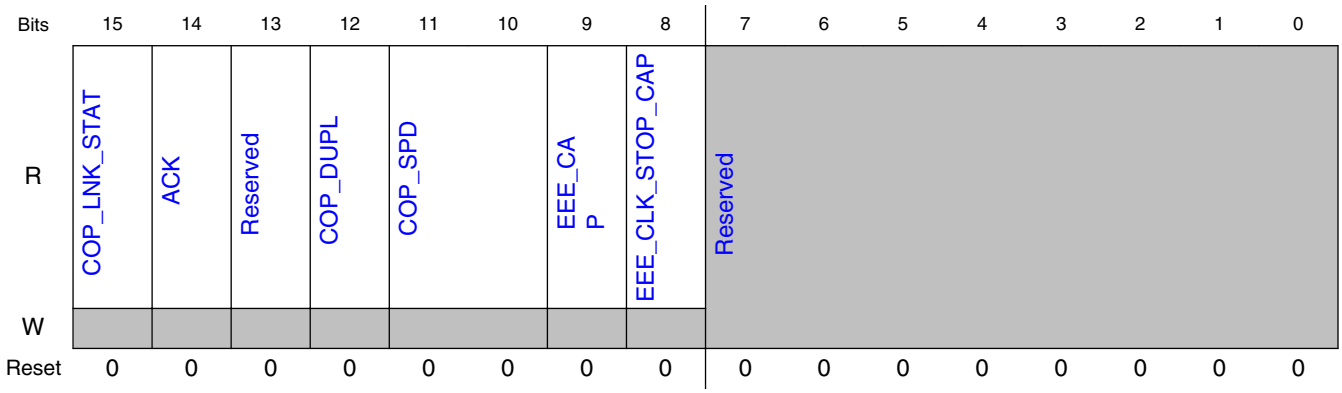
28.6.4.8.1 Offset

Register	Offset
SGMII_LP_DEV_ABIL_1KBX	5h

28.6.4.8.2 Function

The SGMII Partner Ability Register contains the capability status of the 1000Base-X link partner.

28.6.4.8.3 Diagram



28.6.4.8.4 Fields

Field	Function
15 COP_LNK_STAT	Copper Link Status Read only bit, used by the SGMII PHY to advertise the Link Partner Copper status: 0b - Copper interface link is down 1b - Copper interface link is up
14 ACK	Ack Read only bit set to 1. when the Link Partner Copper Interface advertises that it has that received three consecutive matching ability values from the device
13 —	Always 0
12 COP_DUPL	Copper Duplex Read only bit, used by the SGMII PHY to advertise the Link Copper duplex capability: 0b - Copper Interface resolved to Half-Duplex 1b - Copper Interface resolved to Full-Duplex
11-10 COP_SPD	Copper Speed Read only bits, used by the SGMII PHY to advertise the Link Partner Copper interface speed 00b - Copper Interface Speed is 10Mbps 01b - Copper Interface Speed is 100Mbps 10b - Copper Interface Speed is Gigabit 11b - Reserved
9 EEE_CAP	EEE Capability EEE Capability 0b - EEE not supported 1b - EEE supported
8 EEE_CLK_STOP_CAP	EEE Clock Stop Capability EEE Clock Stop Capability 0b - EEE Clock Stop not supported 1b - EEE Clock Stop supported

Table continues on the next page...

Field	Function
7-0 —	Reserved

28.6.4.9 SGMII Partner Ability for SGMII (SGMII_LP_DEV_ABIL_SGMII)

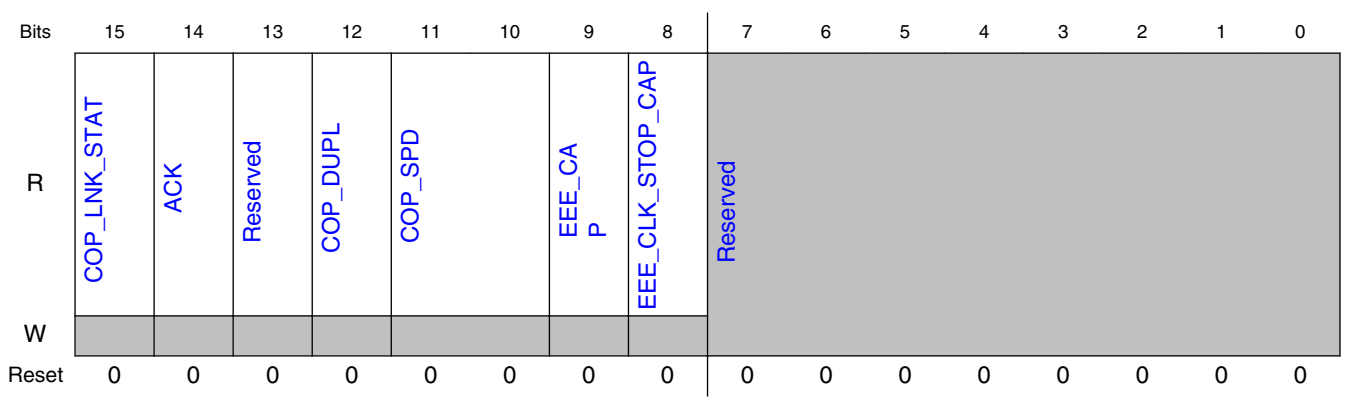
28.6.4.9.1 Offset

Register	Offset
SGMII_LP_DEV_ABIL_SGMII	5h

28.6.4.9.2 Function

The SGMII Partner Ability Register contains the capability status of the SGMII link partner.

28.6.4.9.3 Diagram



28.6.4.9.4 Fields

Field	Function
15 COP_LNK_STAT	Copper Link Status Read only bit, used by the SGMII PHY to advertise the Link Partner Copper status: 0b - Copper interface link is down

Table continues on the next page...

Field	Function
	1b - Copper interface link is up
14 ACK	Ack Read only bit set to 1. when the Link Partner Copper Interface advertises that it has that received three consecutive matching ability values from the device
13 —	Always 0
12 COP_DUPL	Copper Duplex Read only bit, used by the SGMII PHY to advertise the Link Copper duplex capability: 0b - Copper Interface resolved to Half-Duplex 1b - Copper Interface resolved to Full-Duplex
11-10 COP_SPD	Copper Speed Read only bits, used by the SGMII PHY to advertise the Link Partner Copper interface speed 00b - Copper Interface Speed is 10Mbps 01b - Copper Interface Speed is 100Mbps 10b - Copper Interface Speed is Gigabit 11b - Reserved
9 EEE_CAP	EEE Capability EEE Capability 0b - EEE not supported 1b - EEE supported
8 EEE_CLK_STOP_CAP	EEE Clock Stop Capability EEE Clock Stop Capability 0b - EEE Clock Stop not supported 1b - EEE Clock Stop supported
7-0 —	Reserved

28.6.4.10 SGMII AN Expansion (SGMII_AN_EXP)

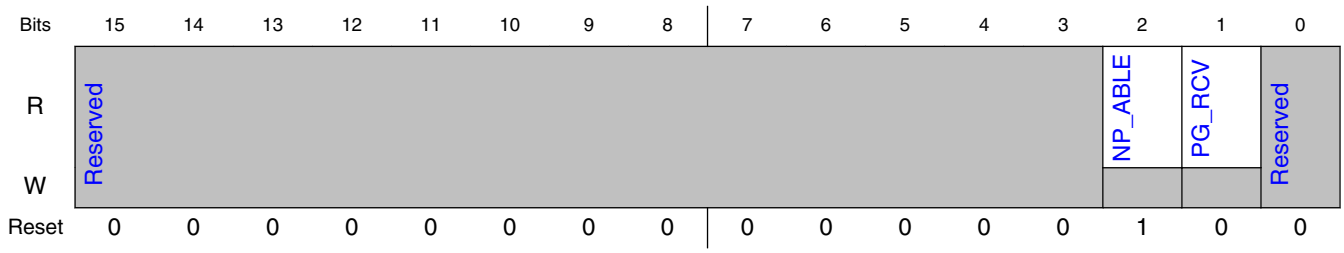
28.6.4.10.1 Offset

Register	Offset
SGMII_AN_EXP	6h

28.6.4.10.2 Function

The SGMII AN Expansion Register contains status bits indicating the PCS next page auto-negotiation capability and status.

28.6.4.10.3 Diagram



28.6.4.10.4 Fields

Field	Function
15-3 —	Reserved
2 NP_ABLE	Next Page Able Read Only bit set to 1 to indicate the PCS does support the Next Page function
1 PG_RCV	Page Received Set to 1 to indicate that a new page has been received with new partner ability available in the PCS register PARTNER_ABILITY. The bit is set to 0 (Reset value) when the system management agent performs a read access.
0 —	Reserved

28.6.4.11 SGMII Next Page Transmit (SGMII_NP_TX)

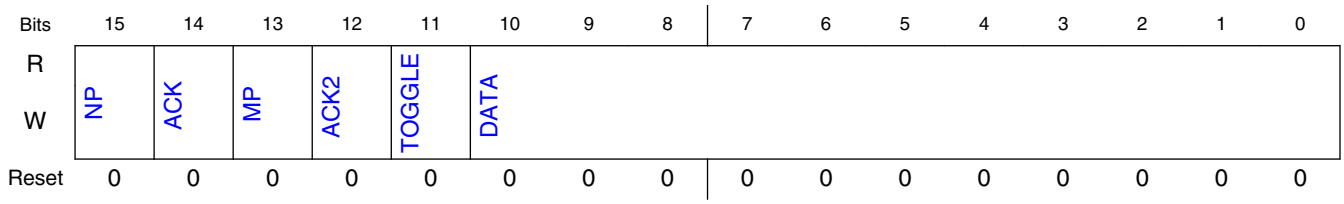
28.6.4.11.1 Offset

Register	Offset
SGMII_NP_TX	7h

28.6.4.11.2 Function

The SGMII NP TX Register contains next page data to transfer to the remote devide. Writing to this register initiates a next page exchange (sets mr_np_loaded variable).

28.6.4.11.3 Diagram



28.6.4.11.4 Fields

Field	Function
15 NP	Next Page Next page indication. Indicates if additional next pages will follow (1) or this is the last page (0).
14 ACK	Ack Acknowledge bit used by the autoneg function during message transfer. It has no meaning to the application and should be written with 0 always and ignored on read.
13 MP	Message Page Message page (1) or unformatted page (0) format.
12 ACK2	Ack 2 Application level acknowledge to indicate acceptance of a message. Use is defined by application layer.
11 TOGGLE	Toggle The toggle bit is used by the auto-negotiation function to keep track of message exchange. It has no meaning to the application and should be written with 0 always and ignored on read.
10-0 DATA	Data 11 bits of data which can either be a message-id (if MP bit 13=1) or unformatted data (if MP bit 13=0). See IEEE 802.3 Annex 28C for message page encodings

28.6.4.12 SGMII LP Next Page Receive (SGMII_NP_RX)

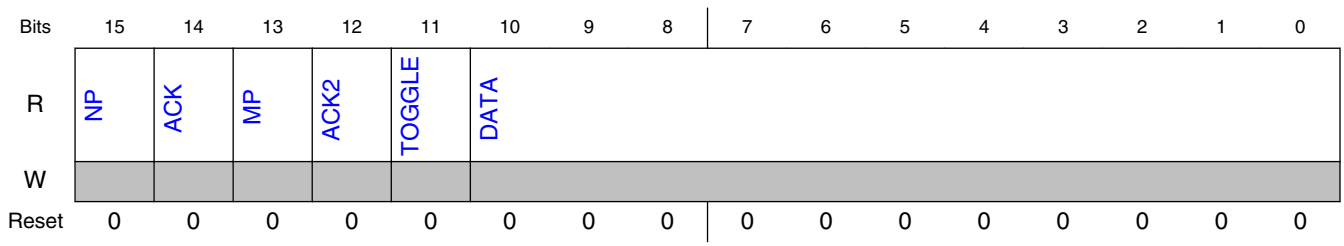
28.6.4.12.1 Offset

Register	Offset
SGMII_NP_RX	8h

28.6.4.12.2 Function

The SGMII NP RX Register contains the next page data received from the remote device during the latest next page exchange. The value is overridden with every new next page. Next page transfers are controlled by the application.

28.6.4.12.3 Diagram



28.6.4.12.4 Fields

Field	Function
15 NP	Next Page Next page indication. Indicates if additional next pages will follow (1) or this is the last page (0).
14 ACK	Ack Acknowledge bit used by the autoneg function during message transfer. It has no meaning to the application and should be written with 0 always and ignored on read.
13 MP	Message Page Message page (1) or unformatted page (0) format.
12 ACK2	Ack 2 Application level acknowledge to indicate acceptance of a message. Use is defined by application layer.
11 TOGGLE	Toggle The toggle bit is used by the auto-negotiation function to keep track of message exchange. It has no meaning to the application and should be written with 0 always and ignored on read.
10-0 DATA	Data 11 bits of data which can either be a message-id (if MP bit 13=1) or unformatted data (if MP bit 13=0). See IEEE 802.3 Annex 28C for message page encodings

28.6.4.13 SGMII Extended Status (SGMII_XTND_STAT)

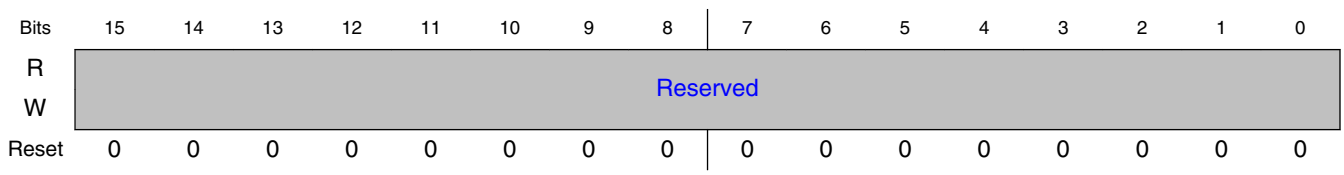
28.6.4.13.1 Offset

Register	Offset
SGMII_XTND_STAT	Fh

28.6.4.13.2 Function

The SGMII Extended Status Register is reserved, as this device does not implement the optional extended status registers.

28.6.4.13.3 Diagram



28.6.4.13.4 Fields

Field	Function
15-0 —	Reserved. Always 0

28.6.4.14 SGMII Scratch (SGMII_SCRATCH)

28.6.4.14.1 Offset

Register	Offset
SGMII_SCRATCH	10h

28.6.4.14.2 Function

The SGMII Scratch Register provides a memory location available to test register read and write operations.

28.6.4.14.3 Diagram

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

28.6.4.14.4 Fields

Field	Function
15-0 SCRATCH	Scratch Scratch field.

28.6.4.15 SGMII Design Revision (SGMII_REV)

28.6.4.15.1 Offset

Register	Offset
SGMII_REV	11h

28.6.4.15.2 Function

The SGMII Revision Register contains revision information for the PCS.

28.6.4.15.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								IP_MJ				IP_MN			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

28.6.4.15.4 Fields

Field	Function
15-8	Reserved

Table continues on the next page...

Field	Function
—	
7-4 IP_MJ	Major Revision Major revision
3-0 IP_MN	Minor Revision Minor revision

28.6.4.16 SGMII Link Timer Lower (SGMII_LINK_TMR_L)

28.6.4.16.1 Offset

Register	Offset
SGMII_LINK_TMR_L	12h

28.6.4.16.2 Function

The SGMII Link Timer Register Lower contains bits 15:1 of the link timer value.

28.6.4.16.3 Diagram

28.6.4.16.4 Fields

Field	Function
15-1 LINK_TMR_L	Link Timer Lower Link timer[15:1]
0 —	Reserved

28.6.4.17 SGMII Link Timer Upper (SGMII_LINK_TMR_H)

28.6.4.17.1 Offset

Register	Offset
SGMII_LINK_TMR_H	13h

28.6.4.17.2 Function

The SGMII Link Timer Register Upper contains bits 20:16 of the link timer value.

28.6.4.17.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											LINK_TMR_U				
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1

28.6.4.17.4 Fields

Field	Function
15-5 —	Reserved
4-0 LINK_TMR_U	Link Timer Upper Link timer[20:16]

28.6.4.18 SGMII IF Mode (SGMII_IF_MODE)

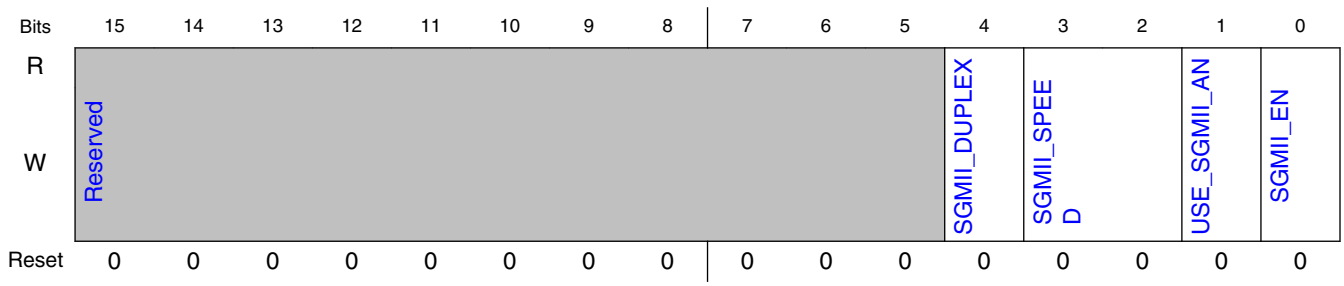
28.6.4.18.1 Offset

Register	Offset
SGMII_IF_MODE	14h

28.6.4.18.2 Function

The SGMII IF Mode Register contains control bits to set the interface mode.

28.6.4.18.3 Diagram



28.6.4.18.4 Fields

Field	Function
15-5 —	Reserved
4 SGMII_DUPLEX	SGMII Duplex SGMII Duplex Mode: Bit ignored when SGMII_EN is set to 0 or USE_SGMII_AN is set to 1. 0b - Full duplex enabled (default)
3-2 SGMII_SPEED	SGMII Speed SGMII Speed. When the PCS operates in SGMII mode (SGMII_EN set to 1) and is programmed not to be automatically configured (USE_SGMII_AN set to 0), sets the PCS speed of operation: Bits ignored when SGMII_EN is set to 0 or USE_SGMII_AN is set to 1. 00b - 10Mbps 01b - 100Mbps 10b - Gigabit 11b - Reserved
1 USE_SGMII_AN	Use SGMII Enable Use the SGMII Auto-Negotiation Results to Program the PCS Speed. When set to 0 (Reset Value), the PCS operation should be programmed with the register bit SGMII_SPEED and SGMII_DUPLEX. When set to 1, the PCS operation is automatically programmed with the Partner abilities advertised during Auto-Negotiation. Ignored when SGMII_EN is set to 0.
0 SGMII_EN	SGMII Enable SGMII Mode Enable. When set to '0' (Reset Value), the PCS operates in standard 1000Base-X Gigabit mode, when set to '1', the PCS operates in SGMII Mode

28.7 Initialization/Application Information

28.7.1 Initialization

All initialization necessary for the SerDes module to start operation is done automatically based on RCW, with the exception of the following protocols:

28.7.1.1 1G SGMII

After initializing the MAC, and prior to initiating any SGMII traffic, perform the following SGMII protocol initialization:

1. SGMII IF Mode register:
   - SGMII_EN=1
   - USE_SGMII_AN=1 if SGMII auto-negotiation is desired
   - SGMII_SPEED (if USE_SGMII_AN=0)
   - SGMII_DUPLEX (if USE_SGMII_AN=0)
2. SGMII Device Ability Register:
   - EEE_CLK_STP_EN=0
   - SGMII=1

To enable SGMII auto-negotiation, perform the following sequence:

1. SGMII Link Timer Register Upper:
   - LINK_TMR_U=0h03
2. SGMII Link Timer Register Lower:
   - LINK_TMR_L=0h06A0
3. SGMII Control Register:
   - AN_EN=1
   - RESTART_AN=1

   Note: Half duplex is not supported.

28.7.1.2 2.5G SGMII

All default SerDes settings are for 1G SGMII rather than 2.5G SGMII. The following settings need to be updated for 2.5G SGMII, following the procedure in Lane Reset and Reconfiguration, prior to initializing the SGMII PCS:

- LNmGCR1[REIDL_TH]
- LNmGCR1[REIDL_EX_SEL]
- LNmGCR1[REIDL_ET_MSB]

- LNmGCR1[ISLEW_RCTL]
- LNmGCR1[OSLEW_RCTL]
- LNmRECR0[GK2OVD]
- LNmRECR0[GK3OVD]
- LNmRECR0[GK2OVD_EN]
- LNmRECR0[GK3OVD_EN]
- LNmTECR0[TEQ_TYPE]
- LNmTECR0[RATIO_PST1Q]
- LNmTECR0[AMP_RED]

See General Control Register 1 - Lane a (LNAGCR1 - LNDGCR1), Receive Equalization Control Register 0 - Lane a (LNARECR0 - LNDRECR0) and Transmit Equalization Control Register 0 - Lane a (LNATECR0 - LNDTECR0) for details.

The PCS initialization sequence for 2.5G SGMII is the same as for 1G SGMII with auto-negotiation disabled (see 1G SGMII). Auto-negotiation is not supported for 2.5G SGMII.

28.7.1.3 1000Base-KX

All default SerDes settings are for 1G SGMII rather than 1000Base-KX. The following settings need to be updated for 1000Base-KX, following the procedure in Lane Reset and Reconfiguration, prior to performing 1000Base-KX auto-negotiation:

- LNmGCR1[REIDL_TH]
- LNmGCR1[REIDL_EX_SEL]
- LNmGCR1[REIDL_ET_MSB]
- LNmTECR0[AMP_RED]

See General Control Register 1 - Lane a (LNAGCR1 - LNDGCR1) and Transmit Equalization Control Register 0 - Lane a (LNATECR0 - LNDTECR0) for details.

After initializing the MAC and SERDES lane(s), and prior to initiating any 1000Base-KX traffic, perform the following 1000Base-KX protocol initialization:

1. Enable the 1000Base-KX AN reference clock by setting PLLnCR0[DLYDIV_SEL]=01, for the PLLn that is the clock source for the SGMII PCS (see TPLL_LES in General Control Register 0 - Lane a (LNAGCR0 - LNDGCR0)).
2. Enable the 1000Base-KX AN module by setting PCCR8[SGMIIp_KX]=1, for each SGMIIp that will run in 1000Base-KX rather than SGMII mode.
3. Initialize SGMII IF Mode register to 0x0008:
   - SGMII_EN=0

- USE_SGMII_AN=0
- SGMII_SPEED=10

4. Initialize 1000Base-KX (Clause 45) AN Millisecond Counter (simulation speedup only) to 0x0002
5. Initialize 1000Base-KX (Clause 45) AN Advertisement Register 1:
   - TX_NONCE=unique value per device
6. Read 1000Base-KX (Clause 45) AN Status Register to clear any previous status
7. Initialize 1000Base-KX (Clause 45) AN Control Register to 0x1200:
   - AN_ENAB=1
   - RST_AN=1

28.7.2 Unused Lanes

Unused lanes should be powered down to save power and avoid noise on adjacent lanes.

Power down the Rx portion of the lanes by setting LNmGCR0[RX_PD]=1 and LNmGCR0[RRST_B]=0.

Power down the Tx portion of the lanes by setting LNmGCR0[TX_PD]=1 and LNmGCR0[TRST_B]=0.

The Rx and Tx portions of the lanes may be independently powered down for uni-directional lanes or links.

28.7.3 Soft Reset and Reconfiguring Procedures

28.7.3.1 Lane Reset and Reconfiguration

To reconfigure a lane (change settings including clock divider or PLL select), perform the following sequence:

1. Put the lane(s) into reset by setting LNmGCR0[TRST_B]=0 and LNmGCR0[RRST_B]=0.
2. Wait at least 50 ns
3. Change the desired per-lane settings
4. Wait at least 120 ns
5. Take the lane(s) out of reset by setting LNmGCR0[TRST_B]=1 and LNmGCR0[RRST_B]=1

Note that if the lanes being reconfigured are grouped for multi-lane or synchronous mode, then the master source clock lane for the group must have TRST_B set to 1 after all the other lanes in the group. The master source clock lane of the group is indicated by LNmGCR0[1STLANE]=1. All lanes p<m (if LNmGCR1[TRSTDIR]=0) or p>m (if LNmGCR1[TRSTDIR]=1) of the master source clock lane until the next lane with LNmGCR0[1STLANE]=1, or the end of the SerDes, are grouped with the master source clock lane. All grouped lanes must have the same setting of TRSTDIR.

It is recommended to disable any controller connected to a lane being reconfigured prior to starting the reconfiguration sequence.

28.7.3.2 Lane Enable After Powerdown

To enable a previously powered down lane, set LNmGCR0[nX_PD]=0 (n=R or T), wait 15 us, then set LNmGCR0[nRST]=1.

Note that if a Tx lane m with LNmGCR0[1STLANE]=1 (master Tx clock lane) is powered down or reset, all Tx lanes p<m (if LNmGCR1[TRSTDIR]=0), or all Tx lanes p>m (if LNmGCR1[TRSTDIR]=1), cannot be used.

28.7.3.3 PLL Reset and Reconfiguration

To reconfigure a PLL, perform the following sequence:

1. Disable all lanes using the PLL to be reconfigured by setting PLLnRSTCTL[SDRST_B]=0
2. Wait at least 50 ns
3. Disable the PLL by setting PLLnRSTCTL[SDEN]=0 and PLLnRSTCTL[PLLRST_B]=0
4. Wait at least 100 ns
5. Change the desired per-PLL settings (VCO frequency, refclk frequency, etc.).
6. Reset the PLL by setting PLLnRSTCTL[RSTREQ]=1
7. Set PLLnRSTCTL[SDEN]=1, PLLnRSTCTL[PLLRST_B]=1, and PLLnRSTCTL[SDRST_B]=1
    - Note this step is not to be combined with setting RSTREQ=1

Note: lane reconfiguration may also be performed while the PLL is disabled by following the first three steps of the lane reset sequence in Lane Reset and Reconfiguration after changing the per-PLL settings, before setting RSTREQ=1, and the last step of the lane reset sequence after the last step of the above PLL reset sequence.

It is recommended to disable any controller connected to a lane selecting the PLL being reconfigured prior to starting the reconfiguration sequence. In the case of PCI Express, the controllers must be disabled to prevent auto-negotiation to 8Gbaud, which can include autonomous PLL reset and reconfiguration.

28.7.4 Quiesce Sequences for System Sleep

To quiece the SerDes module in preparation for System Sleep, the user must first quiesce all controllers and disable transmission/reception of packets.

Chapter 29
Secure Boot and Trust Architecture 2.1

29.1 The SecMon module as implemented on the chip

This section provides details about how the SecMon module is implemented on the chip.

29.1.1 LS1012A SecMon module special consideration

The SecMon module implements the following parameter settings in the chip:

Table 29-1. LS1012A SecMon parameter settings

SecMon parameters	LS1012A parameter value
LP mode support	No

29.2 Trust architecture overview

The Trust Architecture 2.1 is a set of hardware and software techniques designed to support trusted boot and maintenance of the trusted environment during runtime. The Trust architecture is based on capabilities and intellectual property developed by NXP and deployed in NXP QorIQ communications processor chips, with some functions performed by Arm® TrustZone® module. The Trust Architecture 2.1 is distinguished from previous versions of the QorIQ platform's trust architecture through the complementary inclusion of the Arm TrustZone, as appropriate to the needs of secure network and access infrastructure.

The Trust architecture is implemented via a highly integrated combination of trusted software and trusted hardware. This chapter provides a brief overview of the Trust Architecture 2.1 as a whole, followed by detailed information on the Security Fuse Processor (SFP) and Security Monitor (SecMon). Other logic blocks have a role to play

in the trust architecture; however, as these other blocks have significant additional non-trust architecture functionality, they are documented in their own chapters. The primary differences between the Trust Architecture 2.1, and the previous versions of the QorIQ trust architecture as listed below:

- Introduction of Arm TrustZone ‘Secure World’ and ‘Non-Secure World’ concepts
  - Arm ISA general purpose processors rather than Power ISA
- Manufacturing protection support

29.3 Terminology used in this chapter

The addition of Arm general purpose processors with TrustZone to QorIQ LS series processors has the potential to create overlapping terminology and confusion over definitions. For the purposes of this chapter, the following definitions apply:

- TrustZone – Arm’s name for the processor and chip architecture for creating a Trusted Execution Environment.
- QorIQ Trust Architecture - The name for the QorIQ product line’s architecture for achieving a Trusted Partition.
- QorIQ LS Trust Architecture - The name for an enhanced version of QorIQ Trust Architecture which co-exists with and complements Arm TrustZone.
- Software entity - generic term for a program or set of programs with shared purpose and relative independence from other programs.
- Software partition – One or more software entities and the resources they share with each other, but not with other software partitions.
- Isolation - separation of resources belonging to one software partition from all other software partitions. In functional safety applications, isolation means one software partition is unable to affect the operation of other software entities. That is, transactions to system main memory generated by one software entity can’t change the latency of another software entity’s access to system main memory (or to any resource).
  - True isolation in a multi-core chip is extremely difficult to achieve due to a large number of shared physical resources.
- Separation, Strong Partitioning – Like isolation, but with a narrower interpretation of what is being isolated. Software partitions cannot directly access each other’s private resources; however they may indirectly affect each other’s operations through system congestion or other physical effects.
- Isolated execution environment- a set of computing resources restricted (with enforcement) to the use of a single software partition.

    - The isolated execution environment may be a cage for untrusted software or a walled fortress for trusted software.
    - Despite the name, the execution environment would be more accurately described as separated or strongly partitioned than isolated in the functional safety sense.
- Trusted Execution Environment - An isolated execution environment containing a software entity which has passed a verification process (secure boot) and is therefore trusted to perform special operations.
    - In Arm literature, a TEE contains a TrustZone RTOS, likely with at least one trusted application running on the TrustZone RTOS.
    - Some third party vendors of TrustZone RTOSs call their product a TEE.
- Trustlet - An application written to run on a TrustZone RTOS. Third party trustlets are written to expect a standardized set of computing resources, including 64KB of secure RAM.
- Secure World – In Arm literature, software running within the TEE is said to be running in Secure World. Secure World can also include non-CPU masters whose transactions are tagged as NS=0 (Non-Secure = 0, meaning not Non-Secure)
    - Software entities and non-CPU masters whose transactions are NS=0 are also said to run in TrustZone state. Thus TrustZone is frequently used synonymously with Secure World.
    - Some targets can also be in Secure World, in that they reject NS=1 (non-secure =yes) transactions.
    - All Secure World/TEE transactions are tagged with NS=0, allowing them to access:
        - TEE/Secure World only memory and other targets
        - NSW/Non-Secure World memory and other targets
    - Note: NS=0 is an additional qualifier; the TEE also has to be configured for access to the address range
- Non-Secure World- In Arm literature, software running outside the TEE is said to be running in Non-Secure World.
    - Non-Secure World also includes non-CPU masters whose transactions are tagged as NS=1 (Non-Secure = 1, meaning Non-Secure)
    - All NSW transactions are tagged with NS=1, allowing them to access:
        - NSW/Non-Secure World memory and other targets
    - There are no Non-Secure World only targets. Secure World can access Non-Secure World, but the reverse isn’t true.
    - Note: NS=0 is an additional qualifier. Secure World/TEE can be blocked from accessing a Non-Secure World target if the TEE isn’t configured to access the address range; however, nothing prevents Secure World/TEE from changing its own address range permissions.
    - In this chapter, Non-Secure World will be abbreviated to NSW.

- Trusted Partition- A term coined by NXP referring to a software entity which has been verified with secure boot, and isolated from other software entities by means of a hypervisor rather than an Arm TrustZone Monitor + TrustZone RTOS.
  - A Trusted Partition runs in Arm Non-Secure World, and relies on Trust Architecture capabilities beyond the scope of Arm TrustZone to achieve its trustworthiness.
  - The concept of a Trusted Partition exists in QorIQ Trust Architecture, however the need to distinguish types of trusted software only arises in QorIQ LS series products due to the existence of the Arm TEE/Secure World.
  - In this chapter, Trusted Partition will be abbreviated as TP.
- Non-Trusted Partition; will refer specifically to a software partition which is separated/strongly partitioned from other software partitions, but is not trusted.
  - Could be a ‘normal’ partition, or it could be malicious.
  - Runs in Arm Non-Secure World and has not been verified by secure boot.
  - In this chapter, Non-Trusted Partition will be abbreviated as NTP.
- Non-Secure state- the state of the Security Monitor when the device boots with the Intent to Secure fuse =0, or RCW[SB_EN]=0, and a non-fatal security violation (including signature verification failure) occurs.
  - In this state, non-trusted software is allowed to run, but it cannot request the SEC to use persistent or ephemeral secret keys.
- Trusted state - the state of the Security Monitor when the device boots with the Intent to Secure fuse or RCW[SB_EN]=1, and no security violations are detected.
  - In this state, trusted software (at least one trusted partition) is allowed to run, and any software with memory mapped access to the SEC can request the SEC to use persistent or ephemeral secret keys.
- Secure state - the state of the Security Monitor following the Trusted state. Secure state is generally equivalent to the Trusted state. At least one trusted partition is running.

29.3.1 Relationship of Trust Architecture to TrustZone

The terminology described above can best be understood with the help of Figure 29-1. As shown in the figure, TZ Secure World is a security domain within the overall Trust Architecture SoC. Trust Architecture provides hardware secure boot, hardware debug protection, external tamper detection, and device secrets which even TZ Secure World software isn’t able to access. TrustZone Secure World provides a Trusted Execution Environment for a Trustlet to run in, and using the NS transaction attribute, allows the TZ Secure World Trustlet exclusive access to regions of on-chip or off-chip memory, and control of the registers required to configure the exclusive access regions.

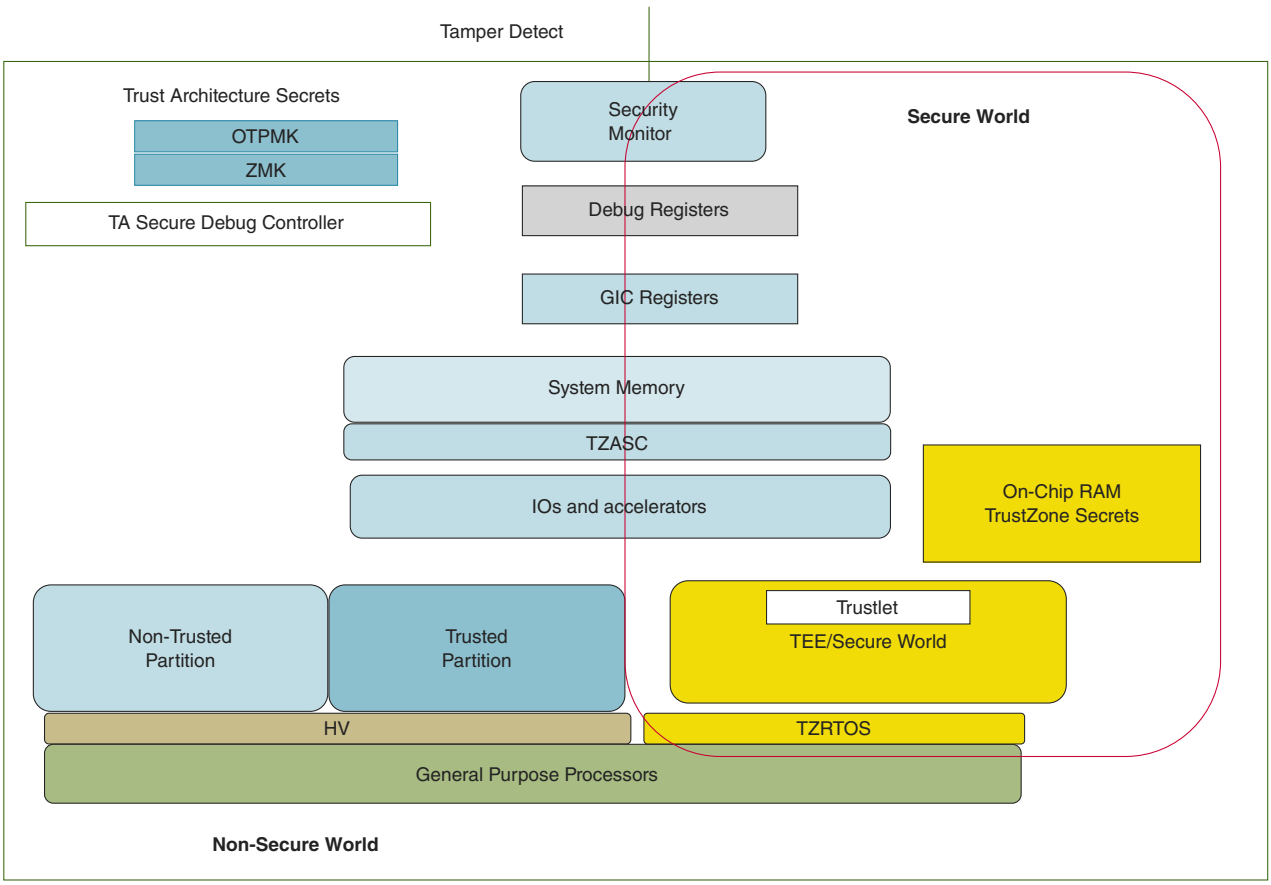


Figure 29-1. Relationship between Arm TrustZone and Trust Architecture

29.4 Objectives of Trust Architecture 2.1

A trusted system is a system that does what its developer and users expect it to do, and specifically does not do other things the developer and/or users would consider harmful.

In the context of the chip's implementation of the Trust Architecture, if developers properly leverage the hardware hooks in the chip, thcan 'trust' that the software they loaded into the system during manufacturing (or during authorized software updates) is the software that executes following system boot.

Once trusted software is in control, the developer can leverage additional Trust Architecture and TrustZone ™ features to keep the trusted code in control of the system and defend against the extraction of system secrets or introduction of malicious software.

The security mechanisms within the Trust Architecture allow users to define and enforce security policies. Examples of security policy violations that can be prevented or heavily mitigated by the Trust Architecture are listed below:

- Unauthorized modifications to developer software and system configuration information (device trees, certificates). Protection consists of both prevention and after the fact detection mechanisms
- Unauthorized exposure of system persistent secrets. These are secrets that are intended to persist between resets of the system. The Trust Architecture persistent secrets include the chip's One Time Programmable Master Key (OTPMK), optional Zeroizable Master Key (ZMK), and any code, factory installed private asymmetric, and pre-shared symmetric keys encrypted by the OTPMK or ZMK and stored to nonvolatile memory.
- Unauthorized exposure of system ephemeral secrets. These are secrets that are intended to be cleared by the system's next reset (or sooner). Trust architecture ephemeral secrets include the chip's Job Descriptor Key Encryption Keys (JDKEKs) and session keys negotiated during normal operation that are encrypted with a JDKEK (also known as, "Black Keys"). Secrets owned by the TEE and stored to Secure World private on-chip or off-chip private memory are also considered ephemeral.

While some developers consider security policy enforcement to be a critical feature of the chip, other developers may not. Consequently the Trust Architecture is disabled by default. Developers not implementing trust features can ignore their existence.

Developers who choose to leverage the Trust Architecture are not dependent on NXP to provision chips or sign code. NXP is not part of the system development or manufacturing chain of trust. Developer provisioning of chips is designed to be simple, with minimal impacts on manufacturing cycle times.

29.5 Trust Architecture as implemented on the chip

29.5.1 TrustZone (TZ) architecture in the Arm Cortex-core general purpose processors (GPPs)

The Arm GPPs support TrustZone in their CPU and cache architectures.

Arm GPPs with Arm security extensions (TrustZone) have multiple execution modes as shown in the figure below,

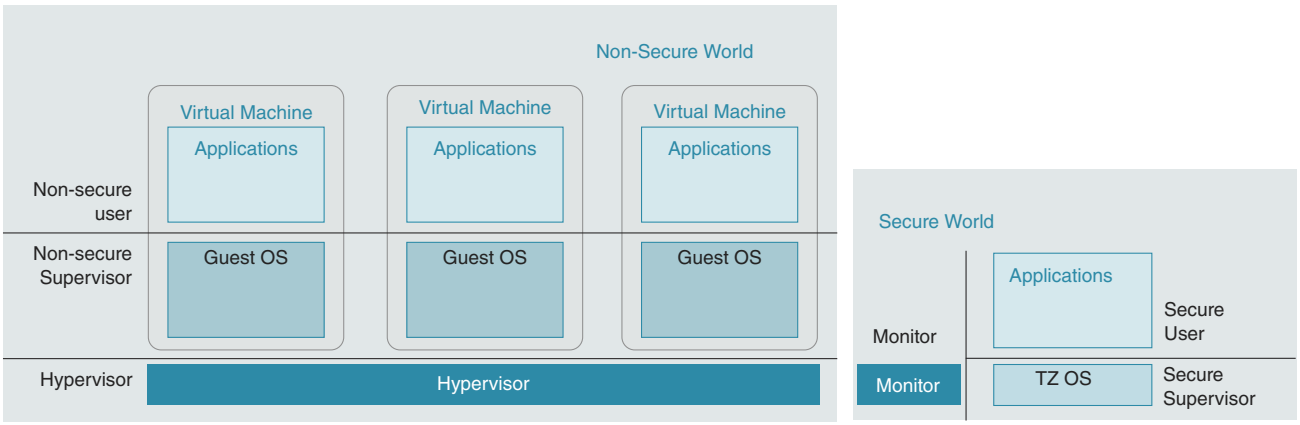


Figure 29-2. Arm TrustZone Non-Secure World and Secure World

As described in Trust architecture overview, the Arm GPP execution modes are divided into two “worlds”, secure and non-secure. Additional Arm virtualization extensions support virtualization in the non-secure world.

The secure world supports a trusted execution environment, with:

- A monitor mode dedicated to switching between worlds
- Separate user and supervisor modes for secure applications and OS
- Increasing levels of trust from user to supervisor
- Secure debug modes, with isolation from non-secure world debug modes

Monitor mode is used solely to mediate switching between secure and non-secure worlds. It should not be confused with any continuous or periodic platform monitoring function. Monitor mode has a dual persona: it acts within the secure world, but also has access to non-secure versions of CPU instructions and banked registers.

The secure world does not include a hypervisor mode. Instead, it is useful to think of the secure world as an extra virtual machine beyond the reach of the hypervisor.

The secure world is intended to provide strictly limited services requiring access to sensitive data or capabilities to the non-secure world via a well-controlled single point of entry.

29.5.1.1 TrustZone during boot

After reset, the Arm CPU begins executing in secure supervisor mode. The subsequent passage through different execution modes depends on platform requirements for the particular CPU:

- With no trusted execution or virtualization requirements, execution remains in the secure world forever, using the secure user and supervisor modes and ignoring the remainder.
- With only trusted execution requirements, the boot sequence launches the TrustZone OS, which partitions the system between secure and non-secure worlds before switching to the non-secure world to launch the non-secure OS, leaving the hypervisor mode inactive. Switch back to the secure world occur whenever secure services are requested or an interrupt configured to be taken by the secure world is triggered.
- With only virtualization requirements, a boot loader in secure supervisor mode ensures that all system resources are allocated to the non-secure world before switching to launch the hypervisor.
- With both trusted execution and virtualization requirements, the two preceding scenarios are merged, with the TrustZone OS taking precedence over the hypervisor.

29.5.1.2 TrustZone during runtime

In an SMP configuration, any CPU can be executing in any mode at any time. In particular, the secure world is not usually “pinned” to a specific CPU, and multiple CPUs could be executing in the secure world concurrently.

Transactions generated by the CPU carry the following flags:

- Instruction fetch. Note that many Arm instructions such as local indirect jumps generate both instruction fetches (for the opcode) and data fetches (for the jump vector), and that compilers and assemblers inevitably intersperse vectors and opcodes. This prevents protecting software confidentiality using “execute-only” privileges.
- Privileged. All the processor modes except non-secure user and secure user are considered privileged.
- Non-secure (NS). Non-secure world execution modes can generate only NS=1 transactions. Secure world transactions typically carry NS=0.

29.5.1.3 Arm core caches and TrustZone

Arm processor caches support:

- NS bit tagging of cache lines to isolate secure and non-secure instructions and data

The secure world can issue non-secure (NS=1) transactions when accessing memory shared with the non-secure world in order to maintain cache coherency.

Explicit cache maintenance operations driven by software preserve isolation:

- Commands issued from non-secure modes affect only non-secure cache lines
- Commands issued from secure user/privileged modes affect only secure cache lines
- Commands issued from monitor mode can affect both secure and non-secure cache lines

Autonomous cache replacement operations driven by the cache controller itself ensure availability. Cache lines are replaced according to the relevant policy (LRU or random) without regard to the line's NS flag: non-secure memory accesses can allocate cache lines previously allocated during secure memory accesses. This implies that non-secure software can indirectly cause secure memory accesses as the cache controller evicts a secure cache line, however the non-secure software has no control over the content or address of the secure memory access.

29.5.2 TrustZone Address Space Controller (TZASC)

The TZASC facilitates partitioning system memory into secure and non-secure memory regions. It supports the following features:

- 2, 4, or 8 region descriptors.
- Programmable starting address (0-modulo-32 KB or larger)
- Power-of-2 region sizes (32 KB,..., 4 GB)
- Ability to subdivide the region into 8 equal-sized segments. Capability to enable/disable the subsegments
- 4-bit access control flags with optional security "inversion". Secure {read, write}, nonsecure {read, write}
- Fixed priority of overlapping region descriptor hits
- Region 0 defined as "background region" as it maps the entire address space
- Ability to enable address speculation for improved system performance. Removes the added 1 cycle of latency on all transactions
- Programmable system response to an access control violation. Optional error termination on the bus cycle and/or assertion of a TZASC interrupt alert.
- Disabled or logically bypassed by programming the CSU_SA1 register
- Register content/fields can be boot-locked by programming the CSU_SA1 register.

For more information about the programming model of TZASC and its features, refer the Arm CoreLink™ TrustZone Address Space Controller TZC-380 chapter.

29.5.3 Central Security Unit

The Central Security Unit (CSU) allows secure world software to change the default access control policies of peripherals/bus slaves, determining which bus masters may access them. This allows peripherals to be separated into distinct security domains.

29.5.4 Platform Control

The platform control stores the status and address attributes of the blocked/failed transactions to the target IP modules whose access controls are defined by CSU register bits and also raise an interrupt and security violation to security monitor.

For more details, refer **Miscellaneous System Control Module (MSCM)**.

29.5.5 Arm generic interrupt controller (GIC)

The Generic interrupt controller (GIC) is a centralized resource for supporting and managing interrupts from peripherals to at least one general purpose Arm processor.

In the context of the Trust Architecture, the Arm GIC supports virtualized interrupts (interrupt delivered to a specific virtual machine) and to secure world. The GIC registers support configuration by TZ secure world, and prevents non-secure world software from disabling or clearing the interrupts of peripherals controlled by TZ secure world.

29.5.6 TrustZone Watchdog (TrustZone WDOG)

The TrustZone WDOG is an additional watchdog timer instance that cannot be programmed or deactivated from TZ non-secure world.. The TrustZone WDOG protects against TZ non-secure world software preventing a switch back to the Secure World, thereby starving security services of access to GPP resources. Once the TrustZone WDOG is activated, it must be serviced by TZ secure world software on a periodic basis. If servicing does not take place before the configured time-out, the TrustZone WDOG asserts a secure interrupt that forces a switch to the Secure World. If it is still not served, the TrustZone WDOG generates a hardware security violation to the Security Monitor (see Security monitor).

The TrustZone WDOG requires a non-gateable clock source, and should be capable of counting up to 256 seconds.

29.5.7 On-Chip RAM (OCRAM)

The chip incorporates two 64 KB on-chip RAMs. Both of the OCRAMs are configured as Secure World by default. OCRAM is used in all secure boot flows involving PBI.

In case of Trust Architecture security monitor hard fail, the OCRAM is zeroized automatically.

29.5.8 Secure debug controller

To enable device debug, the chip integrates a secure debug controller (SDC).

The secure debug controller (SDC) supports four levels of access.

- Open-External debug agents connected to the JTAG interfaces have full access to the memory space. Activation of debug is not considered a security violation, and if the Security Monitor is in trusted/secure state, it remains there. This setting is only appropriate in a lab environment.
- Conditionally closed without notification-External debug agents connected to the JTAG interfaces are prevented from accessing the memory space until the user passes a challenge/response sequence. The user must request a challenge operation, which causes the SDC to output a 64-bit value. The user then inputs a 64-bit response value that the SDC compares to a secret value burned into the SFP. If the comparison passes, the user is given full debug access, as in the open case. If the comparison fails, the SDC signals a security violation to the Security Monitor that triggers a state transition as described in Security monitor. OEMs configure the control fuses to operate in this mode if they want the ability to debug systems with One-Time Programmable Master Key (OTPMK), Zeroizable Master Key (ZMK), or key encryption key (KEK) usage enabled .
- Conditionally closed with notification-Operation is the same as conditionally closed without notification, however passing the challenge/response cycle still causes the SDC to signal a security violation to the Security Monitor. This security violation triggers a state transition, as described in Security monitor. So long as the OEM has not enabled hard fail, the system will continue to operate, however the OTPMK is locked out. Ephemeral system secrets are cleared. OEMs configure the control fuses to operation in this mode if they want to restrict the operations a technician could

initiate, but still want the system to continue to function at some level in order to gather debug information.

- Closed-attempts by external debug agents to access the memory space are always blocked, and are not reported as security violations. The JTAG interface can still be used for boundary scan physical interconnect testing.

The Secure Debug Controller can be configured with access to Secure World or Non-Secure World, with the default being Non-Secure World.

As shown in figure, gaining access to the SoC's memory space via the SDC isn't all-inclusive. The OTPMK, ZMK, and JDKEKs are never readable. Also, because the SDC operates as a non-secure world master by default, it can't read or write to Secure World.

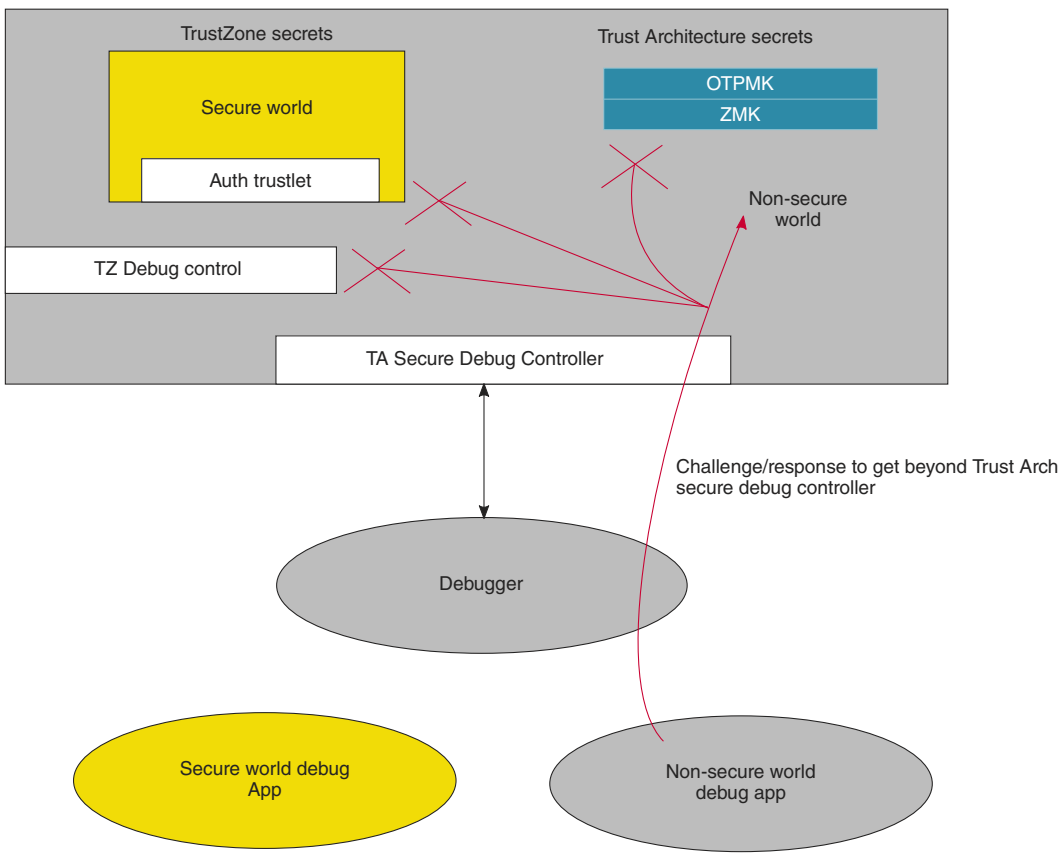


Figure 29-3. Secure Debug Controller's default access

To debug within Secure World, an external Secure World Debug Application has to authenticate itself to an Authentication Trustlet running in Secure World. If authentication is successful, the Trustlet changes the TrustZone Debug Control settings to allow the debugger access to Secure World, including Secure World secret values.

29.5.9 SEC 5.5

The primary function of the SEC module is to accelerate cryptographic operations. Depending on the device's security state, these cryptographic operations may use normal secret keys, such as those typically established for IPsec and SSL, or special persistent and ephemeral keys, including the fuse-based OTPMK, battery backed ZMK, or randomly initialized JDKEKs.

The SEC is capable of accessing both TZ secure World and TZ non-Secure World, with the access permissions defined at the Job Descriptor ring level. Jobs submitted to the SEC on a Job Descriptor ring configured for TZ secure world access can use ns=0 (secure world) for all bus transactions.

At the time OEM software (ESBC) begins execution, all JRs are owned by TZ non-secure world. Subsequently, TZ secure world can take ownership of a SEC Job Descriptor Ring by setting the TZ bit in the Job Ring’s JRICID register to 1. Once a Job Descriptor Ring is owned by TZ secure world, the SEC blocks for all accesses to SEC registers for that Job Descriptor Ring by TZ non-secure world.

In addition to supporting security-state-based usage of special secret keys, the SEC also supports a security violation detection function known as the run time integrity checker (RTIC). The RTIC leverages the SEC's cryptographic hashing capability to periodically check the integrity of designated sections of system memory.

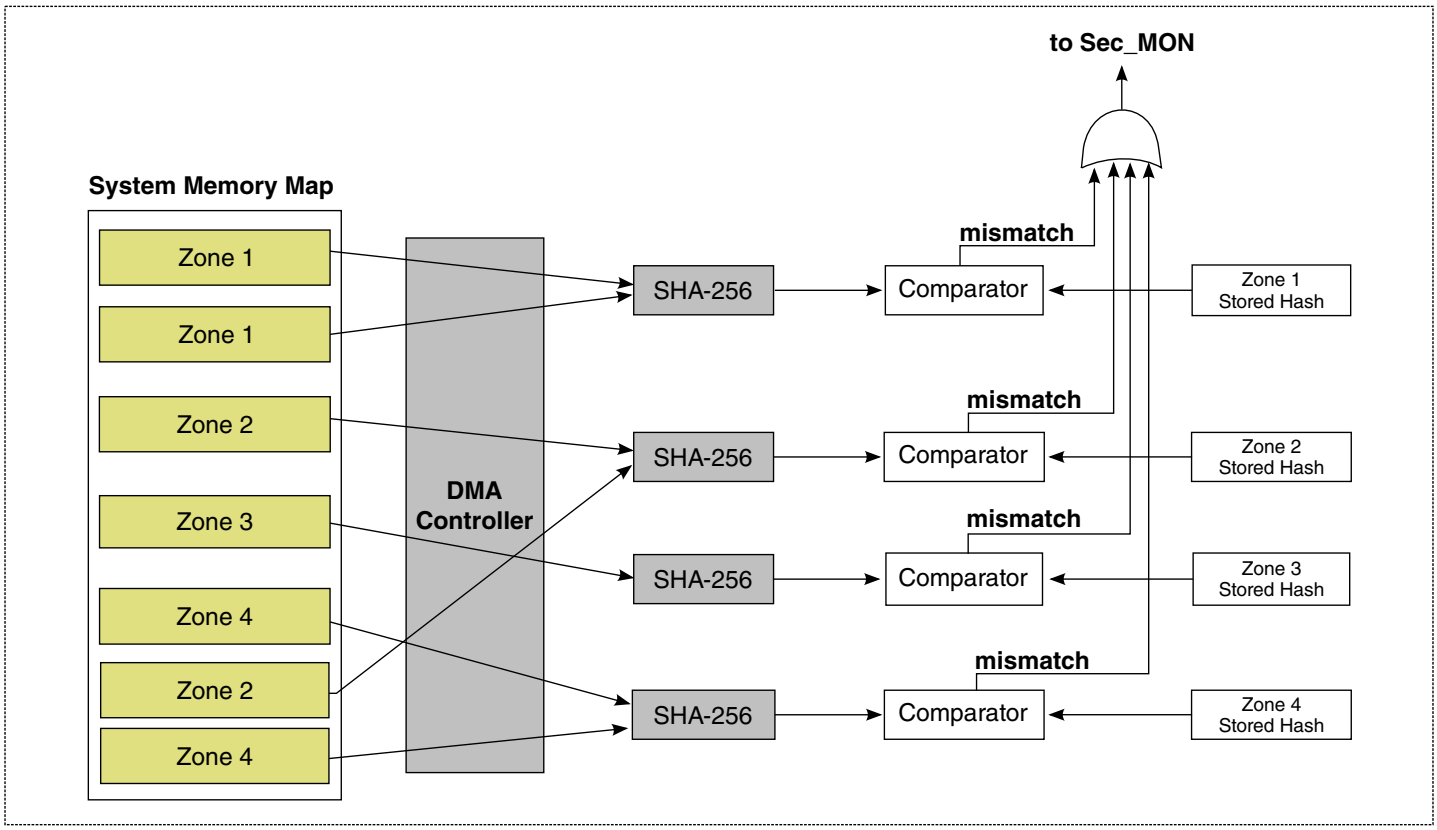


Figure 29-4. SEC module performing memory integrity checking

As shown in the figure above, the SEC RTIC can be configured to periodically calculate a SHA-256 or SHA-512 hash over up to four memory zones (each of which can be defined as one or two contiguous memory blocks). The newly calculated hash is compared against the hash of this memory that was generated when the RTIC was first configured. If the hash value does not match, system memory is considered to have changed outside of program control, and a security violation is reported to the Security Monitor.

The RTIC operates as a background task for the SEC until a configured timer expires. If the timer expires before the RTIC has completed an integrity check cycle, the SEC reports a security violation. The impact of RTIC operation on system level performance depends on the size of the memory zones to be checked, and the frequency of the integrity checks. All RTIC operations use TZ ns=0, allowing it to integrity check TZ secure world and TZ non-secure world memory space.

29.5.10 Internal boot ROM and ISBC

The Internal boot ROM (IBR) contains code known as the internal secure boot code (ISBC).

The ISBC is contained in an internal ROM to ensure that the code cannot be modified. The ISBC is deliberately simple, and its only responsibility is to validate a signature over the next code to execute (referred to as the external secure boot code-ESBC) using information contained in a header file on the ESBC and in the SFP. The secure boot process is explained further in Secure boot sequence.

Trust Architecture 2.1 uses the SEC to perform hashing and public key operations to speed up the ISBC phase of secure boot. Other features supported by the Trust Architecture 2.1 ISBC include:

- Alternate Image – ISBC attempts to validate a secondary image if the primary fails
- Key Revocation – ISBC uses a key revocation list to determine which public key to use when validating an image. The ISBC is also involved in unlocking the SFP to allow key revocation fuses to be set.

The NXP Code Signing Tool supports Trust Architecture 1.x, Trust Architecture 2.0, and Trust Architecture 2.1 devices, however the user must specify the chip part number when creating the signed image. The ISBC in Trust Architecture 2.1 devices will not validate images signed as non-Trust Architecture 2.1 devices.

29.5.11 External tamper-detection

Users have the ability to define system-level, physical security policies and report violations of those physical security policies to the security monitor using tamper detection input signals (TA_TMP_DETECT_B). Examples of potential user-defined external tamper detection circuits include contact switches (detecting when the system's case has been opened), light detection, out-of range temperature or voltage detection, and so on. The user-defined external tamper detection circuitry needs to maintain the chip's tamper detect inputs at the voltage specified in the device’s datasheet, with the security monitor reacting to a voltage drop in this signal. Detection of an external tamper event is reported in the SecMon_HPSVSR and SecMon_LPSR.

29.5.12 Pre-boot loader (PBL)

At system boot, the pre-boot loader (PBL) loads a command and reset configuration word (RCW) from a non-volatile memory interface selected by configuration input signals.

The RCW (and subsequent SW images) can be stored on the following interfaces for both normal and non-secure boot:

- QuadSPI

The PBL writes 512 bits of RCW, beginning with the first RCWSR register in the chip configuration CCSR space, thereby performing minimum chip configuration. Additional commands and configuration data (the PBI Image) can be associated with the first command and RCW, enabling the PBL to perform advanced initialization of general memory mapped space managed by the chip (CCSR space and/or DRAM).

The PBL is described in Pre-Boot Loader.

Use of the PBL is mandatory when performing secure boot. At a minimum, the PBL's PBI Image must include a command and destination/value that causes the PBL to write an address to the SCRATCHRW1 register (also known as the ESBC Pointer Register). If the PBL does not perform this operation (or sets the ESBC pointer to the wrong value), the ISBC will fail to validate the ESBC.

To prevent the PBL from exceeding its beneficial role and becoming an attack vector, the PBL's read/write access is controlled by the setting of the ITS bit in the SFP. If ITS is set, the PBL's ability to write to command and configuration register space is greatly reduced. Once the PBL has completed all operations defined by its command file, the PBL is disabled until the next power on reset and the boot phase begins.

29.5.13 Security fuse processor

The security fuse processor (SFP) has the following roles:

- To physically burn fuses during chip provisioning
- To use the values burned into the fuses to enforce security policy in the pre-boot phase, and to securely pass provisioned persistent secrets to other hardware blocks when the system is in a trusted/secure state.

OEMs wishing to use the Trust Architecture must program fuses. (See the device data sheet and the NXP whitepaper *Manufacturing for Trust* for more information.)

The values to be burned to the fuse block are written to the SFP via memory mapped writes to SFP mirror registers. Prior to burning the fuses, the values in the mirror registers can be read to confirm the desired values have been loaded. Once the OEM is satisfied that the desired values are ready for burning, a write to an instruction register is used to initiate fuse burning. The writes to the SFP mirror registers and instruction register can be initiated via software running on a CPU, or via an external interface, such as the JTAG through the secure debug controller (SDC).

Persistent secret values held in the SFP include the one time programmable master key (OTPMK), and debug response value. Once programmed into the SFP, secret values cannot be read back out. Attempts to read the secret values return all 1s.

Note that secret values in a locked SFP cannot be modified, read, or scanned out of a provisioned chip.

If the OEM has a reason to change the mirror register contents prior to initiating fuse programming, the mirror registers can be cleared with a HRESET (not PORESET). Details of the SFP can be found in Security fuse processor (SFP).

29.5.14 Security monitor

The security monitor senses and controls the security state of the device.

The security monitor includes :

- Security State Machine (SSM)
- Master Key Control block
- Security policy configuration registers

The security monitor (Sec_Mon) receives inputs from hardware and software and uses this information to determine if conditions are safe to allow the to use persistent and ephemeral secrets. Details of the security monitor can be found in Security monitor.

29.6 Code signing

As described in Objectives of Trust Architecture 2.1, the chip helps users build trusted systems, and a trusted system is a system that does what its developer and users expect it to do.

What a system does is defined by its programming, so by definition, a properly configured system that executes an authentic version of the developer's software is considered trusted.

This is an important definition, because it should make it clear that the chip does not understand the intent of the code it executes and has limited ability to protect users from bugs or malware contained in authentic versions of the developer's code. However the chip can mitigate the effect of malware and bugs through partition enforcement and use of the Arm eXecute Never (XN) bit.

The starting point for a trusted platform is the creation (by the developer) of a bug free and malware free code base. Once the developer 'trusts' the code, the developer digitally signs the code so that accidental or deliberate modifications to the code base will be detected during the secure boot cycle.

NOTE

Full details of the code signing processing and error codes that can result from secure boot failures are defined in the code signing tool user's manual. The document, *User Enablement for Secure Boot*, is included in the chip's SDK along with the Code Signing Tool.

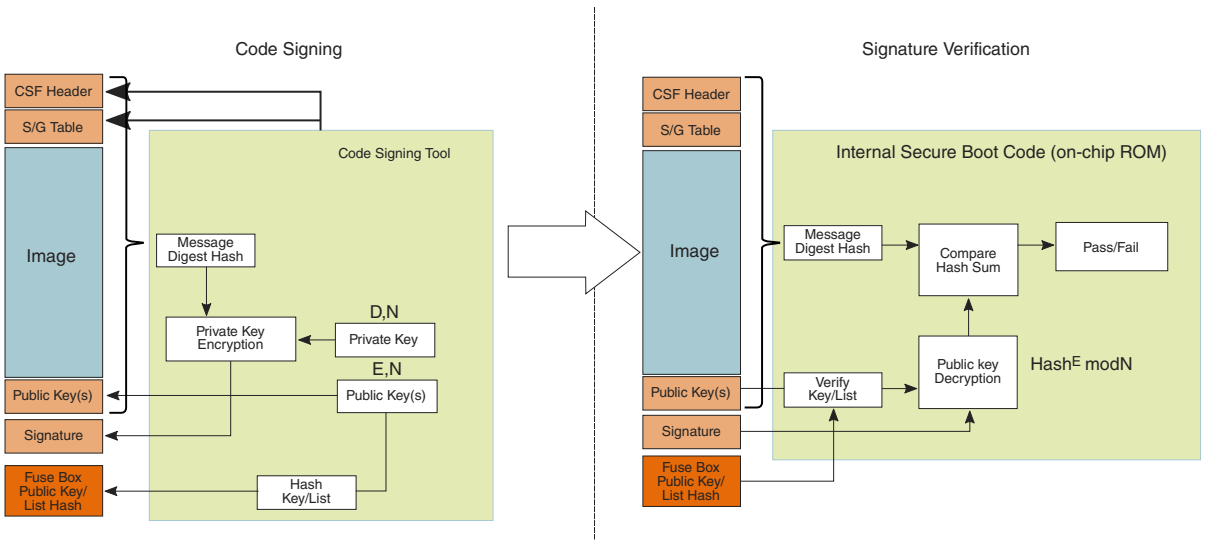


Figure 29-5. Code signing and signature validation

As shown in the figure above, the developer calculates a hash (the chip mandates use of a SHA-256 hash) over the system image. Note that image is loosely defined here to mean executable instructions, configuration information such as device trees, and a command sequence file (CSF) header that is used by the ISBC to validate the image.

The developer generates an RSA public and private key pair. It is the developer's responsibility to tightly control access to the RSA private signature key. If this key is ever exposed, attackers would be able to generate alternate images that would pass secure boot. If this key is ever lost, the developer will be unable to update the image.

The SHA-256 hash is signed using an RSA private signature key. This encrypted hash is known as a digital signature, and the digital signature is appended to the image, with both being written to system non-volatile memory.

29.7 Secure boot sequence

At a high level, secure boot is as simple as the device using an RSA public key to decrypt the signed hash and compare it to a freshly calculated hash over the same system code.

If the comparison passes, the code can be considered authentic.

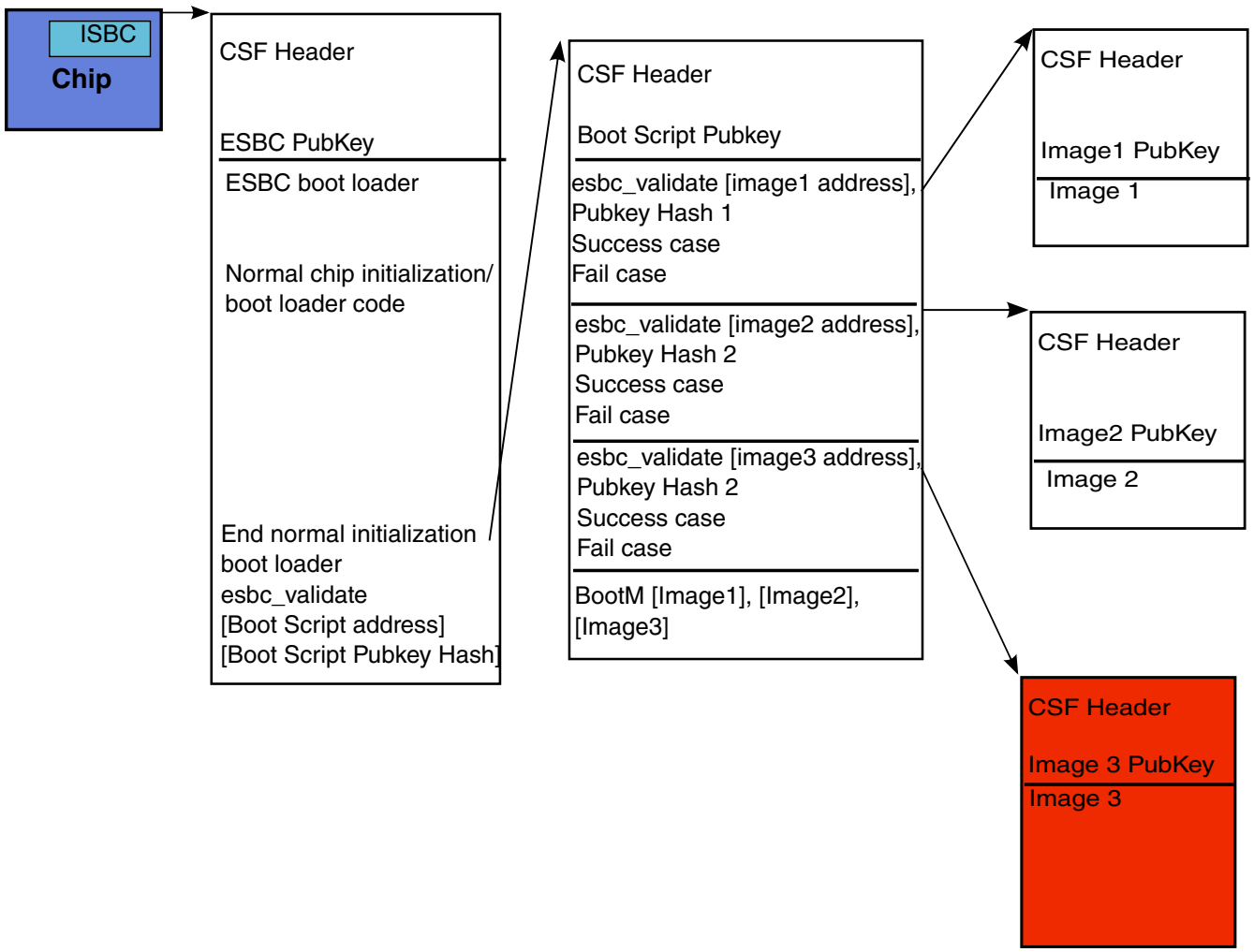


Figure 29-6. Secure boot chain of trust

Although the figure shows the hash being calculated over a single plaintext image, it is possible (even advantageous) for secure boot to occur in stages, with portions of the image to be encrypted (indirectly) with the OTPMK. This prevents attackers from stealing code from flash.

29.7.1 Pre-boot phase

When the device is first powered on, reset control logic blocks all device activity (including scan and debug activity) until fuse values can be accurately sensed.

The most important fuse value at this stage of operation is the intent to secure (ITS) bit. When a user sets ITS, they intend for the system to operate in a secure and trusted manner. The setting of ITS determines the default settings of a range of configuration registers within the chip, essentially locking down interfaces, memory permissions until trusted software is executing.

The PreBoot Loader also runs during the pre-boot phase, reading and executing the PreBoot Image (PBI) from the interface identified by reset config pins. Using secure boot requires a PBI containing a command to load an ESBC pointer to the SCRATCHRW1 register. Additional actions the PBL may take include setting SoC PLLs, configuring the DDR controller, and copying some or all of the ESBC from non-volatile memory into on-chip SRAM or configured DDR. When ITS is set, the PBL is blocked from writing to platform registers which could undermine the SoC's security.

29.7.2 ISBC phase

After the PBL completes executing the PBI, GPP 0 is released to execute the Internal Secure Boot Code from the chip's internal boot ROM. The ISBC runs in TrustZone secure world, and performs the following operations:

1. Platform self tests; ensuring that the chip is in the expected initial state.
2. Policy checks; reading fuses to determine what actions to take should signature validation fail.
3. Signature validation, including locating the Command Sequence File (CSF) header of the external image (read from the DCFG_CCSR_SCRATCHRW1 register), validating the public key to be used in authentication, image authentication, and first instruction validation. Note that NXP generically refers to the external image to be validated as the External Secure Boot Code (ESBC).
4. Error Reporting; if the primary ESBC cannot be successfully validated, the ISBC will write an error code to the chip's DCFG_CCSR_SCRATCHRW2 register. The ISBC will then read DCFG_CCSR_SCRATCHRW3 for a pointer to an alternate or back-up image which OEM's may provision, and repeat step 3. If the alternate image validation fails, the ISBC writes an error code to the chip's DCFG_CCSR_SCRATCHRW4 register.
5. If no errors, the ISBC writes the Sec_Mon command register, transitioning the chip to "Trusted," and the master key (OTPMK or ZMK) is made available to the SEC.
6. The last step in the ISBC is jump to the entry point of the ESBC, as indicated by the CSF Header. ISBC doesn't exit TZ secure world, meaning ESBC will begin execution in TZ secure world.

29.7.3 ESBC phase

Unlike the ISBC, which is in an internal ROM and therefore unchangeable, the ESBC is NXP-supplied reference code, and can be changed by OEMs. The remainder of this section is the description of a reasonable secure boot chain of trust based on NXP's reference external secure boot code. As shown in Figure 29-6, the reference external secure boot code consists of a trusted boot loader (ESBC boot loader), and a trusted bootscript. Future versions of ESBC may be aligned to UEFI, however the same concept of operations applies.

29.7.3.1 Trusted boot loader

At the start of the ESBC phase, the SoC's configuration depends on OEM defined operations performed by the pre-boot loader, and by default operations performed by the ISBC. This is very limited configuration, thereby requiring the first ESBC stage in the chain of trust to perform additional SoC configuration and assist is loading and verifying later code. As noted in ISBC phase, ESBC begins execution in TZ secure world when booting next level image, TZ secure world exits.

It is possible to encrypt portions of the external secure boot code, however the initial instructions of the ESBC must be plaintext. Encrypted portions of the ESBC must be decrypted by the SEC (by request of trusted software) performing a blob decryption operation.

In NXP's reference implementation, the first stage of ESBC is a trusted boot loader, also known as Trusted U-Boot. The trusted boot loader and its associated configuration data are virtually identical to normal uboot, and there is little reason to make them confidential via blob encryption. The trusted boot loader's principal difference from normal uboot is a 'Validate' command. After performing typical SoC initialization operations, such as mapping physical memory, initializing the network interfaces, and loading next-stage software into main memory, the trusted boot loader validates a digital signature over the next stage in the ESBC, the trusted boot script.

The trusted boot loader runs in TZ secure world in order to write configuration registers which are defined as 'secure world only' by default.

29.7.3.2 Trusted Boot Script

In NXP's reference implementation the trusted boot script is a list of additional images to be loaded, validated, and released to execute. The boot script becomes trusted due to the validation performed over it by the trusted boot loader's 'esbc_validate' command. The boot script contains a Validate command for each of the images it loads.

The trusted boot script runs in TZ secure world in order to write configuration registers which are defined as 'secure world only' by default.

29.7.3.3 esbc_validate command

Both the trusted boot loader and trusted bootscript make use of the same reference esbc_validate command. This command essentially repeats the operations performed by the ISBC, and requires the target of the esbc_validate command to have been signed by the QorIQ Code Signing Tool, so that it has a valid CSF header. Users can sign the target with any key from the list bound to the Super Root Key Hash. Note that future implementations of the Code Signing Tool and esbc_validate command may support key hierarchies, and NXP provides its secure boot chain of trust reference implementation as source code to facilitate OEMs replacing NXP's esbc_validate command and ensuing actions with their own schemes.

The esbc_validate command can take the following actions following signature verification:

1. Success Case: The simplest action is to begin execution of the validated code, however the success case can also be used to perform blob decryption. In this case, 'success' code commands the SEC to perform a blob decrypt, with the descriptor telling the SEC where to write the decrypted output. Users can define the esbc_validate command to run in TZ secure world, and use the SEC's secure world job ring to submit the blob descriptor. This allows the SEC to write its output to TZ secure world secure RAM or a secure world only region of external memory. Additional possibilities for success actions include writing to the SecMon HP Command Register to advance the SecMon's state machine from Trusted to Secure.
2. Fail Case: The simplest action is to declare a software security violation by writing to the SecMon's HP Command Register, transitioning the SecMon state machine from Trusted or Secure to Soft Fail. This may be the appropriate reaction to a signature verification failure when validating the boot script, it may be an over-reaction to a signature verification failure of one of the esbc_validate commands in boot script. NXP's reference code follows the same practice as the ISBC, a signature verification failure is a security violation. The SecMon is put into a Soft Fail state, an error code

is written (to the boot console rather than the SCRATCHRWx register), and the chip performs a reset request.

29.7.3.4 Trusted System Images

Trusted system images aren't considered part of the ESBC, the trusted system images are the termination point for NXP's reference chain of trust. It is certainly possible for an image to extend the chain of trust by validating individual applications, but it is expected by this point the validation process would use different methods and different keys than those used by the ISBC and ESBC.

29.7.4 Key revocation

LTA supports revocation of the RSA public keys used by the ISBC to verify the ESBC. The RSA public keys used for this purpose are called Super Root Keys.

In the NXP Code Signing Tool (CST), the OEM defines whether the chip uses a single super root key, or offers a list of super root keys. If using a single super root key, a new flag bit in the CSF header will indicate "Key", otherwise the flag will indicate "Key List". Assuming key list, the OEM populates a list of up to 4 super root keys, and calculates a SHA-256 hash over the list. This hash is written to the SRKH registers in the SFP. See Super Root Key Hash n (SFP_SRKHR*n*) for more information.

As part of code signing, the OEM defines which key in the key list is to be used for validating the image. This key number is included as a new field in the CSF header.

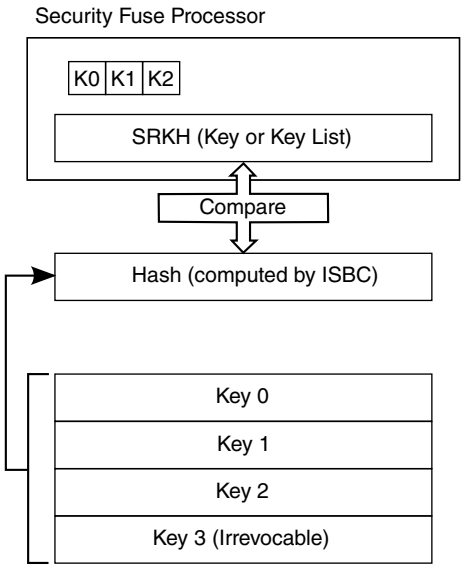


Figure 29-7. ESBC validation with a key list

During secure boot, when the ISBC determines that a key list is in use, it recalculates the hash over the list, and compares it with the hash stored in the SRKH registers. The hash compare is of the original list, revoked keys are still included. If the key list is valid, the ISBC checks the key number indicated in the CSF header against the revocation fuses in the SFP's OEM Security Policy Register (SFP_OSPR). If the key is revoked, the image validation fails. If the key isn't revoked, the ISBC will use it for ESBC validation. The ISBC doesn't enforce using the keys in a specific order.

One possible motivation for an OEM to revoke a super root key is the loss of the associated RSA private key to an attacker. If the attacker has gained access to a legitimate RSA private key, and the attacker can turn on power to the fuse programming circuitry, then the attacker could maliciously revoke keys. To prevent this from being used to permanently disable the system, one super root key does not have an associated revocation fuse.

29.8 Security fuse processor (SFP)

The basic functional uses of the fuses have been described in previous sections.

This section explains SFP fuse read and fuse program operations, SFP registers, and the detailed use of the fuse values themselves.

29.8.1 Fuse programming

SFP mirror registers store values to be written to the array.

The mirror registers are memory mapped, and unless write protected, can be loaded for fuse blowing by software running on the chip, or via an external interface, such as the JTAG.

The chip is enabled for fuse writing by connecting a signal (PROG_SFP) to a 1.8 V source. During reset, PROG_SFP should be tied to GND. After the fuses are written, PROG_SFP must be returned to GND. Once the desired values are written to the mirror registers, the SFP is instructed (by writing to the instruction register) to begin blowing fuses in the array.

The fuse programming sequence and example values are included in the NXP whitepaper, *Manufacturing for Trust*.

29.8.2 Fuse read errors

There are two types of potential fuse read errors; inability of the SFP logic to read the fuse array within a hard-coded time out value, and detection of a misprogrammed or corrupted value in the fuse array.

Either type of error would be reported to the Security Monitor as a SFP security violation. See HPSVSR. The SFP's ability to detect specific bad fuse values is described in Secret Value Hamming Error Status Register (SFP_SVHESR), and the security monitor's ability to detect a bad OTPMK is described in HPSR. If errors are suspected in other fuse registers, software is responsible for checking the actual values against expected values. It is also possible to calculate a checksum over the original values and storing this checksum in one of the OEM Section Scratchpad Registers.

29.8.3 Security fuse processor (SFP) memory map

This section includes the security fuse processor memory map and detailed descriptions of all registers. The default values for the NXP section are typical, the exact values will depend on the values programmed into the fuse array prior to the chip leaving NXP's manufacturing facilities.

There are two types of registers in the SFP; control and status registers, which do not have associated values in the fuse array, and 'mirror' registers, which reflect the values in (or intended to be written to) the fuse array. Mirror registers are further divided into NXP sections of the fuse array and OEM sections of the fuse array.

All registers are 32-bits, and unless otherwise noted, these registers can be accessed with memory-mapped reads and writes.

NOTE

Although the SFP block is provided 4 Kbytes of CCSR space, the SFP memory-mapped registers only occupy the first 256 bytes. Accesses to the reserved space outside the first 256 bytes are aliased to the lower registers and could inadvertently corrupt those register values. Therefore, software should restrict access to only the defined register offsets.

SFP memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1E8_0020	Instruction Register (SFP_INGR)	32	R/W	0000_0000h	29.8.3.1/ 1758
1E8_0024	Secret Value Hamming Error Status Register (SFP_SVHESR)	32	R	0000_0000h	29.8.3.2/ 1758
1E8_0028	SFP Configuration Register (SFP_SFPCR)	32	R/W	See section	29.8.3.3/ 1760
1E8_0038	SFP Version Register (SFP_VERSION)	32	R	See section	29.8.3.4/ 1761
1E8_0200	OEM Security Policy Register (SFP_OSPR)	32	R/W	0000_0000h	29.8.3.5/ 1761
1E8_0204	OEM security policy register 1 (SFP_OSPR1)	32	R/W	0000_0000h	29.8.3.6/ 1764
1E8_0208	Debug Challenge Value Register n (SFP_DCVR0)	32	R/W	0000_0000h	29.8.3.7/ 1765
1E8_020C	Debug Challenge Value Register n (SFP_DCVR1)	32	R/W	0000_0000h	29.8.3.7/ 1765
1E8_0210	Debug Response Value Register n (SFP_DRVR0)	32	W	0000_0000h	29.8.3.8/ 1766
1E8_0214	Debug Response Value Register n (SFP_DRVR1)	32	W	0000_0000h	29.8.3.8/ 1766
1E8_0218	Factory Section Write Protect Register (SFP_FSWPR)	32	R/W	0000_0001h	29.8.3.9/ 1767
1E8_021C	Factory Unique ID Register n (SFP_FUIDR0)	32	R/W	See section	29.8.3.10/ 1768
1E8_0220	Factory Unique ID Register n (SFP_FUIDR1)	32	R/W	See section	29.8.3.10/ 1768
1E8_0224	ISBC Configuration Register (SFP_ISBCCR)	32	R/W	0000_0000h	29.8.3.11/ 1769
1E8_0228	Factory Scratch Pad Fuse Register n (SFP_FSPFR0)	32	R/W	0000_0000h	29.8.3.12/ 1770
1E8_022C	Factory Scratch Pad Fuse Register n (SFP_FSPFR1)	32	R/W	0000_0000h	29.8.3.12/ 1770

Table continues on the next page...

SFP memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1E8_0230	Factory Scratch Pad Fuse Register n (SFP_FSPFR2)	32	R/W	0000_0000h	29.8.3.12/ 1770
1E8_0234	One Time Programmable Master Key n (SFP_OTPMKR0)	32	W	0000_0000h	29.8.3.13/ 1770
1E8_0238	One Time Programmable Master Key n (SFP_OTPMKR1)	32	W	0000_0000h	29.8.3.13/ 1770
1E8_023C	One Time Programmable Master Key n (SFP_OTPMKR2)	32	W	0000_0000h	29.8.3.13/ 1770
1E8_0240	One Time Programmable Master Key n (SFP_OTPMKR3)	32	W	0000_0000h	29.8.3.13/ 1770
1E8_0244	One Time Programmable Master Key n (SFP_OTPMKR4)	32	W	0000_0000h	29.8.3.13/ 1770
1E8_0248	One Time Programmable Master Key n (SFP_OTPMKR5)	32	W	0000_0000h	29.8.3.13/ 1770
1E8_024C	One Time Programmable Master Key n (SFP_OTPMKR6)	32	W	0000_0000h	29.8.3.13/ 1770
1E8_0250	One Time Programmable Master Key n (SFP_OTPMKR7)	32	W	0000_0000h	29.8.3.13/ 1770
1E8_0254	Super Root Key Hash n (SFP_SRKHR0)	32	R/W	0000_0000h	29.8.3.14/ 1772
1E8_0258	Super Root Key Hash n (SFP_SRKHR1)	32	R/W	0000_0000h	29.8.3.14/ 1772
1E8_025C	Super Root Key Hash n (SFP_SRKHR2)	32	R/W	0000_0000h	29.8.3.14/ 1772
1E8_0260	Super Root Key Hash n (SFP_SRKHR3)	32	R/W	0000_0000h	29.8.3.14/ 1772
1E8_0264	Super Root Key Hash n (SFP_SRKHR4)	32	R/W	0000_0000h	29.8.3.14/ 1772
1E8_0268	Super Root Key Hash n (SFP_SRKHR5)	32	R/W	0000_0000h	29.8.3.14/ 1772
1E8_026C	Super Root Key Hash n (SFP_SRKHR6)	32	R/W	0000_0000h	29.8.3.14/ 1772
1E8_0270	Super Root Key Hash n (SFP_SRKHR7)	32	R/W	0000_0000h	29.8.3.14/ 1772
1E8_0274	OEM Unique ID/Scratch Pad Fuse Register n (SFP_OUIDR0)	32	R/W	0000_0000h	29.8.3.15/ 1773
1E8_0278	OEM Unique ID/Scratch Pad Fuse Register n (SFP_OUIDR1)	32	R/W	0000_0000h	29.8.3.15/ 1773
1E8_027C	OEM Unique ID/Scratch Pad Fuse Register n (SFP_OUIDR2)	32	R/W	0000_0000h	29.8.3.15/ 1773
1E8_0280	OEM Unique ID/Scratch Pad Fuse Register n (SFP_OUIDR3)	32	R/W	0000_0000h	29.8.3.15/ 1773
1E8_0284	OEM Unique ID/Scratch Pad Fuse Register n (SFP_OUIDR4)	32	R/W	0000_0000h	29.8.3.15/ 1773

29.8.3.1 Instruction Register (SFP_INGR)

The instruction register (INGR) is the target of software commands intended to read or program the SFP fuse array. As the values in the instruction register are not reflected in the fuse array, there is no write protection for this register. When programming the SFP fuse array, polling INST and ERR provides confirmation that the SFP logic has completed burning fuses. If INST = 00 and ERR = 0, the PROGFB event has completed successfully.

Address: 1E8_0000h base + 20h offset = 1E8_0020h

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

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

SFP_INGR field descriptions

Field	Description
0–22 -	This field is reserved. Reserved
23 ERR	Error status. ERR clears on a write to INST, and becomes valid when INST clears. 0 Transaction completed successfully. 1 An error occurred during the transaction.
24–31 INST	Instruction. INST will clear to IDLE upon completion of any instruction. 00 IDLE. SFP is not actively processing an instruction. 01 READFB. Read the entire fusebox and load the contents into the corresponding mirror registers. 10 PROGFB. Permanently write data from the mirror registers into the fuse array.

29.8.3.2 Secret Value Hamming Error Status Register (SFP_SVHESR)

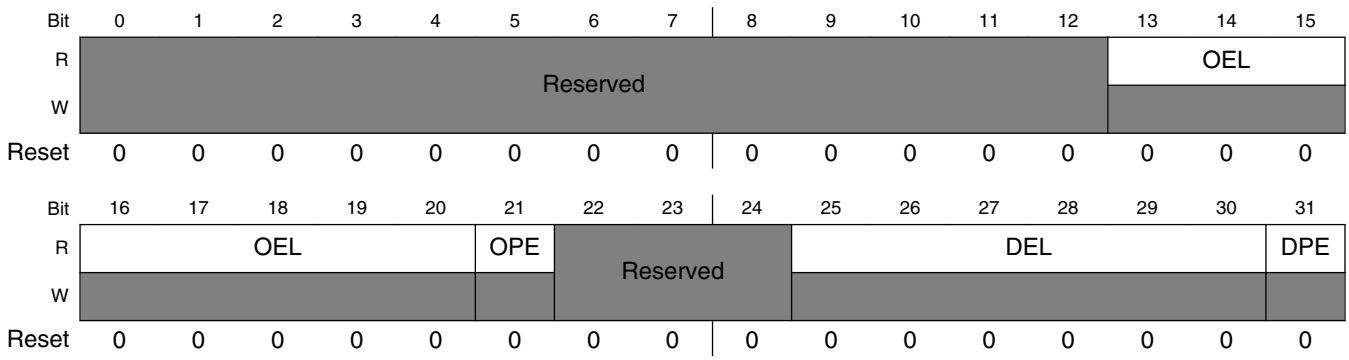
NOTE

To better understand the functionality of this register, the reader is advised to first review Debug Response Value Register n (SFP_DRVR*n*) and One Time Programmable Master Key n (SFP_OTPMKR*n*).

The SFP performs Hamming code error checking on the secret debug response value (DRV) and one time programmable master key (OTPMK). The Secret Value Hamming Error Status Register (SVHESR) provides the results of the Hamming code check. The DRV and OTPMK values cannot be read directly, but the user can use the SVHESR to verify they were written correctly after programming. When the secret values are correct, SVHESR contains all zeroes; otherwise, the error/syndrome fields can be used to determine which value is incorrect along with the error location.

Any non-zero value read from SVHESR indicates a Hamming code error has been detected. The check is performed continually and immediately on whatever data is stored in DRVR and OTPMKR, so SVHESR is always up to date with the secret values in the mirror array. This allows a user to check their secret values for correctness before programming into the fusebox.

Address: 1E8_0000h base + 24h offset = 1E8_0024h



SFP_SVHESR field descriptions

Field	Description
0–12 -	This field is reserved. Reserved
13–20 OEL	OTPMKR error location. Hamming code syndrome bits for OTPMKR. 000000 OTPMKR is error free (OPE == 0) or OTPMKR contains more than one error (OPE == 1). Non-zero OTPMKR contains an error at location encoded in OEL.
21 OPE	OTPMKR parity error. 0 OTPMKR is error free (OEL == 0) or OTPMKR contains more than one error (OEL != 0). 1 OTPMKR contains at least one error.
22–24 -	This field is reserved. Reserved
25–30 DEL	DRVR error location. Hamming code syndrome bits for DRVR. 000000 DRVR is error free (DPE == 0) or DRVR contains more than one error (DPE == 1). Non-zero DRVR contains an error at location encoded in DEL.
31 DPE	DRVR parity error. 0 DRVR is error free (DEL == 0) or DRVR contains more than one error (DEL != 0). 1 DRVR contains at least one error.

29.8.3.3 SFP Configuration Register (SFP_SFPCR)

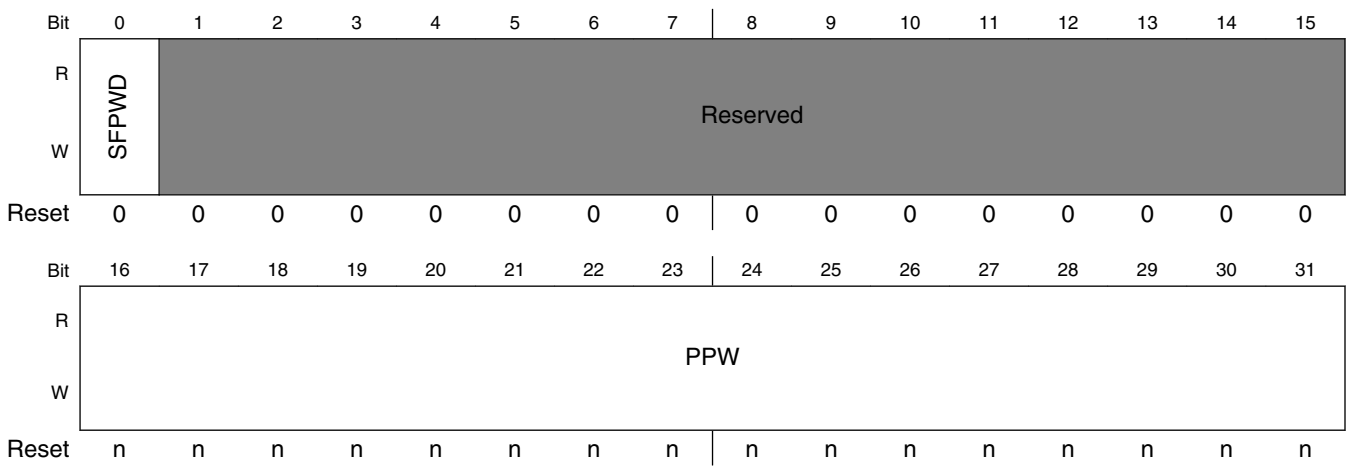
The SFP configuration register (SFPCR) provides configuration values for the fusebox.

For reliable writing of fuses, the write pulse width must be correctly tuned for the frequency/temperature/voltage conditions that prevail at the time of writing. The SFPCR's default value is set appropriately for writing fuses on top frequency chips. When performing fuse programming at other platform frequencies, the default value of PPW must be overwritten prior to writing the Instruction Register.

The SFPCR also includes the Write Disable bit. Trusted software (ESBC) selectively sets this bit to allow or disallow programming of SFP fuses which are outside of the protection of the OEM Security Policy Register's Write Protect fuse. The specific fuse fields protected by the Write Disable are:

- OSPR[Kn] Key Revocation
- OEM Scratch Pad Register 1

Address: 1E8_0000h base + 28h offset = 1E8_0028h



SFP_SFPCR field descriptions

Field	Description
0 SFPWD	SFP write disable **NOTE:** Once SFPWD is set, it can only be cleared by resetting the chip. 0 SFP facilities are available for access (subject to write-protection bits). 1 Writes to SFP mirror registers are blocked. Fuse programming is disabled. **NOTE:** This bit is set by default during execution of the ISBC. OEMs set the 'Leave Writeable' flag in the CSF header to tell the ISBC not to set this bit. The major use case for not setting this bit is key revocation.

Table continues on the next page...

SFP_SFPCR field descriptions (continued)

Field	Description
1–15 -	This field is reserved. Reserved
16–31 PPW	Program pulse width. PPW determines the length of the program strobe used by the fusebox. The reset value is a safe default for programming under typical conditions (at top frequency bin) The optimal value for PPW is calculated as the SFP module input clock frequency (in MHz) * 12 where the SFP module input clock is platform clock/4.

29.8.3.4 SFP Version Register (SFP_VERSION)

VERSION contains the current version of the secure fuse processor module.

Address: 1E8_0000h base + 38h offset = 1E8_0038h

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

SFP_VERSION field descriptions

Field	Description
0–11 -	This field is reserved. Reserved
12–15 MAJOR	Major revision level.
16–19 -	This field is reserved. Reserved
20–23 MINOR	Minor revision level.
24–27 -	This field is reserved. Reserved
28–31 SUB	Sub-version

29.8.3.5 OEM Security Policy Register (SFP_OSPR)

This register configures critical security policy decisions that OEMs need to make if they are concerned with platform trust.

Key revocation. The ISBC supports a key list of up to four keys, three of which can be revoked using the Key Revocation fuses in this register.

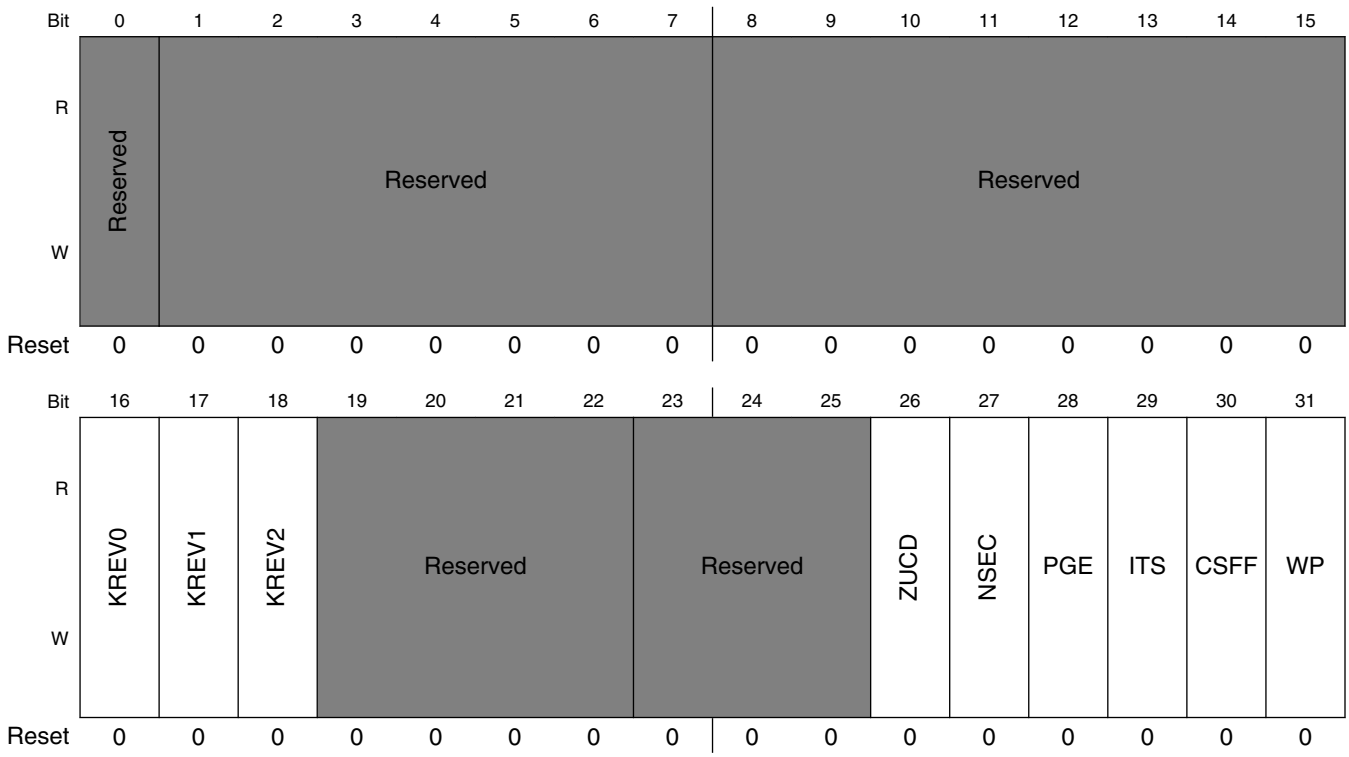
Intent to secure (ITS) signifies the OEM's intention for the system to be secure. Setting ITS forces the booting core to begin execution in the IBR, performing the ISBC signature validation steps described in Internal boot ROM and ISBC. The SB_EN bit in the Reset Configuration Word (RCW) can also trigger the chip to perform secure boot, however no other method is guaranteed to work in the presence of an attacker with physical access to the system. Whether the system actually transitions to a Trusted/Secure state is dependent on successful signature validation and absence of hardware security violations.

Sensitive flip flops are any flops holding secret keys, or intermediate plaintext data that is not eligible for leaving the security monitor, security engine (SEC), SFP, or SDC. Clear Sensitive Flip Flops (CSFF) is a security policy decision related to blocking expert level debug modes collectively known as scan. Scan is the most advanced non-destructive debug mechanism available, and it is considered unlikely that even sophisticated attackers can exploit scan without access to detailed NXP design files. Scan can reach flops that are not accessible through memory mapped reads and writes. When CSFF is set, upon entry into scan mode, all sensitive flip flops are cleared preventing any possibility of extraction of secret values. By clearing these flip flops on scan entry, expert mode debuggers are prevented from extracting secret values. It is strongly recommended that OEMs always set CSFF on production systems.

Setting the Write Protect (WP) bit prevents further writes to the OEM section's mirror registers until the next SoC reset. Once the WP fuse is programmed, writes to mirror registers and further programming of the fuse block is permanently disabled.

The chip is available from NXP in pin compatible export-controlled (SEC enabled) and non-export-controlled (SEC disabled) versions. NXP has additionally implemented a permanent SEC disable bit in the OEM section of the SFP. This allows customers to purchase a single version of the chip (SEC enabled), and permanently convert it to a SEC disabled chip when the presence of an export controlled processor in the system complicates system level export control. Note that the chip’s part markings will indicate an export controlled chip, generally requiring the OEM to document the means by which the processor and system can be legitimately considered to be free of export controlled encryption technology. Users should only set (SEC_DIS) if they intend to permanently disable the SEC. Note that SEC disabled chips cannot perform secure boot.

Address: 1E8_0000h base + 200h offset = 1E8_0200h



SFP_OSPR field descriptions

Field	Description
0 -	This field is reserved. Reserved
1–7 -	This field is reserved. Reserved
8–15 -	This field is reserved.
16 KREV0	Key Revocation: first key 0: Super Root key 0 is valid 1: Super Root First key 0 is revoked This bit is not protected by OSPR WP.
17 KREV1	Key Revocation: second key 0: Super Root key 1 is valid 1: Super Root key 1 is revoked This bit is not protected by OSPR WP.
18 KREV2	Key Revocation: third key 0: Super Root key 2 is valid 1: Super Root key 2 is revoked This bit is not protected by OSPR WP.
19–22 -	This field is reserved.

Table continues on the next page...

SFP_OSPR field descriptions (continued)

Field	Description
23–25 -	This field is reserved.
26 ZUCD	ZUCD: Disable ZUC Crypto Engine 0: SEC ZUC crypto engine is enabled 1: SEC ZUC crypto will be disabled.
27 NSEC	No Encryption Features. 0: All SEC security features are enabled 1: SEC encryption algorithms are disabled (DMA, hashing, RNG, and CRC features available).
28 PGE	PBL gate enable 0 PBL can access most of the memory map. 1 PBL access is restricted.
29 ITS	Intent to Secure.
30 CSFF	Clear Sensitive Flip Flops.
31 WP	Write Protect OEM Section of SFP. The key revocation bits, are not write protected by this bit

29.8.3.6 OEM security policy register 1 (SFP_OSPR1)

The register is set by OEMs to configure the debug access permissions for the secure debug controller (SDC).

NOTE

Debug permissions can still be set in SEC-disabled chips. A SEC-disabled chip won't perform secure boot or support the use of the OTPMK, making the normal difference between Conditionally Closed without Notification and Conditionally Closed with Notification irrelevant. OEMs may still choose between leaving debug open, offering conditional access via the challenge/response, or permanently closing the debug interface.

Address: 1E8_0000h base + 204h offset = 1E8_0204h

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

SFP_OSPR1 field descriptions

Field	Description
0–15 -	Reserved This field is reserved.
16–28 -	This field is reserved.
29–31 DBLEV	Debug Level 000 Wide open: Debug portals are enabled unconditionally. 001 Conditionally open via challenge response, without notification. 01x 01x: Conditionally open via challenge response, with notification. 1xx 1xx: Closed. All debug portals are disabled.

29.8.3.7 Debug Challenge Value Register n (SFP_DCVR*n*)

The DCVR *n* are set by OEMs to configure a 64-bit value which the SDC outputs when the debug interface requests access to the SoC internal space.

During challenge/response cycle, a read to 0x0001_0000_0000_0000 (through JTAG) will access the 64b Secure Debug Challenge Register in Secure Debug Controller

Example:

Original 64b value: d0d0d0d0_c0c0c0c0

Write to DCVR 0: d0d0d0d0

Write to DCVR 1: c0c0c0c0

During challenge/response cycle, the debugger will output c0c0c0c0_d0d0d0d0.

The Debug Challenge Value Registers and Debug Response Value Registers are logically only useful when debug permissions are set to one of the conditional access modes. Regardless, any use of secure boot requires a valid value to be programmed into the Debug Response Value, otherwise the Hamming error will cause the SFP to signal a security violation to the Security Monitor, preventing the chip from reaching the Trusted/ Secure state.

Address: 1E8_0000h base + 208h offset + (4d × i), where i=0d to 1d

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

SFP_DCVR*n* field descriptions

Field	Description
0–31 DCVRn	DCVR0 is the upper part (bits 63-32) and DCVR1 is the lower part (bits 31-0) of the 64-bit debug challenge value. Both of these registers reset to 0x0000_0000 unless programmed into the fuse array.

29.8.3.8 Debug Response Value Register n (SFP_DRVR*n*)

The DRVRs are set by OEMs to configure a 64-bit secret value which is compared to the response value input by the debugger, through the Secure Debug Controller (SDC), after getting the challenge value. There is no required relationship between the challenge and response values. The OEM determines whether the challenge and response value are algorithmically related, or simply linked in a database.

Because the debug response value is effectively a key for opening the SDC, the SFP does not allow the value written to the DRVR to be read out. In order to detect fuse array misprogramming or a fuse reliability failure, the SFP implements an error detection scheme for the debug response value.

Although there is no required algorithmic relationship between the challenge and response values, the response value must be properly constructed to include the error detection code, otherwise the SFP reports an error through SVHESR, Secret Value Hamming Error Status Register (SFP_SVHESR).

The process for programming the debug response value is for the OEM to select a value, then execute a Hamming algorithm that replaces 7 bits in the 64-byte debug response value (key bits 0, 1, 2, 4, 8, 16, 32 with the error detection code.

Example:

DRV with Hamming code applied and written to the DRVRs:

Original 64b value: 01aa00bb_cafecafe

After applying Hamming code: 01aa00bb_cafeca77

Write to DRVR0: cafeca77

Write to DRVR1: 01aa00bb

During challenge/response cycle, write 01aa00bb_cafeca77 to 0x0001_0000_0000_0008 (through JTAG). This is the address of the 64b Secure Debug Response Register in Secure Debug Controller.

A successful challenge/response operation unlocks the SDC. Three failed challenge/response operations trigger a security violation, and lock the SDC against further challenge/response cycles until the SoC is reset. Users can exploit this SDC lock-out by using an intentional failed challenge/response to lock the SDC at the conclusion of a debug session.

Note that it is the OEM's responsibility to control access to their DRV database. Exposure to unauthorized parties can compromise system security, and losing the debug response value for a SoC can permanently prevent the OEM from using SoC's debug resources. NXP does not have a super user debug response value.

Address: 1E8_0000h base + 210h offset + (4d × i), where i=0d to 1d

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

SFP_DRVR*n* field descriptions

Field	Description
0–31 DRVRn	DRVR0 is the upper part (bits 63-32) and DRVR1 is the lower part (bits 31-0) of the 64-bit debug response value. Both of these registers reset to 0x0000_0000 unless programmed into the fuse array.

29.8.3.9 Factory Section Write Protect Register (SFP_FSWPR)

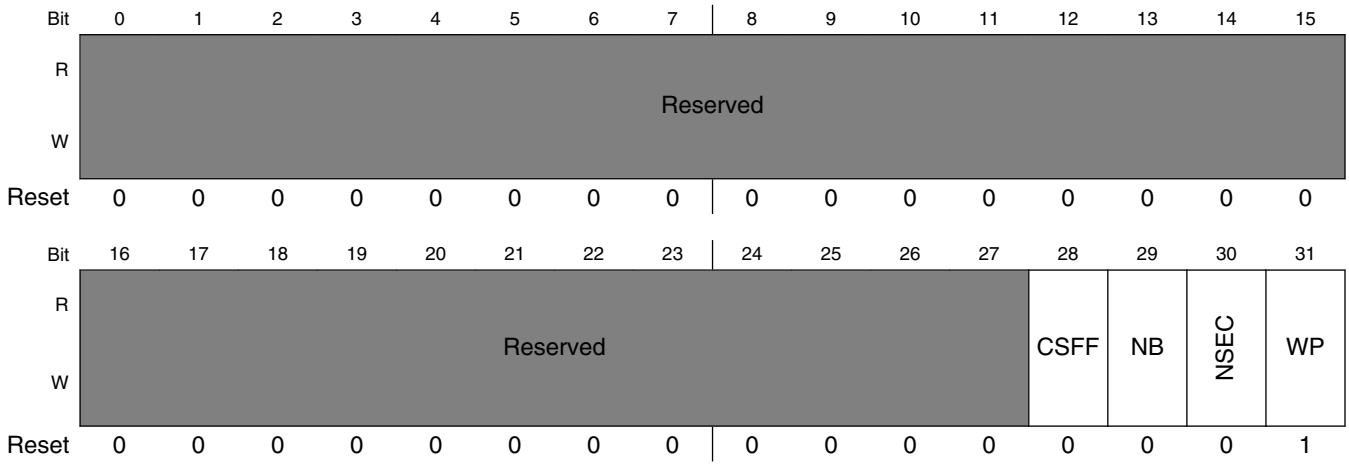
NXP programs the Write Protect (WP) fuse bit on all chips sold. This prevents further writes to the factory (NXP) section of the SFP, including mirror registers and associated fuses.

NOTE

This register is protected by the factory (NXP) write protect bits. See OEM Security Policy Register (SFP_OSPR) and Factory Section Write Protect Register (SFP_FSWPR) for more information on write protection.

The chip is available in pin compatible export-controlled (SEC available) and non-export-controlled (SEC not available) versions. This register has a feature identifier bit that can be used by software to help distinguish between the two versions.

Address: 1E8_0000h base + 218h offset = 1E8_0218h



SFP_FSWPR field descriptions

Field	Description
0–27 -	This field is reserved. Reserved
28 CSFF	Clear Sensitive Flip-flops (factory version) 0 All SFP internal facilities will be accessible through scan (DFT/LSRL) 1 Flops that store secret or sensitive data will be cleared whenever the SOC enters scan mode
29 NB	No battery back-up 0 Battery backup features are enabled. 1 Battery backup features are disabled.
30 NSEC	No Secure Features 0: All SEC security features are enabled 1: SEC security features will be disabled
31 WP	Write protect factory (NXP) section of SFP 0: Registers in the factory (NXP) address space (0x218 - 0x230, 0x288 - 0x2A4) are writeable 1: Writes to registers to the factory (NXP)address space will be blocked

29.8.3.10 Factory Unique ID Register n (SFP_FUIDR*n*)

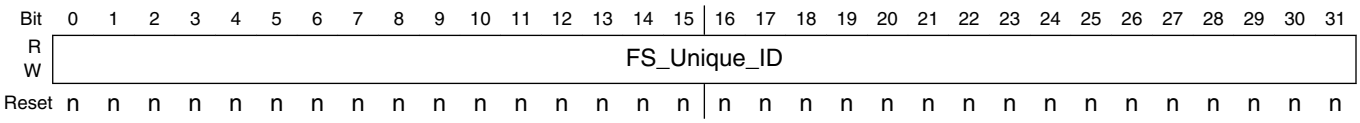
The FUIDR is set (and write protected) by NXP prior to part shipment to provision a pseudo-random software readable value which can be (optionally) included as part of the ESBC for the purposes of digital signature validation by the ISBC. This allows OEMs to create images that can only pass secure boot on a specific chip. OEM can consider this to be an unalterable highly unique value for the purposes of chip identification.

NOTE

The FUID value is the concatenation of a SHA256 hash of a variety of manufacturing information, which is itself globally

unique. The nature of hashes allows for collisions, and collisions may occur within the total population of QorIQ SoCs.

Address: 1E8_0000h base + 21Ch offset + (4d × i), where i=0d to 1d



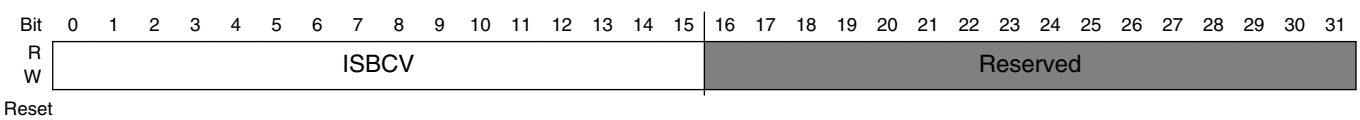
SFP_FUIDR*n* field descriptions

Field	Description
0–31 FS_Unique_ID	Reset to globally unique value.

29.8.3.11 ISBC Configuration Register (SFP_ISBCCR)

The Internal Secure Boot Code (ISBC) can be configured to run in a number of different ways, depending on the specific needs of the part. The ISBC Configuration Register contains a 16-bit configuation field that ISBC reads upon startup.

Address: 1E8_0000h base + 224h offset = 1E8_0224h



SFP_ISBCCR field descriptions

Field	Description
0–15 ISBCV	ISBC Configuration Value
16–31 -	This field is reserved. Reserved

29.8.3.12 Factory Scratch Pad Fuse Register n (SFP_FSPFR*n*)

The FSPFRs are provided purely for their storage function. SFP logic does not use the contents in these registers. NXP may use this register for non-tamperable storage of configuration data. OEMs may observe non-zero values in these registers.

Address: 1E8_0000h base + 228h offset + (4d × i), where i=0d to 2d

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

SFP_FSPFR*n* field descriptions

Field	Description
0–31 FS_SCRATCH_ PAD_FUSES	Factory Scratch Pad Fuses

29.8.3.13 One Time Programmable Master Key n (SFP_OTPMKR*n*)

The OTPMK registers are set by OEMs to configure a 256-bit secret value that becomes available for use by the SEC module to derive the AES blob keys when the security monitor is in the Trusted state or Secure state. The 256-bit secret value is stored in a series of eight 32-bit registers OTPMKR0-OTPMKR7. The primary purpose of the OTPMK is encryption and decryption of additional secret keys (also usable only by the SEC module) that can be used to protect arbitrary data. This level of indirection increases the difficulty of cryptanalysis of the OTPMK.

For obvious reasons, the SFP does not allow the value written to the OTPMK registers to be read out. Once a non-zero value is written to any of the OTPMK registers, any read will return all 1s. In order to detect fuse array misprogramming or a fuse reliability failure, the SFP implements an error detection scheme for the OTPMK. The OTPMK must be properly constructed to include the error detection code, otherwise the SFP reports an error in the Secret Value Hamming Error Status Register (SFP_SVHESR).

The process for programming the OTPMK is for the OEM to select a 256 bit value, then execute a Hamming algorithm that replaces 9 bits in the 256-bit value (0, 1, 2, 4, 8, 16, 32, 64, and 128) with the error detection code.

Example:

256b Initial Random Value:
0x9fb2_71c1_f2f2_4698_fbaa_addb_4241_ecec_19fd_8d72_fc05_be54_5225_5d2a_ef0e_939e

256b OTPMK Value:
0x9fb2_71c1_f2f2_4698_fbaa_addb_4241_eced_19fd_8d72_fc05_be54_5225_5d2b_ef0f_928b

Hamming Value:
0x0000_0000_0000_0000_0000_0000_0000_0001_0000_0000_0000_0000_0000_0001_0001_0115

In this example, the bits changed by the Hamming algorithm are 0,2,4,8,16, 32, and 128.

The word order for writing the value to the One Time Programmable Master Key Registers is:

OTMPKR0: 0x9fb2_71c1

OTMPKR1: 0xf2f2_4698

OTMPKR2: 0xfbaa_addb

OTMPKR3: 0x4241_eced

OTMPKR4: 0x19fd_8d72

OTMPKR5: 0xfc05_be54

OTMPKR6: 0x5225_5d2b

OTMPKR7: 0xef0f_928b

NOTE

The bit order positions of keys in the registers are different from previous versions of the Trust Architecture. For Trust Architecture 2.0, the bit order positions are as follows-

OTPMKR0: key bits 255-224

OTPMKR1: key bits 223-192

OTPMKR2: key bits 191-160

OTPMKR3: key bits 159-128

OTPMKR4: key bits 127-96

OTPMKR5: key bits 95-64

OTPMKR6: key bits 63-32

OTPMKR7: key bits 31-0

Address: 1E8_0000h base + 234h offset + (4d × i), where i=0d to 7d

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

SFP_OTPMKR*n* field descriptions

Field	Description
0–31 OTPMKn	• OTPMKR0: key bits 255-224 • OTPMKR1: key bits 223-192 • OTPMKR2: key bits 191-160 • OTPMKR3: key bits 159-128 • OTPMKR4: key bits 127-96 • OTPMKR5: key bits 95-64 • OTPMKR6: key bits 63-32 • OTPMKR7: key bits 31-0 Reset to all zeros unless programmed into fuse array.

29.8.3.14 Super Root Key Hash n (SFP_SRKHR*n*)

Support for key revocation and a SRK list is new for Trust 2.0; however, it requires no changes to the Super Root Key Hash registers. Key revocation is achieved through updates to the ISBC and the addition of key revocation bits in the OEM Security Policy Register (SFP_OSPR). See Key revocation for more information about the key revocation process.

Address: 1E8_0000h base + 254h offset + (4d × i), where i=0d to 7d

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

SFP_SRKHR*n* field descriptions

Field	Description
0–31 SRKHn	SRKHR0: key hash bits 255-224 SRKHR1: key hash bits 223-192 SRKHR2: key hash bits 191-160 SRKHR3: key hash bits 159-128

SFP_SRKHR*n* field descriptions (continued)

Field	Description
	SRKHR4: key hash bits 127-96 SRKHR5: key hash bits 95-64 SRKHR6: key hash bits 63-32 SRKHR7: key hash bits 31-0 Reset to all zeros unless programmed into fuse array.

29.8.3.15 OEM Unique ID/Scratch Pad Fuse Register n (SFP_OUIDR*n*)

The OUIDRs are provided purely for their storage function. The SFP logic does not use the contents in these registers. OEMs can use these registers for non-tamperable storage of arbitrary data up to 32-bits in length.

NOTE

SFP_OUIDR0-OUIDR2 are protected by the OEM write protect bits; OUIDR3-OUIDR4 are not protected and can be programmed even after write-protect is set. See OEM Security Policy Register (SFP_OSPR) and Factory Section Write Protect Register (SFP_FSWPR) for more information on write protection.

Address: 1E8_0000h base + 274h offset + (4d × i), where i=0d to 4d

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

SFP_OUIDR*n* field descriptions

Field	Description
0–31 UIDSn	Unique ID scratch pad value

29.9 Security monitor

The Security Monitor (Sec_Mon) is a companion module to the security engine (SEC) module. The Security Monitor is the SOC's central reporting point for security-relevant events such as the success or failure of boot software validation and the detection of potential security compromises. This security event information determines whether the SoC (HW and SW) is in the proper state to allow the SEC to use persistent and ephemeral secrets.

Based on values in the SFP and configured bits within the Security Monitor registers, the Security Monitor is able to accept and detect a variety of security violation inputs and perform configured policy enforcement actions.

NOTE

The chip does not support SNVS in battery-backed (LP) mode.

29.9.1 Security monitor features summary

The Security Monitor includes the following features:

- Security state machine
  - Receives configuration inputs from the security fuse processor
  - Receives security violation inputs from the various detectors in the chip
  - Tracks security state
  - Generates security state outputs to the SEC and other logic within the chip
  - Programmable High Assurance Counter (HAC) to control time delay before system hard reset request generation
- Master key checking and control
  - Performs validity checks for the one-time programmable master key and zeroizable master key before allowing the SEC to use them
  - Selects between OTPMK, ZMK, and a test master key output according to the system condition and configuration.
- Zeroizable master key
  - The ZMK can be programmed either by SW or directly by HW
  - Interface to the random number generator for direct HW programming
  - The key can be selected for use in blob operations by HW cryptographic accelerators
  - The key value will be zeroized in case of a critical security violation
- Register access protection
  - The Security Monitor registers can be programmed only when the Security Monitor is in a functional state
  - Some Security Monitor registers/values can only be programmed once per boot cycle
- Violation/tamper detection and reporting
  - Detects (internal to Security Monitor) the following security violations:
    - Scan entry/exit
    - Power glitch
    - Invalid OTPMK (Hamming check failure)
    - ZMK ECC check failure
    - CSU alarm

- Detects (receives from detectors external to Security Monitor) the following security violations:
  - Software violations
  - Hardware security violation inputs
    - Run time integrity checker failure
    - Security fuse processor
    - Secure debug controller activation
    - External tamper detect
- Direct connections to SEC and COP to lock out access to the OTPMK/ZMK, and force zeroization of sensitive information
- Configurably triggers a device hard reset
- Reports to software (interrupts) security violations

29.9.2 Operational states

The Security Monitor incorporates a security state machine (SSM), which is responsible for monitoring system security violations and controlling security state of the system.

The SSM states are defined as follows:

- Init-initial state after system power-on reset
- Check-system performs security checks
- Non-secure (functional)-system operates in non-secure state
- Trusted (functional)-system operates in trusted state
- Secure (functional)-system operates in secure state
- Soft fail-security violation/tamper was detected. The system state may be unpredictable. Access to persistent and ephemeral secrets locked out, zeroization of secrets within the SEC.
- Hard fail-system hard reset is requested. All behaviors of Soft fail, plus zeroization of some SoC caches and main memory, SoC reset initiated.

Three of the SSM states, trusted, secure, and non-secure, are considered functional states.

29.9.3 Signals

This section provides information on the external signals.

Table 29-2. External Signals

Signal name	I/O	Description
TA_TMP_DETECT_B	I	Tamper detect. TA_TMP_DETECT_B is active low. If a tamper sensor is used, a 1K pulldown resistor is strongly recommended to maintain the signal at the specified voltage until a tamper is detected. If Trust is used without tamper sensors, tie TA_TMP_DETECT_B high.

29.9.4 SecMon Register Descriptions

This section contains detailed register descriptions for the SecMon 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.

SecMon 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 SecMon_HP Command Register.

In addition, all privileged access registers (except HPSVSR and LPSR) can be written to only if the system security monitor is in one of the three functional states:

- Non-Secure
- Trusted
- Secure

Certain fields in the SecMon_HP Command Register can be written in non-functional system security monitor states (init, check, soft fail, and hard fail) as follows:

- SSM_ST, SW_SV, and SW_FSV can be written in check or soft fail state.

- The HAC_STOP bit can only be set in soft fail state but can be cleared in either soft fail or a functional state.
- HAC_LOAD, HAC_CLEAR bits and HPSVSR, LPSR Registers can be accessed in soft fail or a functional state.

The system security monitor state does not restrict read access to SecMon registers.

Non-privileged read/write accessible registers are read/write accessible by any software.

System reset causes all registers to be reset.

The following table shows the SecMon main memory map.

29.9.4.1 SecMon Memory Map

Base address: 1E90000h

Offset	Register	Width (In bits)	Access	Reset value
0h	SecMon_HP Lock Register (HPLR)	32	RW	00000000h
4h	SecMon_HP Command Register (HPCOMR)	32	RW	00000000h
Ch	SecMon_HP Security Interrupt Control Register (HPSICR)	32	RW	00000000h
10h	SecMon_HP Security Violation Control Register (HPSVCR)	32	RW	00000000h
14h	SecMon_HP Status Register (HPSR)	32	RO	8000B000h
18h	SecMon_HP Security Violation Status Register (HPSVSR)	32	W1C	80000000h
1Ch	SecMon_HP High Assurance Counter IV Register (HPHACIVR)	32	RW	00000000h
20h	SecMon_HP High Assurance Counter Register (HPHACR)	32	RO	00000000h
64h	SecMon_LP Power Glitch Detector Register (LPPGDR)	32	RW	00000000h
BF8h	SecMon_HP Version ID Register 1 (HPVIDR1)	32	RO	003A0204h
BFCh	SecMon_HP Version ID Register 2 (HPVIDR2)	32	RO	05000100h

29.9.4.2 SecMon_HP Lock Register (HPLR)

29.9.4.2.1 Offset

Register	Offset
HPLR	0h

29.9.4.2.2 Function

The SecMon_HP Lock Register contains lock bits for the SecMon registers. This is a privileged write register.

29.9.4.2.3 Diagram

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

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W								LPTDCR_SL								SET_TO_DIS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

29.9.4.2.4 Fields

Field	Function
31-29 —	Reserved.
28 —	Reserved.
27 —	Reserved.
26 —	Reserved.
25 —	Reserved.
24 —	Reserved.
23-19 —	Reserved.
18 HAC_L	High Assurance Configuration Lock When set, prevents any writes to HPHACIVR, HPHACR, and HAC_EN bit of HPCOMR. Once set, this bit can only be reset by the system reset. 0b - Write access is allowed 1b - Write access is not allowed
17	HP Security Interrupt Control Register Lock

Table continues on the next page...

Field	Function
HPSICR_L	When set, prevents any writes to the HPSICR. Once set, this bit can only be reset by the system reset. 0b - Write access is allowed 1b - Write access is not allowed
16 HPSVCR_L	HP Security Violation Control Register Lock When set, prevents any writes to the HPSVCR. Once set, this bit can only be reset by the system reset. 0b - Write access is allowed 1b - Write access is not allowed
15-9 —	Reserved.
8 LPTDCR_SL	LP Tamper Detectors Configuration Register Soft Lock When set, prevents any writes to the LPTDCR. Once set, this bit can only be reset by the system reset. 0b - Write access is allowed 1b - Write access is not allowed
7 —	Reserved.
6 —	Reserved.
5 —	Reserved.
4 —	Reserved.
3-2 —	Reserved.
1 —	Reserved.
0 SET_TO_DIS	Reserved.

29.9.4.3 SecMon_HP Command Register (HPCOMR)

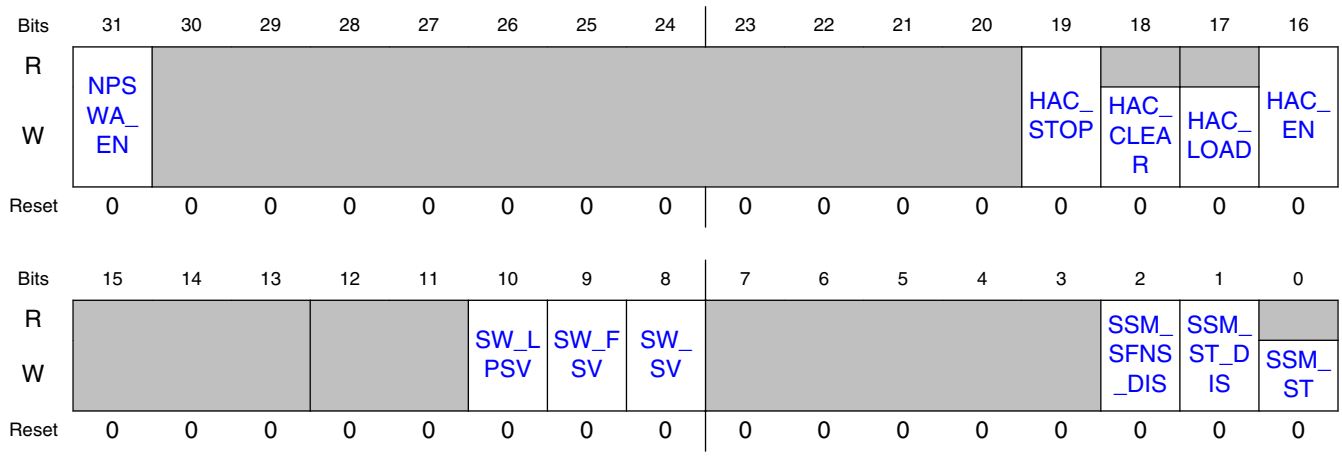
29.9.4.3.1 Offset

Register	Offset
HPCOMR	4h

29.9.4.3.2 Function

The SecMon_HP Command Register contains the command, configuration, and control bits for the SecMon block. Some fields of this register can be written to in check and soft fail states in addition to the standard write access in functional states. This is a privileged write register.

29.9.4.3.3 Diagram



29.9.4.3.4 Fields

Field	Function
31 NPSWA_EN	Non-Privileged Software Access Enable When set, allows non-privileged software to access all SecMon 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 —	Reserved.
19 HAC_STOP	High Assurance Counter Stop This bit can be set only when SSM is in soft fail state. When set, it stops the high assurance counter and prevents transition to the hard fail state. This bit can be cleared in a functional or soft fail state. If the bit is cleared in the soft fail state, the high assurance counter counts down from the place where it was stopped. 0 HAC counter can count down 1 HAC counter is stopped
18 HAC_CLEAR	High Assurance Counter Clear

Table continues on the next page...

Field	Function
	When set, it clears the High Assurance Counter Register. It can be cleared in a functional or soft fail state. If the HAC counter is cleared in the soft fail state, the SSM transitions to the hard fail state if high assurance configuration is enabled (HAC_EN is set). This self-clearing bit is always read as zero. 0b - No Action 1b - Clear the HAC
17 HAC_LOAD	High Assurance Counter Load When set, it loads the High Assurance Counter Register with the value of the High Assurance Counter Load Register. It can be done in a functional or soft fail state. This self-clearing bit is always read as zero. 0b - No Action 1b - Load the HAC
16 HAC_EN	High Assurance Configuration Enable This bit controls the SSM transition from the soft fail to the hard fail state. When this bit is set and software fails to stop the HAC before it expires, the SSM transitions to the hard fail state. This bit cannot be changed once HAC_L bit is set. 0b - High Assurance Configuration is disabled 1b - High Assurance Configuration is enabled
15-13 —	Reserved.
12-11 —	Reserved.
10 SW_LPSV	LP Software Security Violation When set, SecMon_LP treats this bit as a security violation. The LP secure data is zeroized or invalidated according to the configuration. This security violation may result in a system security monitor transition if the LP Security Violation is enabled in the SecMon_HP Security Violation Control Register.
9 SW_FSV	Software Fatal Security Violation When set, the system security monitor treats this bit as a fatal security violation. This security violation has no effect on the LP section. This command results only in the following transitions of the SSM: Check State -> Soft Fail Non-Secure State -> Soft Fail Trusted State -> Soft Fail Secure State -> Soft Fail
8 SW_SV	Software Security Violation When set, the system security monitor treats this bit as a non-fatal security violation. This security violation has no effect on the LP section. This command results only in the following transitions of the SSM: Check -> Non-Secure Trusted -> Soft Fail Secure -> Soft Fail
7-3 —	Reserved.
2	SSM Soft Fail to Non-Secure State Transition Disable

Table continues on the next page...

Field	Function
SSM_SFNS_DIS	When set, it disables the SSM transition from soft fail to non-secure state. Once set after the reset this bit cannot be changed 0b - Soft Fail to Non-Secure State transition is enabled 1b - Soft Fail to Non-Secure State transition is disabled
1 SSM_ST_DIS	SSM Secure to Trusted State Transition Disable When set, disables the SSM transition from secure to trusted state. Once set after the reset, this bit cannot be changed. 0b - Secure to Trusted State transition is enabled 1b - Secure to Trusted State transition is disabled
0 SSM_ST	SSM State Transition Transition state of the system security monitor. This self-clearing bit is always read as zero. This command results only in the following transitions of the SSM: Check State → Non-Secure (when Non-Secure Boot and not in Unsecure Configuration) Check State → Trusted (when Secure Boot or in Unsecure Configuration) Trusted State → Secure Secure State → Trusted (if not disabled by SSM_ST_DIS bit) Soft Fail → Non-Secure (if not disabled by SSM_SFNS_DIS bit)

29.9.4.4 SecMon_HP Security Interrupt Control Register (HPSICR)

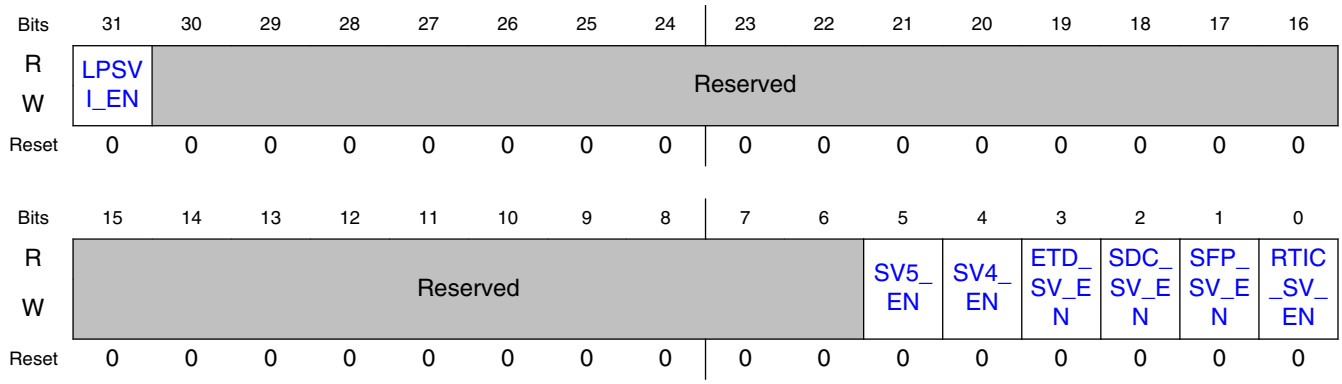
29.9.4.4.1 Offset

Register	Offset
HPSICR	Ch

29.9.4.4.2 Function

The HP Security Interrupt Control Register defines the SecMon security interrupt generation policy. This is a privileged write register.

29.9.4.4.3 Diagram



29.9.4.4.4 Fields

Field	Function
31 LPSVI_EN	LP Security Violation Interrupt Enable This bit enables generating of the security interrupt to the host processor upon security violation signal from the LP section. 0b - LP Security Violation Interrupt is Disabled 1b - LP Security Violation Interrupt is Enabled
30-6 Reserved	Reserved
5 SV5_EN	Central Security Unit Interrupt Enable Setting this bit to 1 enables generation of the security interrupt to the host processor upon detection of the Central Security Unit security violation. 0b - Central Security Unit Interrupt is Disabled 1b - Central Security Unit Interrupt is Enabled
4 SV4_EN	TrustZone-WatchDog Interrupt Enable Setting this bit to 1 enables generation of the security interrupt to the host processor upon detection of the TrustZone-WatchDog security violation. 0b - TrustZone-WatchDog Interrupt is Disabled 1b - TrustZone-WatchDog Interrupt is Enabled
3 ETD_SV_EN	External Tamper Detect Interrupt Enable Setting this bit to 1 enables generation of the security interrupt to the host processor upon detection of the External Tamper Detect security violation. 0b - External Tamper Detect Interrupt is Disabled 1b - External Tamper Detect Interrupt is Enabled
2 SDC_SV_EN	Secure Debug Controller Interrupt Enable Setting this bit to 1 enables generation of the security interrupt to the host processor upon detection of the Secure Debug Controller security violation. 0b - Secure Debug Controller Interrupt is Disabled

Table continues on the next page...

Field	Function
	1b - Secure Debug Controller Interrupt is Enabled
1 SFP_SV_EN	Security Fuse Processor Interrupt Enable Setting this bit to 1 enables generation of the security interrupt to the host processor upon detection of the Security Fuse Processor security violation. 0b - Security Fuse Processor Interrupt is Disabled 1b - Security Fuse Processor Interrupt is Enabled
0 RTIC_SV_EN	RTIC Interrupt Enable Setting this bit to 1 enables generation of the security interrupt to the host processor upon detection of the RTIC security violation. 0b - RTIC Interrupt is Disabled 1b - RTIC Interrupt is Enabled

29.9.4.5 SecMon_HP Security Violation Control Register (HPSVCR)

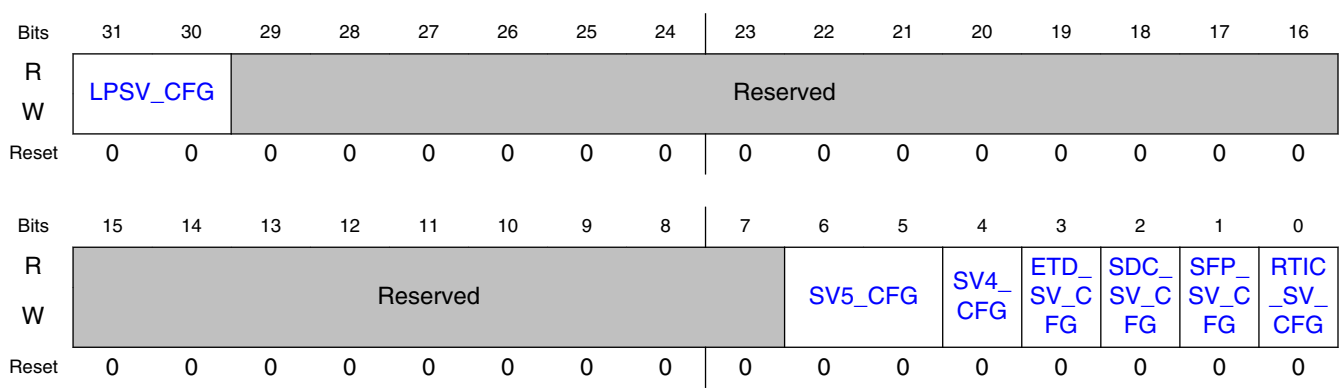
29.9.4.5.1 Offset

Register	Offset
HPSVCR	10h

29.9.4.5.2 Function

The HP Security Violation Control Register defines types for each security violation input. This is a privileged write register.

29.9.4.5.3 Diagram



29.9.4.5.4 Fields

Field	Function
31-30 LPSV_CFG	LP Security Violation Configuration This field configures the LP security violation source. 00b - LP security violation is disabled 01b - LP security violation is a non-fatal violation 1xb - LP security violation is a fatal violation
29-7 Reserved	Reserved.
6-5 SV5_CFG	Central Security Unit Security Violation Configuration This field configures the Central Security Unit Security Violation Input. This setting instructs the SSM how to respond upon detection of the Central Security Unit security violation. 00b - Central Security Unit is disabled 01b - Central Security Unit is a non-fatal violation 1xb - Central Security Unit is a fatal violation
4 SV4_CFG	TrustZone-WatchDog Security Violation Configuration This field configures the TrustZone-WatchDog Security Violation Input. This setting instructs the SSM how to respond upon detection of the TrustZone-WatchDog security violation. 0b - TrustZone-WatchDog is a non-fatal violation 1b - TrustZone-WatchDog is a fatal violation
3 ETD_SV_CFG	External Tamper Detect Security Violation Configuration This field configures the External Tamper Detect Security Violation Input. This setting instructs the SSM how to respond upon detection of the External Tamper Detect security violation. 0b - External Tamper Detect is a non-fatal violation 1b - External Tamper Detect is a fatal violation
2 SDC_SV_CFG	Secure Debug Controller Security Violation Configuration This field configures the Secure Debug Controller Security Violation Input. This setting instructs the SSM how to respond upon detection of the Secure Debug Controller security violation. 0b - Secure Debug Controller is a non-fatal violation 1b - Secure Debug Controller is a fatal violation
1 SFP_SV_CFG	Security Fuse Processor Security Violation Configuration This field configures the Security Fuse Processor Security Violation Input. This setting instructs the SSM how to respond upon detection of the Security Fuse Processor security violation. 0b - Security Fuse Processor is a non-fatal violation 1b - Security Fuse Processor is a fatal violation
0 RTIC_SV_CFG	RTIC Security Violation Configuration This field configures the RTIC Security Violation Input. This setting instructs the SSM how to respond upon detection of the RTIC security violation. 0b - RTIC is a non-fatal violation 1b - RTIC is a fatal violation

29.9.4.6 SecMon_HP Status Register (HPSR)

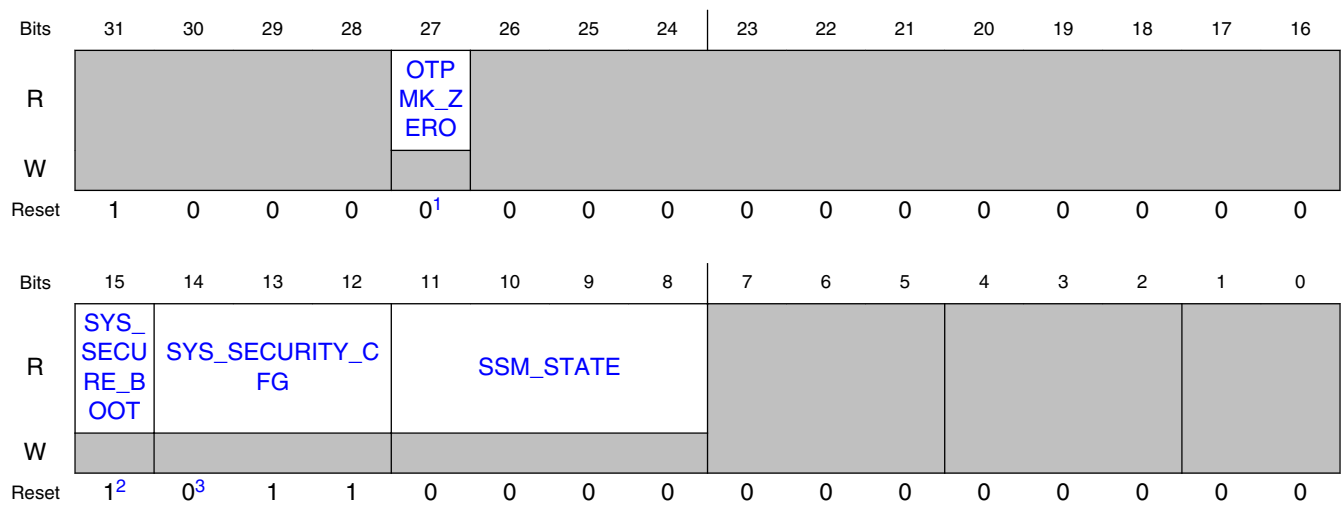
29.9.4.6.1 Offset

Register	Offset
HPSR	14h

29.9.4.6.2 Function

The HP Status Register reflects the internal state of the SecMon.

29.9.4.6.3 Diagram



1. OTPMK_ZERO will normally reset to 0. This will reset to 1 only if the OTPMK has not been programmed in the fuse bank.
2. In a chip in the field, SYS_SECURE_BOOT will normally reset to 1. It will reset to 0 only in a test chip.
3. SYS_SECURITY_CFG will normally reset to 011b in a chip in the field. It will reset to 000b only in a chip in the Fabrication facility. After the NXP-programmable fuses have been blown at the Fabrication facility SYS_SECURITY_CFG will reset to 001b. The OEM is expected to blow the OEM-programmable fuses, after which SYS_SECURITY_CFG will reset to 011b.

29.9.4.6.4 Fields

Field	Function
31-28 —	Reserved.
27 OTPMK_ZERO	One Time Programmable Master Key is Equal to Zero. When set, this bit always triggers "bad key" violation.

Table continues on the next page...

Field	Function
	0b - The OTPMK is not zero. 1b - The OTPMK is zero.
26-16 —	Reserved.
15 SYS_SECURE_BOOT	System Secure Boot If SYS_SECURE_BOOT is 1, the chip boots from internal ROM.
14-12 SYS_SECURITY_CFG	System Security Configuration This field indicates the security configuration of SecMon, defined as follows: 000b - Unsecured - the configuration of chips that can be booted without going through secure boot 010b - Intent to Secure - the configuration of chips that will only boot via secure boot
11-8 SSM_STATE	System Security Monitor State This field contains the encoded state of the SSM's state machine. The encoding of the possible states are: 0000b - Init 0001b - Hard Fail 0011b - Soft Fail 1000b - Init Intermediate (transition state between Init and Check - SSM stays in this state only one clock cycle) 1001b - Check 1011b - Non-Secure 1101b - Trusted 1111b - Secure
7-5 —	Reserved.
4-2 —	Reserved.
1-0 —	Reserved.

29.9.4.7 SecMon_HP Security Violation Status Register (HPSVSR)

29.9.4.7.1 Offset

Register	Offset
HPSVSR	18h

29.9.4.7.2 Function

The HP Security Violation Status Register reflects the HP domain security violation records. Write a 1 to Central Security Unit-0 to clear the corresponding security violation detection flag. Note that this does not automatically clear the security violation signal that is connected to the input, so the security violation may immediately be detected again

29.9.4.7.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	LP_SEC_VIO															
W																
Reset	1[1]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SW_LPSV	SW_FSV	SW_SV								SV5	SV4	ETD_SV	SDC_SV	SFP_SV	RTIC_SV
W											w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. This bit will reset to 1 on LP Section POR because the Power Glitch Detector will indicate that an LP section power glitch has occurred.

29.9.4.7.4 Fields

Field	Function
31 LP_SEC_VIO	LP Security Violation A security volation was detected in the SecMon low power section.
30-28 —	Reserved.
27-16 —	Reserved.
15 SW_LPSV	LP Software Security Violation This bit is a read-only copy of the SW_LPSV bit in the HP Command Register.
14 SW_FSV	Software Fatal Security Violation This bit is a read-only copy of the SW_FSV bit in the HP Command Register.
13 SW_SV	Software Security Violation This bit is a read-only copy of the SW_SV bit in the HP Command Register.
12-6	Reserved.

Table continues on the next page...

Field	Function
—	
5 SV5	Central Security Unit security violation was detected. 0b - No Central Security Unit security violation was detected. 1b - Central Security Unit security violation was detected.
4 SV4	TrustZone-WatchDog security violation was detected. 0b - No TrustZone-WatchDog security violation was detected. 1b - TrustZone-WatchDog security violation was detected.
3 ETD_SV	External Tamper Detect security violation was detected. 0b - No External Tamper Detect security violation was detected. 1b - External Tamper Detect security violation was detected.
2 SDC_SV	Secure Debug Controller security violation was detected. 0b - No Secure Debug Controller security violation was detected. 1b - Secure Debug Controller security violation was detected.
1 SFP_SV	Security Fuse Processor security violation was detected. 0b - No Security Fuse Processor security violation was detected. 1b - Security Fuse Processor security violation was detected.
0 RTIC_SV	RTIC security violation was detected. 0b - No RTIC security violation was detected. 1b - RTIC security violation was detected.

29.9.4.8 SecMon_HP High Assurance Counter IV Register (HPHACIVR)

29.9.4.8.1 Offset

Register	Offset
HPHACIVR	1Ch

29.9.4.8.2 Function

The SecMon_HP High Assurance Counter IV Register contains the initial value for the high assurance counter.

29.9.4.8.3 Diagram

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

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

29.9.4.8.4 Fields

Field	Function
31-0 HAC_COUNTER_IV	High Assurance Counter Initial Value This register is used to set the starting count value to the high assurance counter. This register cannot be programmed when HAC_L bit is set.

29.9.4.9 SecMon_HP High Assurance Counter Register (HPHACR)

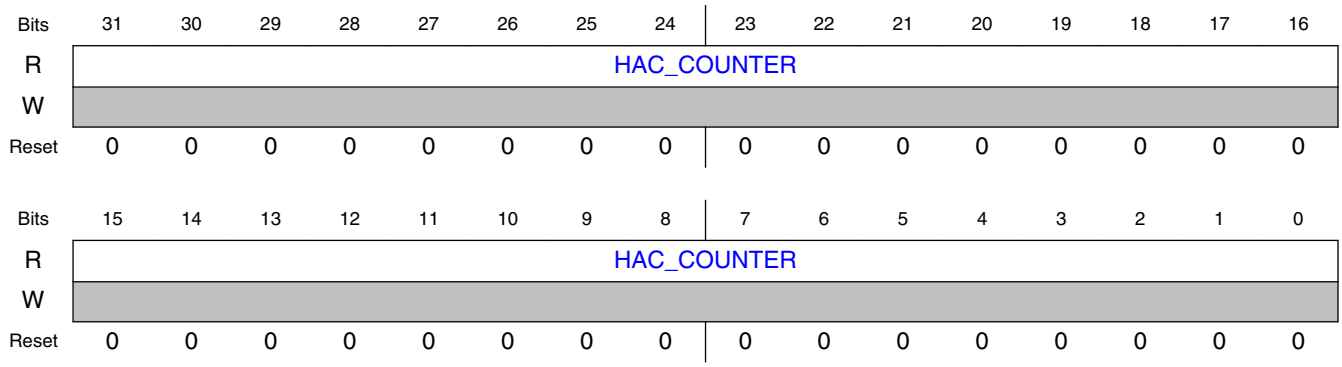
29.9.4.9.1 Offset

Register	Offset
HPHACR	20h

29.9.4.9.2 Function

The SecMon_HP High Assurance Counter Register contains the value of the high assurance counter. The high assurance counter is a delay introduced before the system security monitor transitions from soft fail to hard fail state if this transition is enabled.

29.9.4.9.3 Diagram



29.9.4.9.4 Fields

Field	Function
31-0 HAC_COUNTER	High Assurance Counter When the HAC_EN bit is set and the SSM is in the soft fail state, this counter starts to count down with the system clock. When the counter reaches zero, the SSM transitions to the Hard Fail State. • When HAC_STOP bit is set, the HAC Counter is stopped. • When HAC_CLEAR bit is set, the HAC Counter is cleared. • When HAC_LOAD bit is set, the HAC Counter is loaded with the value of the HPHACIVR.

29.9.4.10 SecMon_LP Power Glitch Detector Register (LPPGDR)

29.9.4.10.1 Offset

Register	Offset
LPPGDR	64h

29.9.4.10.2 Function

The SecMon_LP Power Glitch Detector Register is a 32-bit read/write register that is used for storing the power glitch detector value, as described in Power glitch detector (PGD).

29.9.4.10.3 Diagram

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

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

29.9.4.10.4 Fields

Field	Function
31-0 PGD	Power Glitch Detector Value

29.9.4.11 SecMon_HP Version ID Register 1 (HPVIDR1)

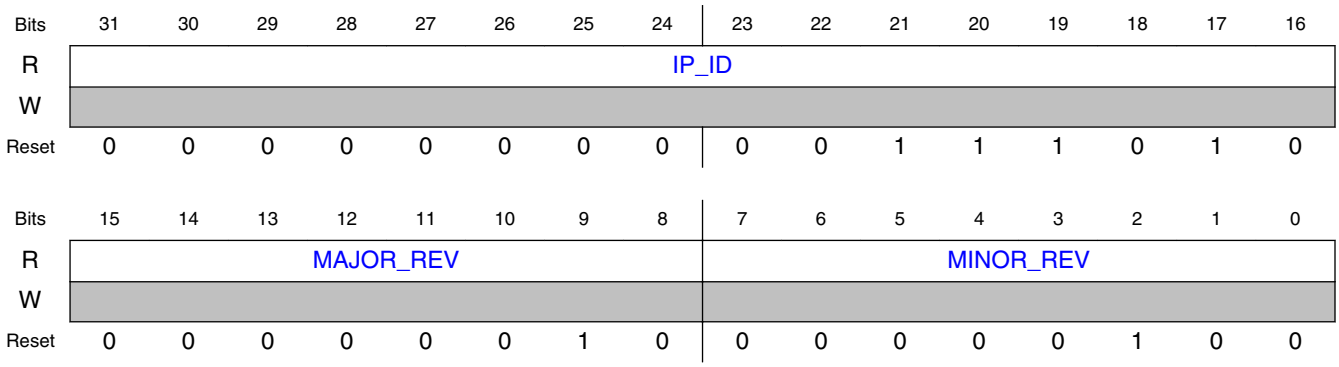
29.9.4.11.1 Offset

Register	Offset
HPVIDR1	BF8h

29.9.4.11.2 Function

The SecMon_HP Version ID Register 1 is a read-only register that contains the current version of the SecMon. The version consists of a module ID, a major version number, and a minor version number.

29.9.4.11.3 Diagram



29.9.4.11.4 Fields

Field	Function
31-16 IP_ID	SecMon block ID
15-8 MAJOR_REV	SecMon block major version number
7-0 MINOR_REV	SecMon block minor version number

29.9.4.12 SecMon_HP Version ID Register 2 (HPVIDR2)

29.9.4.12.1 Offset

Register	Offset
HPVIDR2	BFCh

29.9.4.12.2 Function

The SecMon_HP Version ID Register 2 is a read-only register that indicates the current version of the SecMon. Version ID register 2 consists of the following fields: integration options, ECO revision, and configuration options.

29.9.4.12.3 Diagram

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

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ECO_REV								CONFIG_OPT							
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

29.9.4.12.4 Fields

Field	Function
31-24 IP_ERA	IP Era 00h - Era 1 or 2 03h - Era 3 04h - Era 4 05h - Era 5
23-16 INTG_OPT	SecMon Integration Options
15-8 ECO_REV	SecMon ECO Revision
7-0 CONFIG_OPT	SecMon Configuration Options

29.9.5 Security Monitor functional description

Security Monitor incorporates a security state machine (SSM) that detects security violation inputs and manages the system security state in accordance with configured security policy.

The portion of the Security Monitor which is only on while the SoC is powered is referred to as the High Power (HP) section; the battery-backed portion is referred to as the Low Power (LP) section.

The following sections describe in detail the theory and basic design principles of the Security Monitor.

29.9.5.1 SM_HP description

The SM_HP contains all security monitor status and configuration registers. The configuration registers control security violation detection and response.

The SM_HP also incorporates System Security Monitor, Zeroizable Master Key programming mechanism and OTPMK logic.

29.9.5.1.1 System security monitor

The purpose of the system security monitor (SSM) is to monitor system security violations and control security state of the system.

The states and transitions of the security state machine and associated key permissions are shown in the following figure.

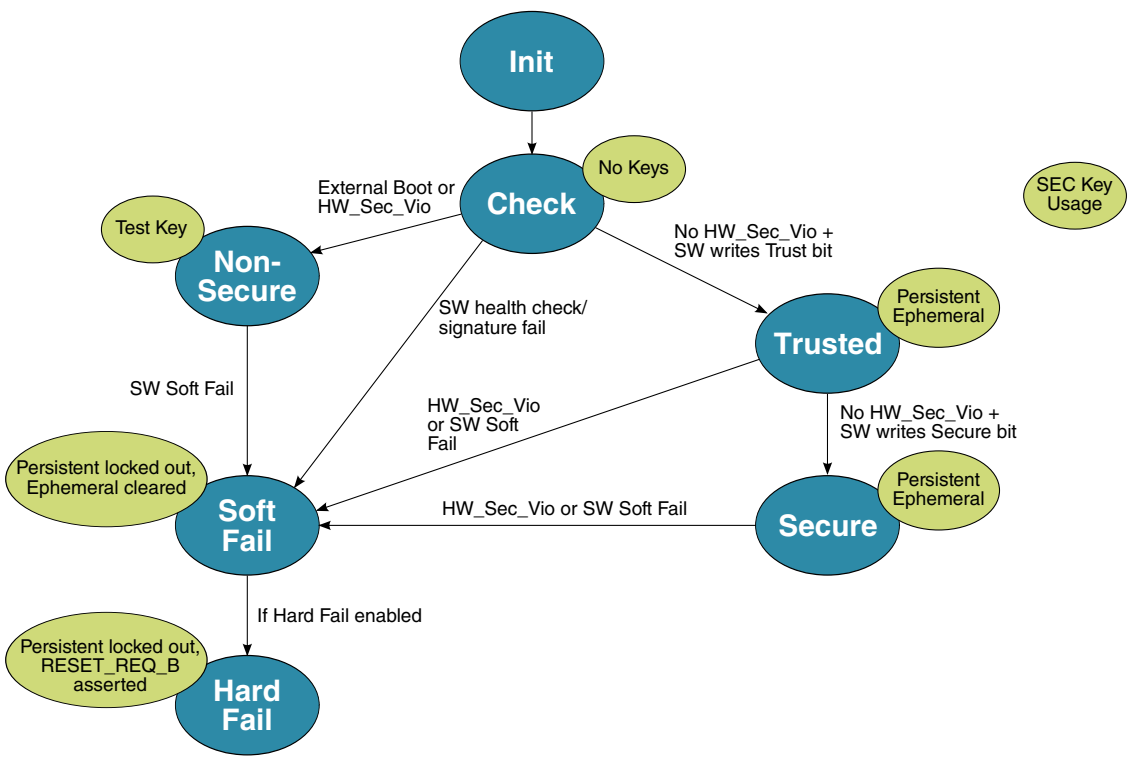


Figure 29-8. Security monitor states

29.9.5.1.2 State definitions

- **Init:** The Security Monitor SSM enters this state at Power On Reset. The SSM remains in this state until the reset logic determines that clocks are stable and fuse values can be read accurately. In this state the SSM ignores all security violations

sources; neither recording nor responding to them. The Security Monitor registers cannot be programmed in this state.

- **Check:** While in Check, the Security Monitor receives hardware inputs, including the fuse values from the SFP, security violations, and a pass/fail indication from the ISBC as described in the Secure boot sequence section.

  During Check, the Security Monitor determines if the booting core has begun execution in the IBR. If not, the SSM transitions to Non-Secure.

  Assuming the system is attempting secure boot, the setting of the ITS bit (see OEM Security Policy Register (SFP_OSPR)) determines the SSM's transition in the presence of hardware or software security violations.

  - If the ITS fuse is set, and any violation occurs or ISBC fails ESBC authentication, the Security Monitor transitions to Soft Fail.
  - If the ITS fuse is not set, and any violation occurs or ISBC fails ESBC authentication, the Security Monitor transitions to Non-Secure.
  - If there are no hardware security violations and ISBC validates the ESBC, the security monitor transitions to Trusted.
- **Non-Secure:** As noted above, this is the default state of the SSM whenever the device is not configured for secure boot. The fact that the Security Monitor is in the Non-Secure state is transparent to end users. If ITS=0 and secure boot is initiated by some other method, a secure boot failure transitions the SSM to this state, and allows the non-secure system to execute the suspect ESBC. Any requests by the ESBC for the SEC to use the OTPMK, ZMK, or KEK cause test keys (256 bits, all zeroes) to be used, rather than the real OTPMK , ZMK, or secret KEK. The master key input (OTPMK, ZMK, or the XOR of the two) to SEC is forced to be all zeros, but the KEK is not. The KEK is a random value that can be read or overwritten. This state exists mainly to support debug. Note: As discussed in Trust architecture overview, SecMon Non-Secure is not the same as TrustZone non-secure world.
- **Trusted:** If there are no hardware security violations and the ISBC validates the ESBC, the ISBC writes the SSM_ST bit in the HPCOMR and the Security Monitor SSM transitions to Trusted. In this state the Trusted and Secure mode indication signals are asserted to the SEC, and the SEC is allowed to use the provisioned OTPMK, ZMK, and secret KEKs.

  While in Trusted state, if there are no hardware security violations and the ESBC or subsequent trusted software writes the SSM_ST bit in the HPCOMR, the Security Monitor SSM transitions to Secure state.Note: As discussed in Trust architecture overview, SecMon Trusted is not the same as TrustZone Trusted Execution Environment.

- **Secure:** While in this state the Secure mode indication signal is asserted and the Trusted mode indication signal is de-asserted to the SEC.

If a security violation condition is detected the SSM immediately (without clock) transitions to the Soft Fail state. Transition from the Secure state back to the Trusted state can be triggered by software if this transition has not been disabled. In the device, there is limited difference between Trusted and Secure State. This distinction is more relevant in other NXP devices that also use the security monitor. Note: As discussed in Trust architecture overview, SecMon Secure is not the same as TrustZone secure world.

29.9.5.2 HP security violation policy

All HP security violation sources are classified into the following categories:

- Disabled violation: SSM does not react on this violation
- Non-fatal security violation: Results in the following transition of the SSM:
  - Check --> Non-Secure
  - Trusted --> Soft Fail
  - Secure --> Soft Fail
- Fatal security violation: SSM transitions to the Soft Fail state from Check, Non-Secure, Trusted, or Secure state

Regardless of the category, all security violation events are recorded in the corresponding status registers.

Table 29-3. SM_HP security violation sources

Security violation source	Default behavior	Configuration options[1]	Comments
Scan exit violation	Enable	-	Asserted upon scan exit and stays active until system reset
Software fatal violation	Enable	-	Asserted by software
Software non-fatal violation	Non-fatal	-	Asserted by software
Bad master key violation	Non-Fatal	-	Asserted when: - OTPMK does not pass validity check - ZMK ECC Check failure (if ZMK ECC Check is enabled) - ZMK is zero when it is selected for use
Security violation Inputs 0-4	Non-fatal	• Non-fatal • Fatal	Asserted on the security violation input ports 0-4

1. The configuration is set in the HP security violation control register (HPSVCR). Once set the configuration can be locked to prevent further changes.

29.9.5.3 Master key checking and control

As mentioned in Trust architecture overview, the Trust Architecture has secrets. Specifically it allows OEMs to provision an AES-256 key which is usable by the SEC, and only when the SecMon is in the Trusted or Secure state. This AES-256 key is called the Master Key, and as shown in figure 10-4, the Master Key can be generated by a few different methods.

- One Time Programmable Master Key (OTPMK): OEMs can permanently program an OTPMK into the SFP's OTPMK registers. The original value of the OTPMK can be determined by any method the OEM decides, however the value programmed into the OTPMK registers must embed a hamming value in order to allow the SecMon to perform hardware key checking. Because the OTPMK can't be read back after fuse programming, a hardware hamming check is the only way to identify issues arising from a fuse programming error or fuse reliability. To use the Trust Architecture, a valid OTPMK must be programmed into the SFP, otherwise the SecMon's bad key check will report an error which prevents the chip from reaching the Trusted/Secure state.
- Zeroizable Master Key (ZMK): OEMs have the option to provision a ZMK in the ZMK registers of the battery backed low power (LP) section of the SecMon. Unlike the OTPMK, which once installed, can never be erased, only locked out, the ZMK can be erased in response to fatal security violations. OEMs have two options for provisioning a ZMK.
  - Software Programming: As shown in the figure, OEMs can directly write to the SecMon's ZMK registers. The original value of the ZMK can be determined by any method the OEM decides, additional ECC protection bits are automatically generated and stored along with the ZMK to detect any future corruption when the ZMK_Valid bit is set in the LP Master Key Control Register. See section 10.8.8 for details of setting and locking the ZMK with software programming.
  - Hardware Programming: : As shown in the figure, writing to the SecMon's HP_COM Register will cause the SecMon to load a 256b random value from the SEC's random number generator and load it into the ZMK registers. Additional ECC protection bits are automatically generated and stored along with the ZMK to detect any future corruption. See section 10.8.8 for details of setting and locking the ZMK with software programming.
- Combined Master Key (CMK): Via the LP Master Key Control Register, the Master Key can be taken from the OTPMK registers, ZMK registers, or it can be formed by the bitwise XOR of the two. This bitwise XOR is called the CMK.

Note that the Master Key is only used by the SEC to encrypt and decrypt blob keys. It is never used directly on user selected data. More information on blob operations, see the SEC Reference Manual. Additionally, the exact value of the Master Key during blob key encrypt/decrypt operations changes depending on the state of the SecMon at the time of

the operation, and the SEC interface used to submit the blob operation. This allows different Trusted Partitions (each ‘owning’ a different job submission interface) to create blobs which the other cannot decrypt, despite them both being trusted to create their own blobs. This also allows TZ secure world to create blobs which TZ non-secure world can’t decrypt, meaning the OEM must be aware of the SSM state and job submission interface at the time the blob is created.

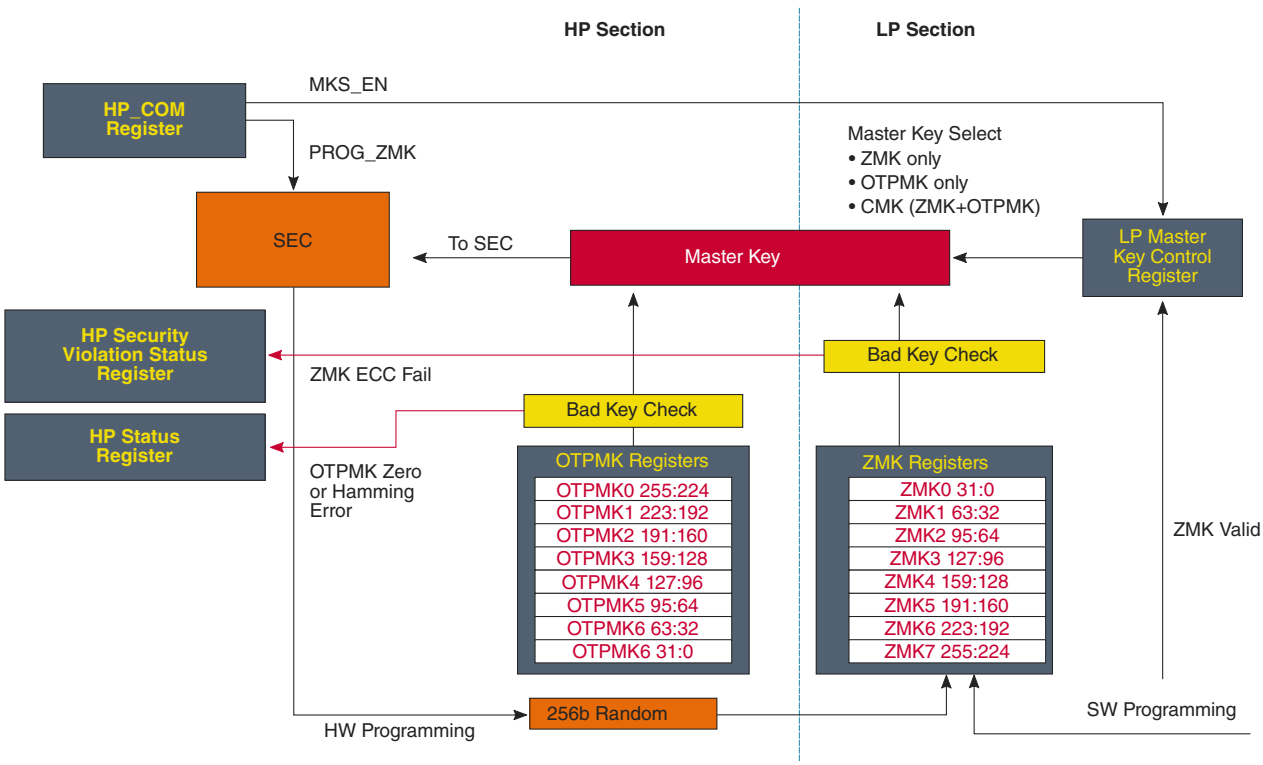


Figure 29-9. Master Key Selection and Error Checking

29.9.6 Initialization guidelines

The following steps should be completed to properly initialize the Security Monitor (following successful completion of secure boot):

1. Transition SSM from trusted state to secure state (if not already done by ESBC). The power glitch detection is always enabled; however, the Security Monitor's response to receiving a signal from the PGD circuitry is controlled by HPSVCR[LPSV_CFG]. Immediately after LP POR the state of this register is zero and power glitch violation is asserted. Therefore, before programming any feature in Security Monitor module it is recommended to set the proper value into the LPPGDR and to clear the power glitch record in the LP status register. See Power glitch detector (PGD) for more details.

2. User-specific: Enable security violations and interrupts in HPSVCR and HPSICR registers
3. User-specific: Program Security Monitor general functions/configurations
4. User-specific: Select Master Key (default is OTPMK)
5. User-specific: Set lock bits. Bit 31 (reserved) of the SM_HP lock register should be set before starting anyfunctional operation. This bit disables the use of alternate keys, which are not supported in the device. See SM_HP Lock Register (SECMON_HPLR) , for details on the SM_HP lock register.

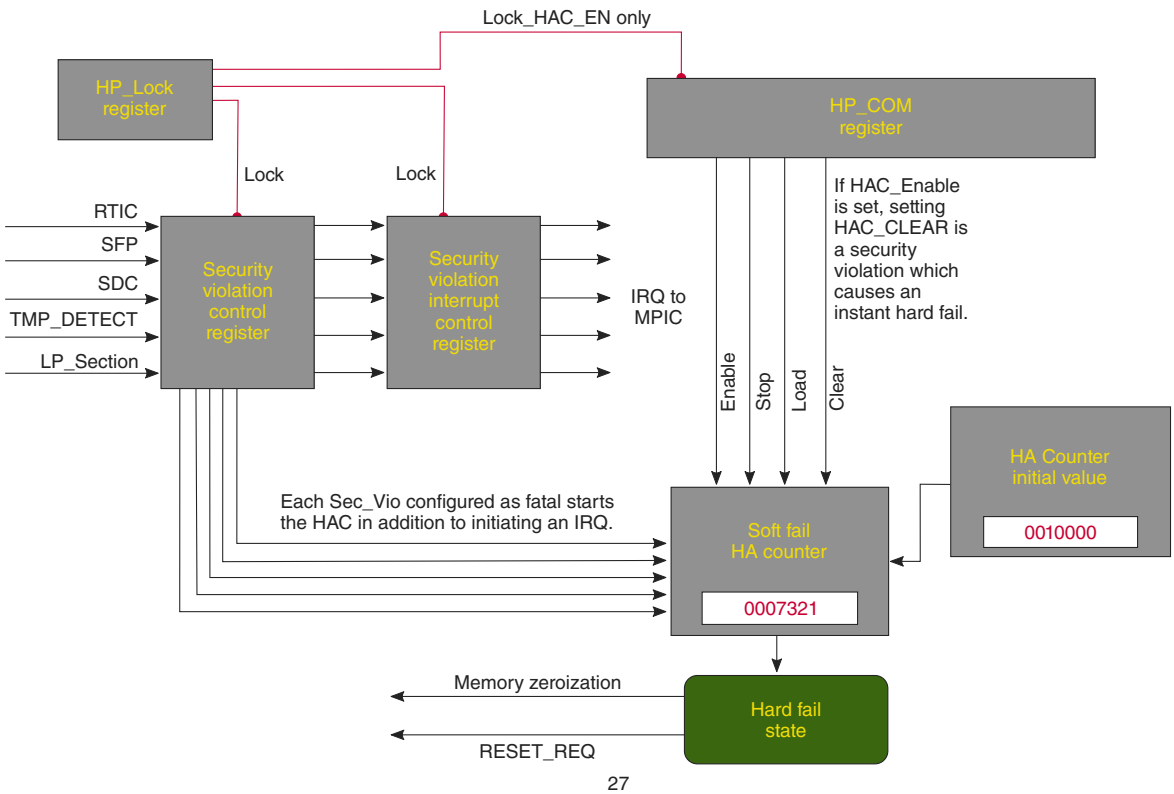


Figure 29-10. Relationship Between the Registers

29.9.7 Zeroizable master key programming guidelines

The Zeroizable Master Key (ZMK) can be programmed either by software or by hardware. Refer to Master key checking and control for the description below.

To program ZMK by software do the following:

- Verify that SECMON_LPMKCR [ZMK_HWP] is not set
- Verify that the ZMK registers are not locked for reads and writes. Check SECMON_HPLR [ZMK_WSL], [ZMK_RSL] or SECMON_LPLR [ZMK_WHL], [ZMK_WHL] are not set.

- Write key value to the ZMK registers
- Verify that the intended key value is written
  - Set SECMON_LPMKCR [ZMK_VAL] to indicate the ZMK is ready to be used the SEC.
  - (Optional but recommended) Set SECMON_LPMKCR [ZMK_ECC_EN] to enable ZMK error correction code verification. Software can verify that the correct nine bit codeword is generated by reading ZMK_ECC_VALUE field
- (Optional but recommended) Block software read accesses to the ZMK registers and ZMK_ECC_VALUE field by setting SECMON_HPLR [ZMK_RSL] and SECMON_LPLR [ZMK_RHL]
- (Optional but recommended) Block software write accesses to the ZMK registers by setting SECMON_HPLR [ZMK_WSL] and SECMON_LPLR [ZMK_WHL]
- Set SECMON_HPCOMR [MKS_EN] and SECMON_LPMKCR [MASTER_KEY_SEL] to select combination of OTPMK and ZMK to be provided to the SEC
- (Optional but recommended) Block software write accesses to the MASTER_KEY_SEL field by setting MKS lock bit
- (Optional but recommended) Block software write accesses to SECMON_LPMKCR [MASTER_KEY_SEL] by setting SECMON_HPLR [MKS_SL].

To program ZMK by hardware do the following:

- Verify that the ZMK registers are not locked for writes. Check that SECMON_HPLR [ZMK_WSL], or SECMON_LPLR [ZMK_WHL] are not set
- Set SECMON_LPMKCR [ZMK_HWP]
- Set SECMON_HPCOMR [PROG_ZMK]
- Poll for SECMON_LPMKCR [ZMK_VAL] bit to be set. This bit is set by hardware at the end of the ZMK programming cycle
- (Optional but recommended) Set SECMON_LPMKCR [ZMK_ECC_EN] to enable ZMK error correction code verification by hardware. Note that the ZMK Registers and ZMK_ECC_VALUE field cannot be read by software in the hardware programming mode, meaning hardware ZMK_ECC checking is the only way to determine if the ZMK is corrupted.
- (Optional but recommended) Block hardware programming option of the ZMK Registers by setting SECMON_LPLR [ZMK_WHL]
- Set SECMON_HPCOMR [MKS_EN] and SECMON_LPMKCR [MASTER_KEY_SEL] to select combination of OTPMK and ZMK to be provided to the SEC
- (Optional but recommended) Block software write accesses to SECMON_LPMKCR [MASTER_KEY_SEL] by setting SECMON_HPLR [MKS_SL].

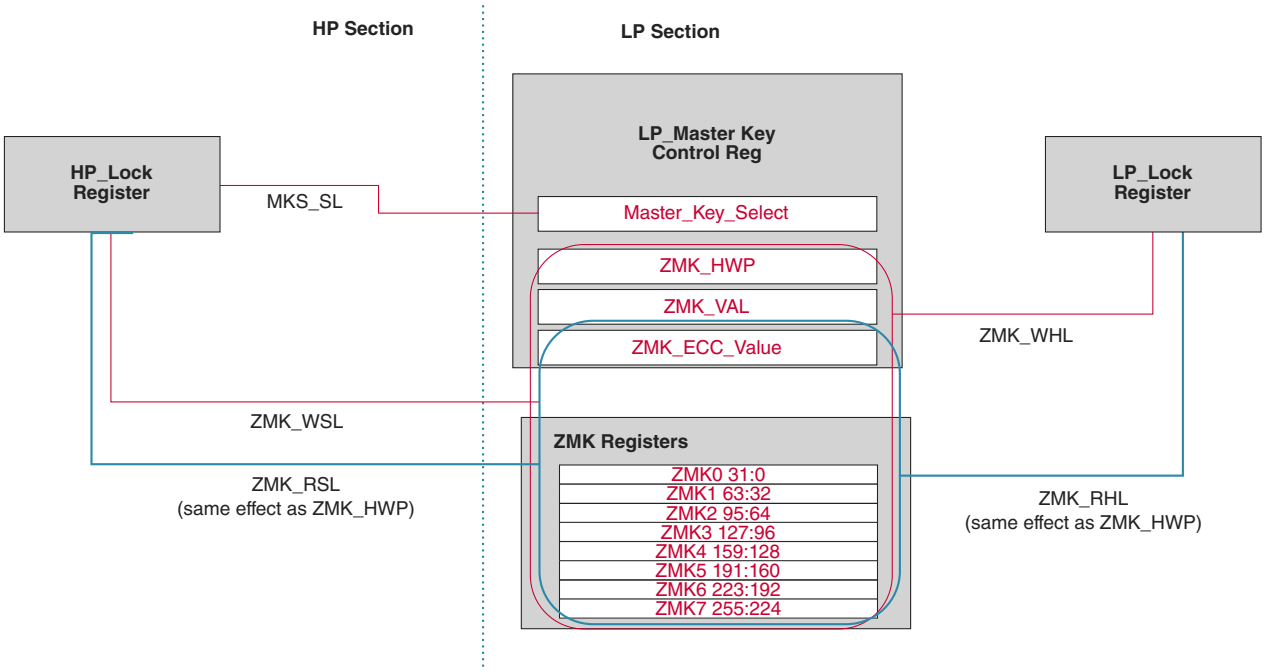


Figure 29-11. Locking the Key Selection and ZMK Registers

29.10 Manufacturing Protection

The Trust Architecture 2.1 supports a manufacturing protection feature OEMs may use to gain the following assurances:

- The OEM is building systems using legitimate NXP chips of the correct type.
- The contract manufacturer is burning the chip's fuses correctly (especially the SRKH)
- The system has booted securely and authorized OEM software is running on the chip prior to provisioning of credentials
- OEM additional secrets can safely be downloaded to the chip

This section provides an overview of this optional capability implemented through a combination of the ISBC and SEC. The details of SEC descriptors for manufacturing protection routines can be found in the SEC Programmer's Reference Manual.

The manufacturing protection process includes steps taken at the OEM once per production lot and steps taken at the contract manufacturer (or upon field installation).

29.10.1 OEM actions per production lot

As shown in Figure, the following steps occur in the OEM's development lab once per production lot, where the definition of a production lot is all the devices sharing the same super root keys (and accordingly the SRKH):

1. The OEM burns the super root key hash into fuses.
2. The OEM invokes the private key derivation function built into the chip. The private key is derived from the SRKH and a NXP secret value built into the device.
3. Once the private key has been generated, another SEC routine is run that generates a complementary public key to the private key that is stored inside the device. The public key is exported from the device and stored by the OEM for future reference.

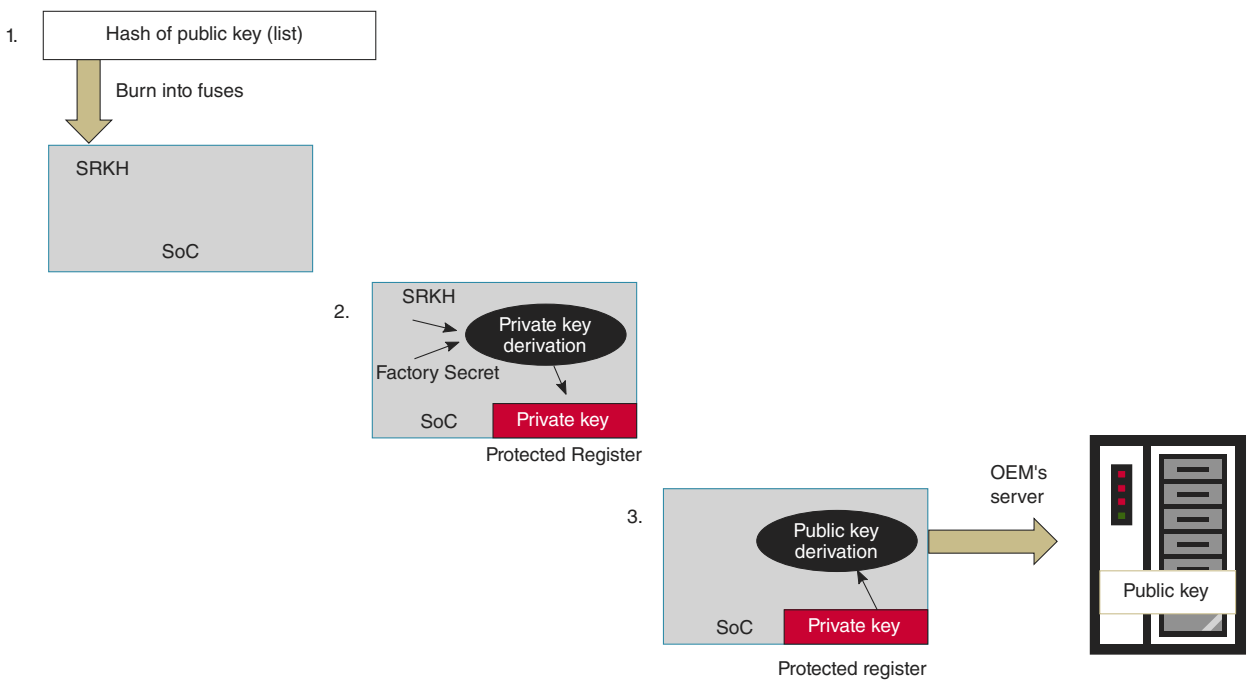


Figure 29-12. OEM actions per production lot

29.10.2 Contract manufacturer actions per system

As shown in figure below, the following steps occur at the contract manufacturer's site once per manufactured system:

1. The manufacturer customizes the chip by burning the hash of the OEM's SRKH into the fuses of each chip.

2. The chip executes secure boot, then executes OEM-supplied software that contacts the OEM's server and delivers a message signed with the chip's derived private key. The message should include an untamperable unique identifier per system; the factory (NXP) Unique ID from the SFP is intended to support this, but a system level unique ID could also be used.

3. The OEM's server authenticates the message using the previously stored public key. The server logs which unique identifiers it has received messages from, and if the system is considered valid to receive credentials, the OEM server provides them. If the credentials contain secret values, the system and OEM server can set up an encrypted tunnel to exchange them. The system should leverage the chip's blob encryption capability to securely store the credentials in non-volatile memory.

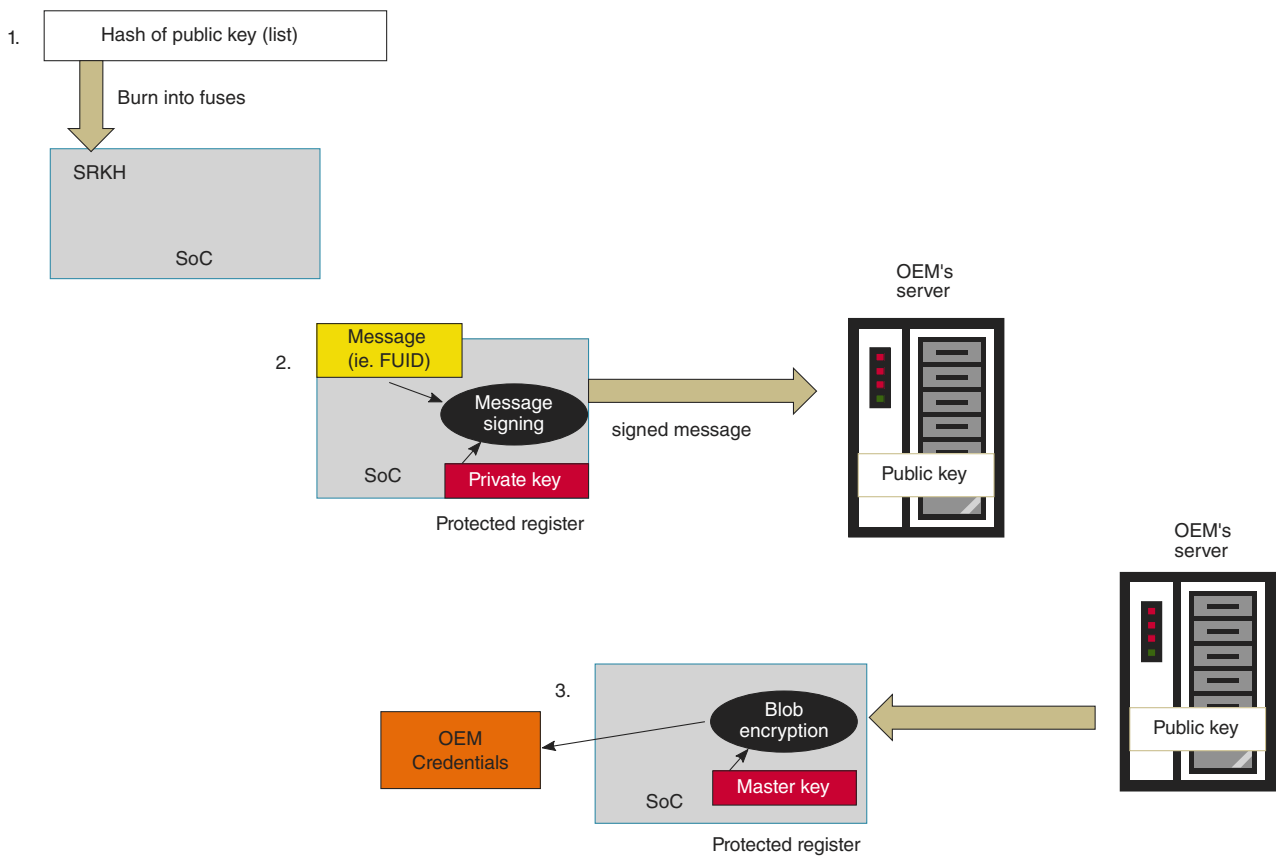


Figure 29-13. Contract manufacturer actions per system

Chapter 30
Serial Peripheral Interface (SPI)

30.1 The SPI module as implemented on the chip

This section provides details about how the SPI module is implemented on the chip.

30.1.1 LS1012A SPI module integration

The following table describes the SPI module integration into this chip:

Table 30-1. SPI module integration

Module	Module Base address
SPI	210_0000

30.1.2 LS1012A SPI signals

The following table lists the SoC signal names and their corresponding SPI module signal names used in this chapter:

Table 30-2. LS1012A SPI signals

LS1012A signal name	SPI module signal
SPI_CS_B[0:2][1]	PCS[0:2]
SPI_CLK	SCK
SPI_MISO	SIN
SPI_MOSI	SOUT

1. By default, this signal is active high. The active low state of this signal is register controlled.

30.1.3 LS1012A SPI module special consideration

The SPI module implements the following parameter settings in the chip:

Table 30-3. LS1012A SPI parameter settings

SPI parameters	LS1012A parameter value
NUM_CTAR_PP	4
TX_FIFO_DEPTH_PP	16
RX_FIFO_DEPTH_PP	16
No. of SPI controllers	1
SP1 Chip Selects	3
Slave mode support	No
Stop mode support	Yes. Refers to the LPM20 low power mode of the chip.
Debug mode support	No
MTFE mode support	No

NOTE

Any reference to the features that are not supported in the chip should be ignored.

30.1.4 SPI_MCR register mapping

The following table shows the SPI_MCR register reset configuration in the chip.

Table 30-4. SPI_MCR register mapping

Absolute address (hex)	Register name	Reset value
210_0000	Module Configuration Register (SPI_MCR)	80004001h

30.2 Introduction

The serial peripheral interface (SPI) module provides a synchronous serial bus for communication between a chip and an external peripheral device.

30.2.1 Block Diagram

The block diagram of this module is as follows:

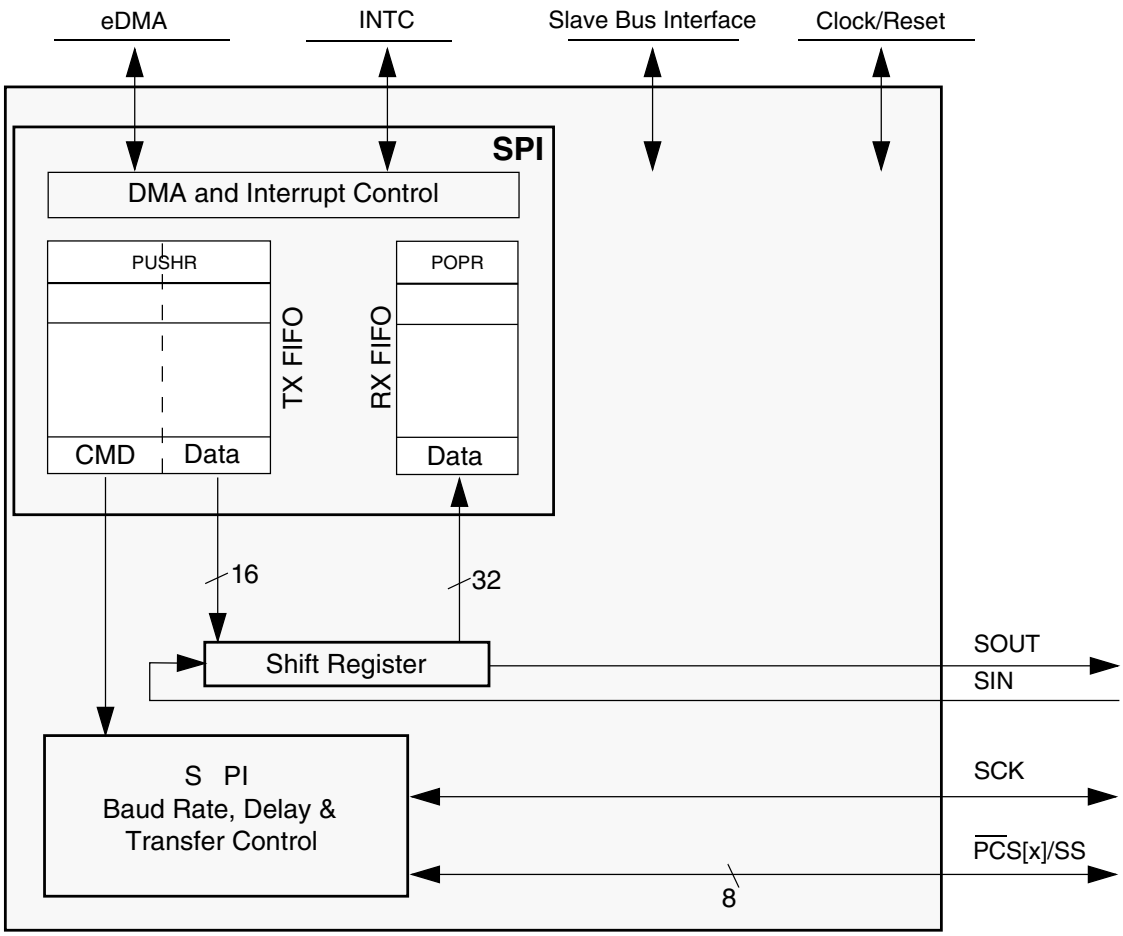


Figure 30-1. SPI Block Diagram

30.2.2 Features

The module supports the following features:

- Full-duplex, three-wire synchronous transfers
- Master mode
- Buffered transmit operation using the transmit first in first out (TX FIFO) with depth of 16 entries
- Support for 8/16-bit accesses to the PUSH TX FIFO Register Data Field
- Buffered receive operation using the receive FIFO (RX FIFO) with depth of 16 entries
- TX and RX FIFOs can be disabled individually for low-latency updates to SPI queues
- Visibility into TX and RX FIFOs for ease of debugging
- Programmable transfer attributes on a per-frame basis:

- 4 transfer attribute registers
- 4 extended transfer attribute registers
- Serial clock (SCK) with programmable polarity and phase
- Various programmable delays
- Programmable serial frame size: 4 to 32
  - SPI frames longer than 32 bits can be supported using the continuous selection format.
- Continuously held chip select capability
- Parity control

- 6 peripheral chip selects (PCSes), expandable to 6 with external demultiplexer
- Deglitching support for up to 6 peripheral chip selects (PCSes) with external demultiplexer
- DMA support for adding entries to TX FIFO and removing entries from RX FIFO:
  - TX FIFO is not full (TFFF)
  - RX FIFO is not empty (RFDF)
  - CMD FIFO is not full (CMDFFF)
- Interrupt conditions:
  - End of Queue reached (EOQF)
  - TX FIFO is not full (TFFF)
  - CMD FIFO is not full (CMDFFF)
  - Transfer of current frame complete (TCF)
  - Transfers due from current command frame complete (CMDTCF)
  - Attempt to transmit with an empty Transmit FIFO (TFUF)
  - RX FIFO is not empty (RFDF)
  - Frame received while Receive FIFO is full (RFOF)
  - SPI Parity Error (SPEF)
  - Data present in TX FIFO while CMD FIFO is empty (TFIWF)
- Global interrupt request line
- Power-saving architectural features:

- Support for Stop mode
- Support for Doze mode

30.2.3 Interface configurations

30.2.3.1 SPI configuration

The Serial Peripheral Interface (SPI) configuration allows the module to send and receive serial data. This configuration allows the module to operate as a basic SPI block with internal FIFOs supporting external queue operation. Transmitted data and received data reside in separate FIFOs. The host CPU or a DMA controller read the received data from the Receive FIFO and write transmit data to the Transmit FIFO.

For queued operations, the SPI queues can reside in system RAM, external to the module. Data transfers between the queues and the module FIFOs are accomplished by a DMA controller or host CPU. The following figure shows a system example with DMA, SPI, and external queues in system RAM.

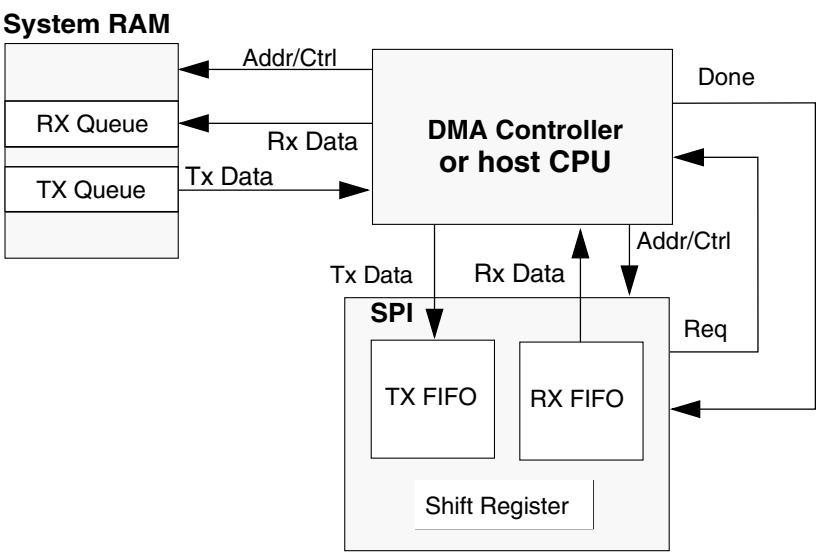


Figure 30-2. SPI with queues and DMA

30.2.4 Modes of Operation

The module supports the following modes of operation that can be divided into two categories:

- Module-specific modes:

    - Master mode
    - Module Disable mode
- Chip-specific modes:
    - External Stop mode

The module enters module-specific modes when the host writes a module register. The chip-specific modes are controlled by signals external to the module. The chip-specific modes are modes that a chip may enter in parallel to the block-specific modes.

30.2.4.1 Master Mode

Master mode allows the module to initiate and control serial communication. In this mode, these signals are controlled by the module and configured as outputs:

- SCK
- SOUT
- PCS[*x*]

30.2.4.2 Module Disable Mode

The Module Disable mode can be used for chip power management. The clock to the non-memory mapped logic in the module can be stopped while in the Module Disable mode.

30.2.4.3 External Stop Mode

External Stop mode is used for chip power management. The module supports the Peripheral Bus Stop mode mechanism. When a request is made to enter External Stop mode, it acknowledges the request and completes the transfer that is in progress. When the module reaches the frame boundary, it signals that the protocol clock to the module may be shut off.

30.3 Module signal descriptions

This table describes the signals on the boundary of the module that may connect off chip (in alphabetical order).

Table 30-5. Module signal descriptions

Signal	Master mode	Slave mode	I/O
PCS0	Peripheral Chip Select 0 (O)		I/O
PCS[1:3]	Peripheral Chip Selects 1–3	(Unused)	O
PCS4	Peripheral Chip Select 4	(Unused)	O
PCS5/ PCSS_b	Peripheral Chip Select 5 /Peripheral Chip Select Strobe	(Unused)	O
SCK	Serial Clock (O)	Serial Clock (I)	I/O
SIN	Serial Data In	Serial Data In	I
SOUT	Serial Data Out	Serial Data Out	O

30.3.1 PCS0—Peripheral Chip Select

Master mode: Peripheral Chip Select 0 (O)—Selects an SPI slave to receive data transmitted from the module.

NOTE

Do not tie the SPI slave select pin to ground. Otherwise, SPI cannot function properly.

30.3.2 PCS1–PCS3—Peripheral Chip Selects 1–3

Master mode: Peripheral Chip Selects 1–3 (O)—Select an SPI slave to receive data transmitted by the module.

30.3.3 PCS4—Peripheral Chip Select 4

Master mode: Peripheral Chip Select 4 (O)—Selects an SPI slave to receive data transmitted by the module.

30.3.4 PCS5/PCSS_b—Peripheral Chip Select 5/Peripheral Chip Select Strobe

Master mode:

- Peripheral Chip Select 5 (O)—Used only when the peripheral-chip-select strobe is disabled (MCR[PCSSE]). Selects an SPI slave to receive data transmitted by the module.
- Peripheral Chip Select Strobe (O)—Used only when the peripheral-chip-select strobe is enabled (MCR[PCSSE]). Strobes an off-module peripheral-chip-select demultiplexer, which decodes the module's PCS signals other than PCS5, preventing glitches on the demultiplexer outputs.

30.3.5 SCK—Serial Clock

Master mode: Serial Clock (O)—Supplies a clock signal from the module to SPI slaves.

30.3.6 SIN—Serial Input

Master mode: Serial Input (I)—Receives serial data.

30.3.7 SOUT—Serial Output

Master mode: Serial Output (O)—Transmits serial data.

30.4 SPI register descriptions

Register accesses to memory addresses that are reserved or undefined result in a transfer error. Any Write access to the POPR and RXFRn also results in a transfer error.

NOTE

While the module is in the running state, do not write to these registers:

- PISRn
- DIMRn,DPIRn,and SSRn
- CTAREn
- TRIG
- TSL and TS_CONF

30.4.1 SPI Memory map

SPI base address: 210_0000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Module Configuration Register (MCR)	32	RW	0000_4001h
8h	Transfer Count Register (TCR)	32	RW	0000_0000h
Ch - 18h	Clock and Transfer Attributes Register (In Master Mode) (CTAR0 - CTAR3)	32	RW	7800_0000h
2Ch	Status Register (SR)	32	W1C	0201_0000h
30h	DMA/Interrupt Request Select and Enable Register (RSER)	32	RW	0000_0000h
34h	PUSH TX FIFO Register In Master Mode (PUSHR)	32	RW	0000_0000h
38h	POP RX FIFO Register (POPR)	32	RO	0000_0000h
3Ch - 78h	Transmit FIFO Registers (TXFR0 - TXFR15)	32	RO	0000_0000h
7Ch - B8h	Receive FIFO Registers (RXFR0 - RXFR15)	32	RO	0000_0000h
11Ch - 128h	Clock and Transfer Attributes Register Extended (CTARE0 - CTAR E3)	32	RW	0000_0001h
13Ch	Status Register Extended (SREX)	32	RO	0000_0000h

30.4.2 Module Configuration Register (MCR)

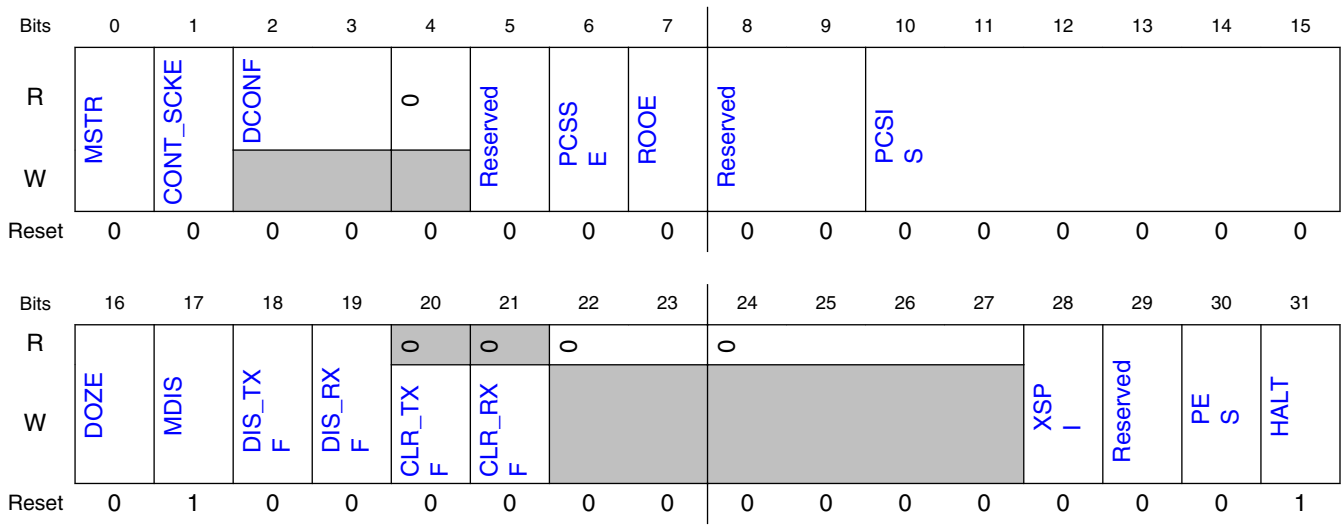
30.4.2.1 Offset

Register	Offset
MCR	0h

30.4.2.2 Function

Contains bits to configure various attributes associated with the module operations. The HALT and MDIS bits can be changed at any time, but the effect takes place only on the next frame boundary. Only the HALT and MDIS bits in the MCR can be changed, while the module is in the Running state.

30.4.2.3 Diagram



30.4.2.4 Fields

Field	Function
0 MSTR	Master/Slave Mode Select Enables either Master mode (if supported) or Slave mode (if supported) operation. 0b - Reserved 1b - Enables Master mode
1 CONT_SCKE	Continuous SCK Enable Enables the Serial Communication Clock (SCK) to run continuously. 0b - Continuous SCK disabled. 1b - Continuous SCK enabled.
2-3 DCONF	SPI Configuration. Selects among the different configurations of the module. 00b - SPI 01b - Reserved 10b - Reserved 11b - Reserved
4 —	Reserved
5 —	Reserved
6 PCSSE	Peripheral Chip Select Strobe Enable Enables the PCS5/ PCSS_b to operate as a PCS Strobe output signal. 0b - PCS5/PCSS_b is used as the Peripheral Chip Select[5] signal. 1b - PCS5/PCSS_b is used as an active-low PCS Strobe signal.
7	Receive FIFO Overflow Overwrite Enable

Table continues on the next page...

Field	Function
ROOE	In the RX FIFO overflow condition, configures the module to ignore the incoming serial data or overwrite existing data. If the RX FIFO is full and new data is received, the data from the transfer, generating the overflow, is ignored or shifted into the shift register. 0b - Incoming data is ignored. 1b - Incoming data is shifted into the shift register.
8-9 —	Always write the reset value to this field.
10-15 PCSIS	Peripheral Chip Select x Inactive State Determines the inactive state of PCSx. **NOTE:** The effect of this bit only takes place when module is enabled. Ensure that this bit is configured correctly before enabling the SPI interface. 000000b - The inactive state of PCSx is low. 000001b - The inactive state of PCSx is high.
16 DOZE	Doze Enable Provides support for an externally controlled Doze mode power-saving mechanism. 0b - Doze mode has no effect on the module. 1b - Doze mode disables the module.
17 MDIS	Module Disable Allows the clock to be stopped to the non-memory mapped logic in the module effectively putting it in a software-controlled power-saving state. The reset value of the MDIS bit is parameterized, with a default reset value of 1. 0b - Enables the module clocks. 1b - Allows external logic to disable the module clocks.
18 DIS_TXF	Disable Transmit FIFO When the TX FIFO is disabled, the transmit part of the module operates as a simplified double-buffered SPI. This bit can be written only when the MDIS bit is cleared. 0b - TX FIFO is enabled. 1b - TX FIFO is disabled.
19 DIS_RXF	Disable Receive FIFO When the RX FIFO is disabled, the receive part of the module operates as a simplified double-buffered SPI. This bit can only be written when the MDIS bit is cleared. 0b - RX FIFO is enabled. 1b - RX FIFO is disabled.
20 CLR_TXF	Clear TX FIFO Flushes the TX FIFO. Writing a 1 to CLR_TXF clears the TX FIFO Counter. The CLR_TXF bit is always read as zero. 0b - Do not clear the TX FIFO counter. 1b - Clear the TX FIFO counter.
21 CLR_RXF	CLR_RXF Flushes the RX FIFO. Writing a 1 to CLR_RXF clears the RX Counter. The CLR_RXF bit is always read as zero. **NOTE:** After every RX FIFO clear operation (MCR [CLR_RXF] = 0b1) following a RX FIFO overflow (SR [RFOF] = 0b1) scenario, immediately perform a single POP from the RX FIFO and discard the read data. The POP and discard operation should be completed before the reception of new incoming frame. 0b - Do not clear the RX FIFO counter. 1b - Clear the RX FIFO counter.
22-23	Reserved

Table continues on the next page...

Field	Function
—	
24-27 —	Reserved
28 XSPI	Extended SPI Mode This bit enables usage of CTARE (Command and Transfer Attribute Register Extended) Registers. CTARE registers allow the user to send up to 32 bit SPI frames. Command Cycling is also enabled which allows the user to send multiple Data Frames using a single Command Frame. When MCR[DIS_TXF] is asserted, the Extended SPI Mode cannot be used to transmit SPI frames which are more than 16 bits in size. 0b - Normal SPI Mode. Frame size can be up to 16 bits. Command Cycling is not available in this mode. 1b - Extended SPI Mode. Up to 32 bit SPI Frames along with Command Cycling is Enabled.
29 —	Reserved
30 PES	Parity Error Stop Controls SPI operation when a parity error is detected in a received SPI frame. 0b - SPI frame transmission continues. 1b - SPI frame transmission stops.
31 HALT	Halt The HALT bit starts and stops frame transfers. See Start and Stop of Module transfers 0b - Start transfers. 1b - Stop transfers.

30.4.3 Transfer Count Register (TCR)

30.4.3.1 Offset

Register	Offset
TCR	8h

30.4.3.2 Function

TCR contains a counter that indicates the number of SPI transfers made. The transfer counter is intended to assist in queue management. Do not write the TCR when the module is in the Running state.

30.4.3.3 Diagram

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

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

30.4.3.4 Fields

Field	Function
0-15 SPI_TCNT	SPI Transfer Counter Counts the number of SPI transfers the module makes. The SPI_TCNT field increments every time the last bit of an SPI frame is transmitted. A value written to SPI_TCNT presets the counter to that value. SPI_TCNT is reset to zero at the beginning of the frame when the CTCNT field is set in the executing SPI command. The Transfer Counter wraps around; incrementing the counter past 65535 resets the counter to zero.
16-31 —	Reserved

30.4.4 Clock and Transfer Attributes Register (In Master Mode) (CTAR0 - CTAR3)

30.4.4.1 Offset

For a = 0 to 3:

Register	Offset
CTARa	Ch + (a × 4h)

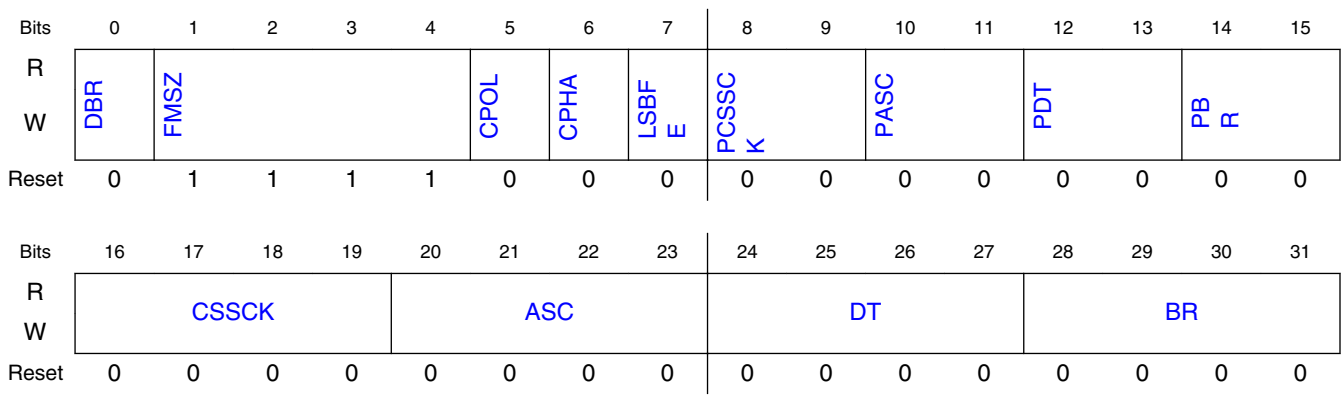
30.4.4.2 Function

CTAR registers are used to define different transfer attributes. Do not write to the CTAR registers while the module is in the Running state.

In Master mode, the CTAR registers define combinations of transfer attributes such as frame size, clock phase and polarity, data bit ordering, baud rate, and various delays.

When the module is configured as a SPI master, the CTAS field in the command portion of the TX FIFO entry selects which of the CTAR registers is used.

30.4.4.3 Diagram



30.4.4.4 Fields

Field	Function
0 DBR	Double Baud Rate Doubles the effective baud rate of the Serial Communications Clock (SCK). This field is used only in master mode. It effectively halves the Baud Rate division ratio, supporting faster frequencies, and odd division ratios for the Serial Communications Clock (SCK). When the DBR bit is set, the duty cycle of the Serial Communications Clock (SCK) depends on the value in the Baud Rate Prescaler and the Clock Phase bit as listed in the following table. See the BR field description for details on how to compute the baud rate.

Table 30-6. SPI SCK Duty Cycle

DBR	CPHA	PBR	SCK Duty Cycle
0	any	any	50/50
1	0	00	50/50
1	0	01	33/66
1	0	10	40/60
1	0	11	43/57

Table continues on the next page...

Field	Function
	Table 30-6. SPI SCK Duty Cycle (continued) DBR / CPHA / PBR / SCK Duty Cycle 1 / 1 / 00 / 50/50 1 / 1 / 01 / 66/33 1 / 1 / 10 / 60/40 1 / 1 / 11 / 57/43 0b - The baud rate is computed normally with a 50/50 duty cycle. 1b - The baud rate is doubled with the duty cycle depending on the Baud Rate Prescaler.
1-4 FMSZ	Frame Size The number of bits transferred per frame is equal to the FMSZ value plus 1. Regardless of the transmission mode, the minimum valid frame size value is 4.
5 CPOL	Clock Polarity Selects the inactive state of the Serial Communications Clock (SCK). For successful communication between serial devices, the devices must have identical clock polarities. When the Continuous Selection Format is selected, switching between clock polarities without stopping the module can cause errors in the transfer due to the peripheral device interpreting the switch of clock polarity as a valid clock edge. **NOTE:** In case of Continuous SCK mode, when the module goes in low power mode(disabled), inactive state of SCK is not guaranteed. 0b - The inactive state value of SCK is low. 1b - The inactive state value of SCK is high.
6 CPHA	Clock Phase Selects which edge of SCK causes data to change and which edge causes data to be captured. This bit is used in both master and slave mode. For successful communication between serial devices, the devices must have identical clock phase settings. In Continuous SCK mode, the bit value is ignored and the transfers are done as if the CPHA bit is set to 1. 0b - Data is captured on the leading edge of SCK and changed on the following edge. 1b - Data is changed on the leading edge of SCK and captured on the following edge.
7 LSBFE	LSB First Specifies whether the LSB or MSB of the frame is transferred first. 0b - Data is transferred MSB first. 1b - Data is transferred LSB first.
8-9 PCSSCK	PCS to SCK Delay Prescaler Selects the prescaler value for the delay between assertion of PCS and the first edge of the SCK. See the CSSCK field description for information on how to compute the PCS to SCK Delay. Refer PCS to SCK Delay (t_{CSC}) for more details. 00b - PCS to SCK Prescaler value is 1. 01b - PCS to SCK Prescaler value is 3. 10b - PCS to SCK Prescaler value is 5. 11b - PCS to SCK Prescaler value is 7.
10-11 PASC	After SCK Delay Prescaler Selects the prescaler value for the delay between the last edge of SCK and the negation of PCS. See the ASC field description for information on how to compute the After SCK Delay. Refer After SCK Delay (t_{ASC}) for more details. 00b - Delay after Transfer Prescaler value is 1.

Table continues on the next page...

Field	Function
	01b - Delay after Transfer Prescaler value is 3. 10b - Delay after Transfer Prescaler value is 5. 11b - Delay after Transfer Prescaler value is 7.
12-13 PDT	Delay after Transfer Prescaler Selects the prescaler value for the delay between the negation of the PCS signal at the end of a frame and the assertion of PCS at the beginning of the next frame. The PDT field is only used in master mode. See the DT field description for details on how to compute the Delay after Transfer. Refer Delay after Transfer (t_{DT}) for more details. 00b - Delay after Transfer Prescaler value is 1. 01b - Delay after Transfer Prescaler value is 3. 10b - Delay after Transfer Prescaler value is 5. 11b - Delay after Transfer Prescaler value is 7.
14-15 PBR	Baud Rate Prescaler Selects the prescaler value for the baud rate. This field is used only in master mode. The baud rate is the frequency of the SCK. The protocol clock is divided by the prescaler value before the baud rate selection takes place. See the BR field description for details on how to compute the baud rate. 00b - Baud Rate Prescaler value is 2. 01b - Baud Rate Prescaler value is 3. 10b - Baud Rate Prescaler value is 5. 11b - Baud Rate Prescaler value is 7.
16-19 CSSCK	PCS to SCK Delay Scaler Selects the scaler value for the PCS to SCK delay. This field is used only in master mode. The PCS to SCK Delay is the delay between the assertion of PCS and the first edge of the SCK. The delay is a multiple of the protocol clock period, and it is computed according to the following equation: $t_{CSC} = (1/f_P) \times PCSSCK \times CSSCK$. The following table lists the delay scaler values.

Table 30-7. Delay Scaler Encoding

Field Value	Delay Scaler Value
0000	2
0001	4
0010	8
0011	16
0100	32
0101	64
0110	128
0111	256
1000	512
1001	1024
1010	2048
1011	4096
1100	8192
1101	16384
1110	32768
1111	65536

Table continues on the next page...

Field	Function
	Refer PCS to SCK Delay (t_{CSC}) for more details.
20-23 ASC	After SCK Delay Scaler Selects the scaler value for the After SCK Delay. This field is used only in master mode. The After SCK Delay is the delay between the last edge of SCK and the negation of PCS. The delay is a multiple of the protocol clock period, and it is computed according to the following equation: $t_{ASC} = (1/f_P) \times PASC \times ASC$ See Delay Scaler Encoding table in CTARn[CSSCK] bit field description for scaler values. Refer After SCK Delay (t_{ASC}) for more details.
24-27 DT	Delay After Transfer Scaler Selects the Delay after Transfer Scaler. This field is used only in master mode. The Delay after Transfer is the time between the negation of the PCS signal at the end of a frame and the assertion of PCS at the beginning of the next frame. In the Continuous Serial Communications Clock operation, the DT value is fixed to one SCK clock period, The Delay after Transfer is a multiple of the protocol clock period, and it is computed according to the following equation: $t_{DT} = (1/f_P) \times PDT \times DT$ See Delay Scaler Encoding table in CTARn[CSSCK] bit field description for scaler values.
28-31 BR	Baud Rate Scaler Selects the scaler value for the baud rate. This field is used only in master mode. The prescaled protocol clock is divided by the Baud Rate Scaler to generate the frequency of the SCK. The baud rate is computed according to the following equation: SCK baud rate = $(f_P / PBR) \times [(1+DBR)/BR]$ The following table lists the baud rate scaler values.

Table 30-8. Baud Rate Scaler

CTARn[BR]	Baud Rate Scaler Value
0000	2
0001	4
0010	6
0011	8
0100	16
0101	32
0110	64
0111	128
1000	256
1001	512
1010	1024
1011	2048
1100	4096
1101	8192
1110	16384
1111	32768

30.4.5 Status Register (SR)

30.4.5.1 Offset

Register	Offset
SR	2Ch

30.4.5.2 Function

SR contains status and flag bits. The bits reflect the status of the module and indicate the occurrence of events that can generate interrupt or DMA requests. Software can clear flag bits in the SR by writing a 1 to them. Writing a 0 to a flag bit has no effect. This register may not be writable in Module Disable mode due to the use of power saving mechanisms.

30.4.5.3 Diagram

Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TCF	TXRXS	0	EOQF	0	0	TFFF	BSYF	CMDTCF	0	SPEF	0	RFOF	TFIWF	RFDF	CMDFFF
W	W1C			W1C			W1C		W1C		W1C		W1C	W1C		W1C
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1

Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	TXCTR				TXNXTPTR				RXCTR				POPNXTPTR			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

30.4.5.4 Fields

Field	Function
0	Transfer Complete Flag

Table continues on the next page...

Field	Function
TCF	Indicates that all bits in a frame have been shifted out. TCF remains set until it is cleared by writing a 1 to it. 0b - Transfer not complete. 1b - Transfer complete.
1 TXRXS	TX and RX Status Reflects the run status of the module. 0b - Transmit and receive operations are disabled (The module is in Stopped state). 1b - Transmit and receive operations are enabled (The module is in Running state).
2 —	Reserved
3 EOQF	End of Queue Flag Indicates that the last entry in a queue has been transmitted when the module is in Master mode. The EOQF bit is set when the TX FIFO entry has the EOQ bit set in the command halfword and the end of the transfer is reached. The EOQF bit remains set until cleared by writing a 1 to it. When the EOQF bit is set, the TXRXS bit is automatically cleared. 0b - EOQ is not set in the executing command. 1b - EOQ is set in the executing SPI command.
4 —	Reserved
5 —	Reserved
6 TFFF	Transmit FIFO Fill Flag Indicates whether there is an available location to be filled in the FIFO. Either DMA or an interrupt can be used to add another entry to the FIFO. Note that this bit is set if at least one location is free in the FIFO. The TFFF bit can be cleared by writing 1 to it or by acknowledgement from the DMA controller to the TX FIFO full request, when the TX FIFO is full. **NOTE:** The reset value of this bit is 0 when the module is disabled,(MCR[MDIS]=1). 0b - TX FIFO is full. 1b - TX FIFO is not full.
7 BSYF	Busy Flag. This bit is valid only when DSPI_MCR[XSPI] is enabled. Indicates that the current Command Frame is being used for transmiting multiple data frames. This bit is not set for the last Data Frame of a Cyclic command Transfer or when DSPI_CTARE[DTCP] = 1. Refer Command First In First Out (CMD FIFO) Buffering Mechanism for more details. 0b - No Cyclic Command Transfer in Progress. 1b - Cyclic Command Transfer is in progress. Current Data Frame is not the last data frame for on-going cyclic command transfer..
8 CMDTCF	Command Transfer Complete Flag. Indicates that the last Data frame for the current Cyclic Command has been transmitted. Hence this bit is set only for the last Data Frame of a Cyclic Command Transfer or when DSPI_CTARE[DTCP] = 1. The bit remains set until it is cleared by writing a '1' to it. 0b - Data Transfer by current Command not complete. 1b - Data Transfer by current Command is complete.
9 —	Reserved
10 SPEF	SPI Parity Error Flag Indicates that a SPI frame with parity error had been received. The bit remains set until it is cleared by writing a 1 to it.

Table continues on the next page...

Field	Function
	0b - No parity error. 1b - Parity error has occurred.
11 —	Reserved
12 RFOF	Receive FIFO Overflow Flag Indicates an overflow condition in the RX FIFO. The field is set when the RX FIFO and shift register are full and a transfer is initiated. The bit remains set until it is cleared by writing a 1 to it. 0b - No Rx FIFO overflow. 1b - Rx FIFO overflow has occurred.
13 TFIWF	Tranmit FIFO Invalid Write Flag Indicates Data Write on TX FIFO while CMD FIFO is empty. Without a Command, the Data entries present in TXFIFO are invalid. This bit remains set until it is cleared by writing a '1' to it. 0b - No Invalid Data present in TX FIFO. 1b - Invalid Data present in TX FIFO since CMD FIFO is empty.
14 RFDF	Receive FIFO Drain Flag Provides a method for the module to request that entries be removed from the RX FIFO. Note that this bit is set if at least one location can be read from the FIFO. The RFDF bit can be cleared by acknowledgement from the DMA controller when the RX FIFO is empty. 0b - RX FIFO is empty. 1b - This bit auto-clears on every RXFR read performed.
15 CMDFFF	Command FIFO Fill Flag Indicates whether there is an available location to be filled in the FIFO. Either a DMA request or an interrupt indication can be used to add another entry to the FIFO. Note that this bit is set if at least one location is free in the FIFO. The CMDFFF is cleared by writing a '1' to it or by acknowledgement from the DMA controller to the CMD FIFO full request. 0b - CMD FIFO is full. 1b - CMD FIFO is not full.
16-19 TXCTR	TX FIFO Counter Indicates the number of valid entries in the TX FIFO. The TXCTR is incremented every time the PUSHR is written. The TXCTR is decremented every time an SPI command is executed and the SPI data is transferred to the shift register.
20-23 TXNXTPTR	Transmit Next Pointer Indicates which TX FIFO entry is transmitted during the next transfer. The TXNXTPTR field is updated every time SPI data is transferred from the TX FIFO to the shift register. **NOTE:** When TX FIFO is cleared by setting DSPI_MCR[CLR_TXF] to 1, this field does not update immediately to 0. Only when the next transfer starts, this field reflects the latest value TXNXTPTR =1.
24-27 RXCTR	RX FIFO Counter Indicates the number of entries in the RX FIFO. The RXCTR is decremented every time the POPR is read. The RXCTR is incremented every time data is transferred from the shift register to the RX FIFO.
28-31 POPNXTPTR	Pop Next Pointer Contains a pointer to the RX FIFO entry to be returned when the POPR is read. The POPNXTPTR is updated when the POPR is read.

30.4.6 DMA/Interrupt Request Select and Enable Register (RSER)

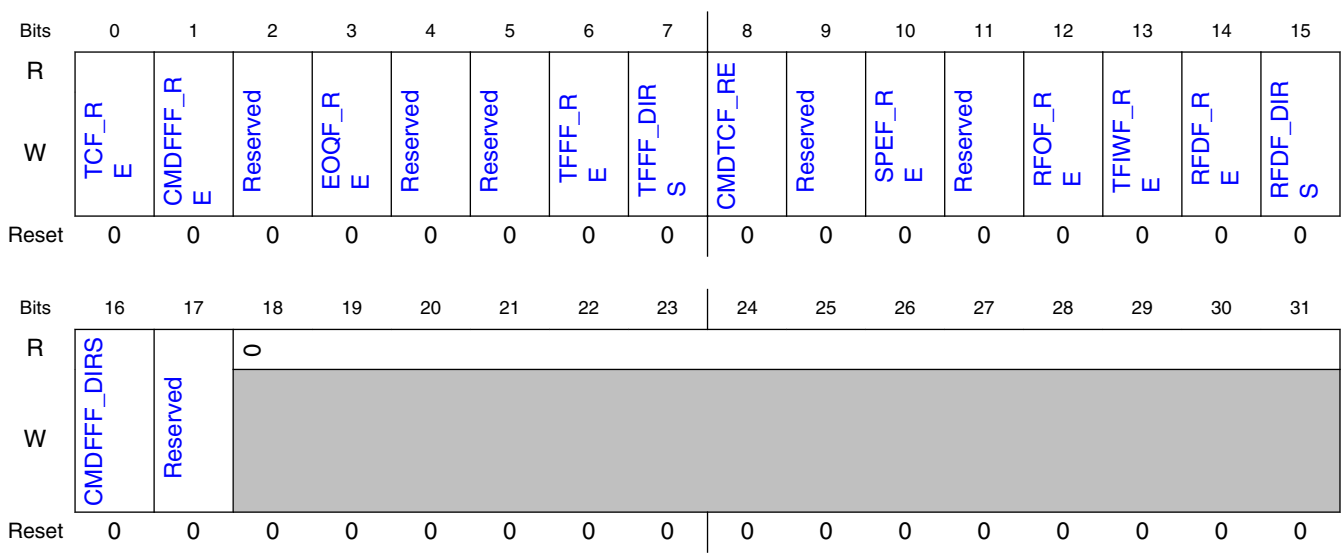
30.4.6.1 Offset

Register	Offset
RSER	30h

30.4.6.2 Function

RSER controls DMA and interrupt requests. Do not write to the RSER while the module is in the Running state.

30.4.6.3 Diagram



30.4.6.4 Fields

Field	Function
0 TCF_RE	Transmission Complete Request Enable Enables TCF flag in the SR to generate an interrupt request. 0b - TCF interrupt requests are disabled. 1b - TCF interrupt requests are enabled.

Table continues on the next page...

Field	Function
1 CMDFFF_RE	Command FIFO Fill Flag Request Enable. Enables the CMDFFF flag in the SR to generate a request. The CMDFFF_DIRS bit selects between generating an interrupt request or a DMA request. 0b - CMDFFF interrupts or DMA requests are disabled. 1b - CMDFFF interrupts or DMA requests are enabled.
2 —	Always write the reset value to this field.
3 EOQF_RE	Finished Request Enable Enables the EOQF flag in the SR to generate an interrupt request. 0b - EOQF interrupt requests are disabled. 1b - EOQF interrupt requests are enabled.
4 —	Reserved
5 —	Always write the reset value to this field.
6 TFFF_RE	Transmit FIFO Fill Request Enable Enables the TFFF flag in the SR to generate a request. The TFFF_DIRS bit selects between generating an interrupt request or a DMA request. 0b - TFFF interrupts or DMA requests are disabled. 1b - TFFF interrupts or DMA requests are enabled.
7 TFFF_DIRS	Transmit FIFO Fill DMA or Interrupt Request Select Selects between generating a DMA request or an interrupt request. When SR[TFFF] and RSER[TFFF_RE] are set, this field selects between generating an interrupt request or a DMA request. 0b - TFFF flag generates interrupt requests. 1b - TFFF flag generates DMA requests.
8 CMDTCF_RE	Command Transmission Complete Request Enable. The CMDTCF_RE bit enables CMDTCF flag in the SR to generate an interrupt request. 0b - CMDTCF interrupt requests are disabled. 1b - CMDTCF interrupt requests are enabled.
9 —	Always write the reset value to this field.
10 SPEF_RE	SPI Parity Error Request Enable Enables the SPEF flag in the SR to generate an interrupt request. 0b - SPEF interrupt requests are disabled. 1b - SPEF interrupt requests are enabled.
11 —	Always write the reset value to this field.
12 RFOF_RE	Receive FIFO Overflow Request Enable Enables the RFOF flag in the SR to generate an interrupt request. 0b - RFOF interrupt requests are disabled. 1b - RFOF interrupt requests are enabled.
13 TFIWF_RE	Transmit FIFO Invalid Write Request Enable. Enables the TFIWF flag in the SR to generate an interrupt request. 0b - TFIWF interrupt requests are disabled. 1b - TFIWF interrupt requests are enabled.

Table continues on the next page...

Field	Function
14 RFDF_RE	Receive FIFO Drain Request Enable Enables the RFDF flag in the SR to generate a request. The RFDF_DIRS bit selects between generating an interrupt request or a DMA request. 0b - RFDF interrupt or DMA requests are disabled. 1b - RFDF interrupt or DMA requests are enabled.
15 RFDF_DIRS	Receive FIFO Drain DMA or Interrupt Request Select Selects between generating a DMA request or an interrupt request. When the RFDF flag bit in the SR is set, and the RFDF_RE bit in the RSER is set, the RFDF_DIRS bit selects between generating an interrupt request or a DMA request. 0b - Interrupt request. 1b - DMA request.
16 CMDFFF_DIRS	Command FIFO Fill DMA or Interrupt Request Select Selects between generating a DMA request or an interrupt request. When the CMDFFF flag bit in the SR is set, and the CMDFFF_RE bit in the RSER is set, the CMDFFF_DIRS bit selects between generating an interrupt request or a DMA request. 0b - CMDFFF flag generates interrupt requests. 1b - CMDFFF flag generates DMA requests.
17 —	Always write the reset value to this field.
18-31 —	Reserved

30.4.7 PUSH TX FIFO Register In Master Mode (PUSHR)

30.4.7.1 Offset

Register	Offset
PUSHR	34h

30.4.7.2 Function

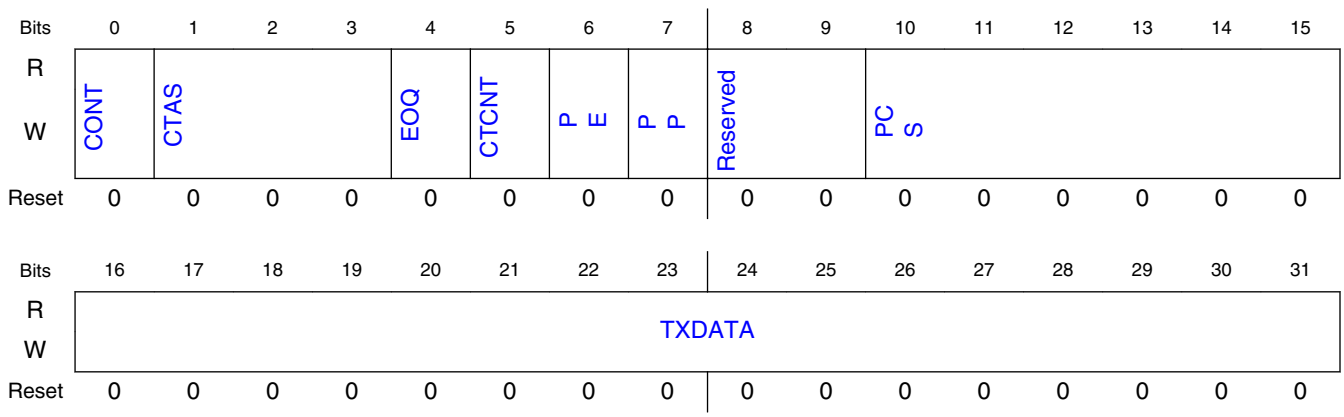
Specifies data to be transferred to the TX FIFO and CMD FIFO. User must write 16-bits data into TXDATA field. An 8- or 16-bit write access to the TXDATA field transfers 16 bits of data bus to the TX FIFO. A write access to the command fields transfers the 16 bits of command information to the CMD FIFO. In Master mode, the register transfers 16 bits of data to the TX FIFO and 16 bits of command information to the CMD FIFO.

If Extended SPI Mode is disabled (MCR[XSPI]), the TX FIFO and CMD FIFO must be filled simultaneously. In other words, you must perform write accesses to both the data and command fields for every PUSHR operation. With Extended SPI Mode disabled and both the TX FIFO and CMD FIFO are written to and read from simultaneously, they behave as a single 32 bit FIFO. When Extended SPI mode is enabled (MCR[XSPI]), the TX FIFO and CMD FIFO can be written to independently.

A read access of PUSHR returns the topmost TX FIFO and CMD FIFO entries concatenated.

When the module is disabled, writing to this register does not update the FIFO. Therefore, any reads performed while the module is disabled return the last PUSHR write performed while the module was still enabled.

30.4.7.3 Diagram



30.4.7.4 Fields

Field	Function
0 CONT	Continuous Peripheral Chip Select Enable Selects a continuous selection format. The bit is used in SPI Master mode. The bit enables the selected PCS signals to remain asserted between transfers. 0b - Return PCSn signals to their inactive state between transfers. 1b - Keep PCSn signals asserted between transfers.
1-3 CTAS	Clock and Transfer Attributes Select Selects which CTAR to use in master mode to specify the transfer attributes for the associated SPI frame. See the chip specific section for details to determine how many CTARs this device has. You should not program a value in this field for a register that is not present. 000b - CTAR0

Table continues on the next page...

Field	Function
	001b - CTAR1 010b - CTAR2 011b - CTAR3 100b - Reserved 101b - Reserved 110b - Reserved 111b - Reserved
4 EOQ	End Of Queue Host software uses this bit to signal to the module that the current SPI transfer is the last in a queue. At the end of the transfer, the EOQF bit in the SR is set. 0b - The SPI data is not the last data to transfer. 1b - The SPI data is the last data to transfer.
5 CTCNT	Clear Transfer Counter Clears the TCNT field in the TCR register. The TCNT field is cleared before the module starts transmitting the current SPI frame. 0b - Do not clear the TCR[TCNT] field. 1b - Clear the TCR[TCNT] field.
6 PE	Parity Enable Enables parity bit transmission and parity reception check for the SPI frame. 0b - No parity bit included/checked. 1b - Parity bit is transmitted instead of the last data bit in the frame; parity is checked for the received frame.
7 PP	Parity Polarity Controls the polarity of the parity bit transmitted and checked. 0b - Even Parity: the number of 1 bits in the transmitted frame is even. The SR[SPEF] bit is set if the number of 1 bits is odd in the received frame. 1b - Odd Parity: the number of 1 bits in the transmitted frame is odd. The SR[SPEF] bit is set if the number of 1 bits is even in the received frame.
8-9 —	Always write the reset value to this field.
10-15 PCS	PCS Select which PCS signals are to be asserted for the transfer. 000000b - Negate the PCS[x] signal. 000001b - Assert the PCS[x] signal.
16-31 TXDATA	Transmit Data Holds SPI data to be transferred according to the associated SPI command.

30.4.8 POP RX FIFO Register (POPR)

30.4.8.1 Offset

Register	Offset
POPR	38h

30.4.8.2 Function

POPR is used to read the RX FIFO. Eight- or sixteen-bit read accesses to the POPR have the same effect on the RX FIFO as 32-bit read accesses. A write to this register will generate a Transfer Error.

30.4.8.3 Diagram

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

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

30.4.8.4 Fields

Field	Function
0-31 RXDATA	Received Data Contains the SPI data from the RX FIFO entry to which the Pop Next Data Pointer points.

30.4.9 Transmit FIFO Registers (TXFR0 - TXFR15)

30.4.9.1 Offset

For a = 0 to 15:

Register	Offset
TXFRa	3Ch + (a × 4h)

30.4.9.2 Function

TXFRn registers provide visibility into the TX FIFO for debugging purposes. Each register is an entry in the TX FIFO. The registers are read-only and cannot be modified. Reading the TXFRx registers does not alter the state of the TX FIFO. When the module is operating in Extended SPI mode, reading TXFRn registers is invalid.

30.4.9.3 Diagram

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

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

30.4.9.4 Fields

Field	Function
0-15 TXCMD_TXDATA	Transmit Command or Transmit Data In Master mode the TXCMD field contains the command that sets the transfer attributes for the SPI data.
16-31 TXDATA	Transmit Data Contains the SPI data to be shifted out.

30.4.10 Receive FIFO Registers (RXFR0 - RXFR15)

30.4.10.1 Offset

For a = 0 to 15:

Register	Offset
RXFRa	7Ch + (a × 4h)

30.4.10.2 Function

RXFRn provide visibility into the RX FIFO for debugging purposes. Each register is an entry in the RX FIFO. The RXFR registers are read-only. Reading the RXFRx registers does not alter the state of the RX FIFO. The field MCR[MDIS] must be 0 when RXFR is read.

30.4.10.3 Diagram

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

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

30.4.10.4 Fields

Field	Function
0-31 RXDATA	Receive Data Contains the received SPI data.

30.4.11 Clock and Transfer Attributes Register Extended (CTARE0 - CTARE3)

30.4.11.1 Offset

For a = 0 to 3:

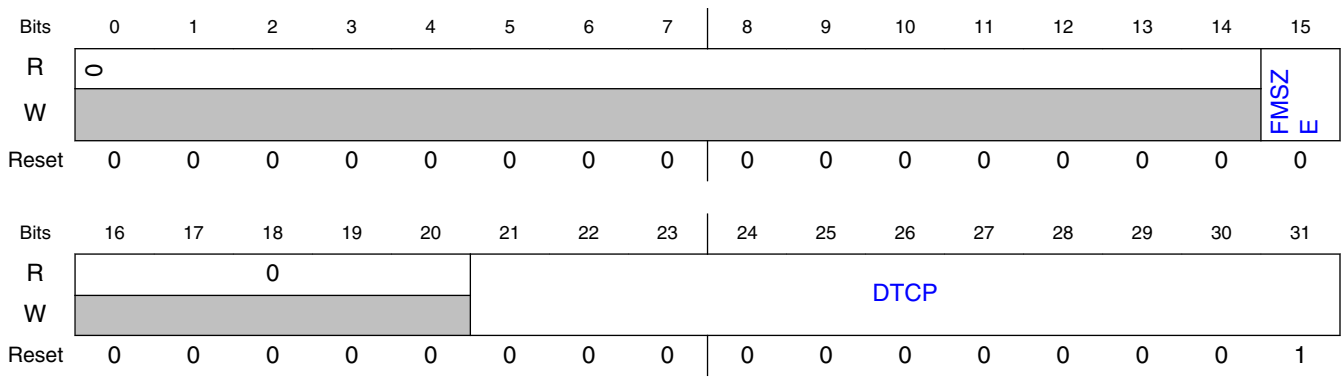
Register	Offset
CTAREa	11Ch + (a × 4h)

30.4.11.2 Function

CTARE registers are used to define the extended transfer attributes for an SPI frame. These registers are valid only when Extended SPI mode is enabled (MCR[XSPI]).

When the module is configured as a SPI master, the CTAS field in CMD FIFO entry selects which of the CTARE registers is used.

30.4.11.3 Diagram



30.4.11.4 Fields

Field	Function
0-14 —	Reserved
15 FMSZE	Frame Size Extended This field is valid only when MCR[XSPI] is set. This field concatenated with CTAR[FMSZ] defines the Frame size of the SPI frames to be transmitted. Effective frame size would be the concatenation of {CTARE[FMSZE], CTAR[FMSZ]} plus 1. 0b - Default Mode. Up to 16 bit SPI frames can be transfered. 1b - Up to 32 bit SPI frames can be transfered. Each Frame transfer will be a result of 2 simulataneous TX FIFO Pop Operation.
16-20 —	Reserved
21-31	Data Transfer Count Preload

Field	Function
DTCP	This field is valid only when SPIx_MCR[XSPI] is set. This field defines the number of data frames (whose size is defined by CTARE[FMSZE] and CTAR[FMSZ]) to be transmitted using the Command frame which selected this SPIx_CTARE register. The value 0 is reserved and should not be written in this field. The default value of this field is 1.

30.4.12 Status Register Extended (SREX)

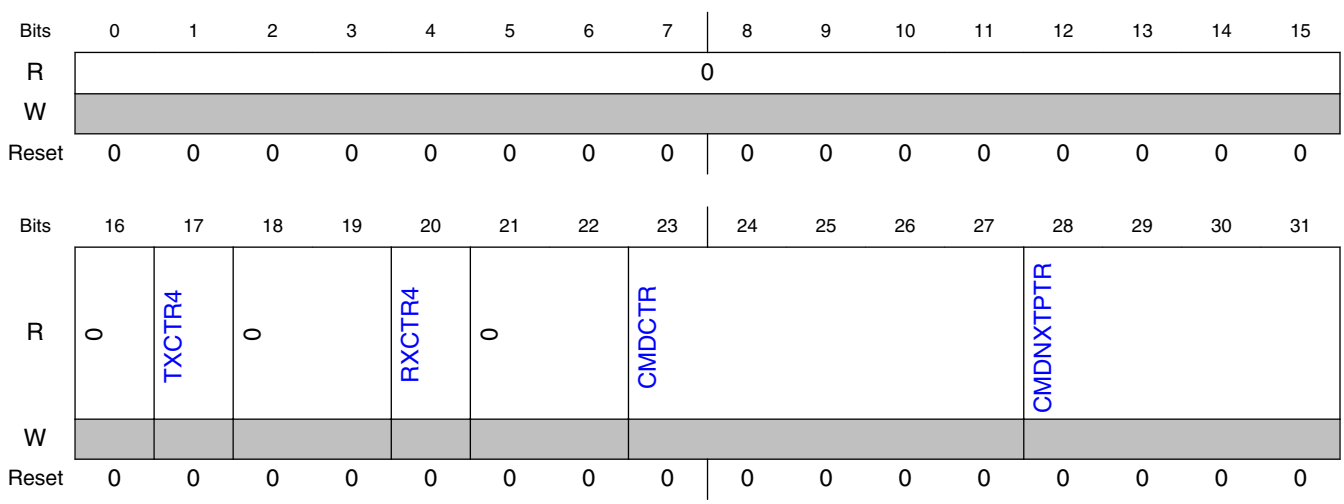
30.4.12.1 Offset

Register	Offset
SREX	13Ch

30.4.12.2 Function

The register contains status fields. The fields reflect the status of the module and indicate the occurrence of events. This register is not writable.

30.4.12.3 Diagram



30.4.12.4 Fields

Field	Function
0-16 —	Reserved
17 TXCTR4	TX FIFO Counter[4] This bit is an extension of SR[TXCTR]. The concatenated field {TXCTR4, TXCTR} indicates the number of valid entries in the TX FIFO. This field is incremented every time the PUSHR is written. And this field is decremented every time an SPI command is executed and the SPI data is transferred to the shift register.
18-19 —	Reserved
20 RXCTR4	RX FIFO Counter[4] This bit is an extension of SR[RXCTR]. The concatenated field {RXCTR4, RXCTR} indicates the number of entries in the RX FIFO. This field is decremented every time the POPR is read. And this field is incremented every time data is transferred from the shift register to the RX FIFO.
21-22 —	Reserved
23-27 CMDCTR	CMD FIFO Counter Indicates the number of entries in the CMD FIFO. The CMDCTR is incremented every time the command part of PUSHR is written. The CMDCTR is decremented every time a SPI command is executed (all data frames due to current command frame have been transmitted).
28-31 CMDNXTPTR	Command Next Pointer Indicates which CMD FIFO Entry is used during the next transfer. The CMDNXTPTR field is updated every time SPI data due to current command have been transmitted.

30.5 Functional description

The module supports full-duplex, synchronous serial communications between chips and peripheral devices.The SPI configuration transfers data serially using a shift register and a selection of programmable transfer attributes.

The module has the following configuration

- The SPI Configuration in which the module operates as a basic SPI or a queued SPI.

The DCONF field in the Module Configuration Register (MCR) determines the module Configuration. SPI configuration is selected when DCONF within SPIx_MCR is 0b00.

The CTARn registers hold clock and transfer attributes. The SPI configuration allows to select which CTAR to use on a frame by frame basis by setting a field in the SPI command. The Extended SPI Mode (SPIx_MCR[XSPI]) further allows the usage CTAREn (CTARn Extended) registers which allows the user to send multiple data frames using a single command frame.

For more information, see the fields of CTAR registers. See CTAREx registers for information on the fields of CTARE registers.

Typical master to slave connections are shown in the following figure. When a data transfer operation is performed, data is serially shifted a predetermined number of bit positions. Because the modules are linked, data is exchanged between the master and the slave. The data that was in the master shift register is now in the shift register of the slave, and vice versa. At the end of a transfer, the Transfer Control Flag(TCF) bit in the Shift Register(SR) is set to indicate a completed frame transfer.

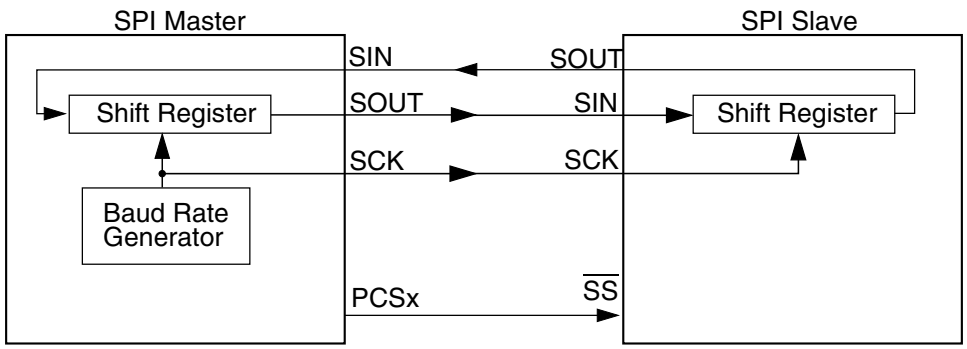


Figure 30-3. Serial protocol overview

Generally, more than one slave device can be connected to the module master. 6 Peripheral Chip Select (PCS) signals of the module masters can be used to select which of the slaves to communicate with.

The SPI configuration shares transfer protocol and timing properties which are described independently of the configuration in Transfer formats. The transfer rate and delay settings are described in Module baud rate and clock delay generation.

30.5.1 Start and Stop of module transfers

The module has two operating states: Stopped and Running. Both the states are independent of it's configuration. The default state of the module is Stopped. In the Stopped state, no serial transfers are initiated in Master mode. The Stopped state is also a safe state for writing the various configuration registers of the module without causing undetermined results. In the Running state serial transfers take place.

The TXRXS bit in the SR indicates the state of module. The bit is set if the module is in Running state.

The module starts or transitions to Running when all of the following conditions are true:

- SR[EOQF] bit is clear
- Chip is not in the Debug mode or the MCR[FRZ] bit is clear
- MCR[HALT] bit is clear

The module stops or transitions from Running to Stopped after the current frame when any one of the following conditions exist:

- SR[EOQF] bit is set
- Chip in the Debug mode and the MCR[FRZ] bit is set
- MCR[HALT] bit is set

State transitions from Running to Stopped occur on the next frame boundary if a transfer is in progress, or immediately if no transfers are in progress.

30.5.2 Serial Peripheral Interface (SPI) configuration

The SPI configuration transfers data serially using a shift register and a selection of programmable transfer attributes. The module is in SPI configuration when the DCONF field in the MCR is 0b00. The SPI frames can be 32 bits long. The host CPU or a DMA controller transfers the SPI data from the external to the module RAM queues to a TX FIFO buffer. The received data is stored in entries in the RX FIFO buffer. The host CPU or the DMA controller transfers the received data from the RX FIFO to memory external to the module. The operation of FIFO buffers is described in the following sections:

- Transmit First In First Out (TX FIFO) buffering mechanism
- Command First In First Out (CMD FIFO) Buffering Mechanism
- Receive First In First Out (RX FIFO) buffering mechanism

The interrupt and DMA request conditions are described in Interrupts/DMA requests.

In Master mode the module initiates and controls the transfer according to the fields of the executing SPI Command.

30.5.2.1 Master mode

In SPI Master mode, the module initiates the serial transfers by controlling the SCK and the PCS signals. The executing SPI Command determines which CTARs will be used to set the transfer attributes and which PCS signals to assert. The command field also contains various bits that help with queue management and transfer protocol. See PUSH TX FIFO Register In Master Mode (PUSHR) for details on the SPI command fields. The data in the executing TX FIFO entry is loaded into the shift register and shifted out on the Serial Out (SOUT) pin. In SPI Master mode, each SPI frame to be transmitted has a command associated with it, allowing for transfer attribute control on a frame by frame basis.In Extended SPI Master Mode, multiple SPI frames can have a single command associated with them allowing for efficient SPI frame transfers requiring common transfer attributes. In addition, the Extended SPI Mode allows for larger frame sizes of up to 32 bits.

30.5.2.2 FIFO disable operation

The FIFO disable mechanisms allow SPI transfers without using the TX FIFO, CMD FIFO or RX FIFO. The module operates as a double-buffered simplified SPI when the FIFOs are disabled. The Transmit and Receive side of the FIFOs are disabled separately. Setting the MCR[DIS_TXF] bit disables the TX FIFO and CMD FIFO, and setting the MCR[DIS_RXF] bit disables the RX FIFO.

The FIFO disable mechanisms are transparent to the user and to host software. Transmit data and commands are written to the PUSHR and received data is read from the POPR.

When the TX FIFO and CMD FIFO are disabled:

- SR[TFFF], SR[TFUF], SR[CMDFFF], SREX[CMDCTR] and SR[TXCTR] behave as if there is a one-entry FIFO
- The contents of TXFRs, SR[TXNXTPTR] and SREX[CMDNXTPTR] are undefined

Similarly, when the RX FIFO is disabled, the RFDF, RFOF, and RXCTR fields in the SR behave as if there is a one-entry FIFO, but the contents of the RXFR registers and POPNXTPTR are undefined.

30.5.2.3 Transmit First In First Out (TX FIFO) buffering mechanism

The TX FIFO functions as a buffer of SPI data for transmission. The TX FIFO holds 16 words, each consisting of SPI data. The number of entries in the TX FIFO is device-specific. SPI data is added to the TX FIFO by writing to the Data Field of module PUSH FIFO Register (PUSHR). TX FIFO entries can only be removed from the TX FIFO by being shifted out or by flushing the TX FIFO.

The TX FIFO Counter field (TXCTR) in the module Status Register (SR) indicates the number of valid entries in the TX FIFO. The TXCTR is updated every time a 8- or 16-bit write takes place to PUSHR[TXDATA] or SPI data is transferred into the shift register from the TX FIFO.

The TXNXTPTR field indicates the TX FIFO Entry that will be transmitted during the next transfer. The TXFRn Registers are invalid in the Extended SPI Mode, since the TX FIFO and CMD FIFO can be used independently. The TXNXTPTR field is incremented every time SPI data is transferred from the TX FIFO to the shift register. The maximum value of the field is equal to the maximum implemented TXFR number and it rolls over after reaching the maximum.

30.5.2.3.1 Filling the TX FIFO

Host software or other intelligent blocks can add (push) entries to the TX FIFO and CMD FIFO by writing to the PUSHR. When the TX FIFO is not full, the TX FIFO Fill Flag (TFFF) in the SR is set. The TFFF bit is cleared when TX FIFO is full and the DMA controller indicates that a write to PUSHR is complete. Writing a '1' to the TFFF bit also clears it. The TFFF can generate a DMA request or an interrupt request. See Transmit FIFO Fill Interrupt or DMA Request for details.

The module ignores attempts to push data to a full TX FIFO, and the state of the TX FIFO does not change and no error condition is indicated.

30.5.2.3.2 Draining the TX FIFO

The TX FIFO entries are removed (drained) by shifting SPI data out through the shift register. Entries are transferred from the TX FIFO to the shift register and shifted out as long as there are valid entries in the TX FIFO. Every time an entry is transferred from the TX FIFO to the shift register, the TX FIFO Counter decrements by one. When Extended SPI Mode (SPIx_MCR[XSPI]) is enabled, if the frame size of SPI Data to be transmitted is more than 16 bits, then it causes two Data entries to be popped from TX FIFO simultaneously which are transferred to the shift register. The first of the two popped entries forms the 16 least significant bits of the SPI frame to be transmitted. Such an operation also causes TX FIFO Counter to decrement by two. At the end of a transfer, the TCF bit in the SR is set to indicate the completion of a transfer. The TX FIFO is flushed by writing a '1' to the CLR_TXF bit in MCR.

If an external bus master initiates a transfer with a module slave while the slave's TX FIFO is empty, the Transmit FIFO Underflow Flag (TFUF) in the slave's SR is set. See Transmit FIFO Underflow Interrupt Request for details.

30.5.2.4 Command First In First Out (CMD FIFO) Buffering Mechanism

The CMD FIFO functions as a buffer of SPI command used for SPI data transmission. When Extended SPI Mode (MCR[XSPI]) is disabled, the TX FIFO and CMD FIFO must be filled together, i.e. write enables should be given for both the Data and Command fields while performing a PUSHR operation.When Extended SPI Mode (MCR[XSPI]) is enabled, the TX FIFO and CMD FIFO can be filled independently.

The CMD FIFO holds 16 words, each representing SPI command fields. The number of entries in the CMD FIFO is device-specific. SPI Command is added to the CMD FIFO by writing to the command field of SPI PUSH FIFO Register (PUSHR). CMD FIFO entries can only be removed from the CMD FIFO by being shifted out (to help transmit SPI data) or by flushing the CMD FIFO.

When Extended SPI Mode (MCR[XSPI]) is disabled, every CMD FIFO entry has a corresponding single TX FIFO entry attached to it because both these FIFO's are filled simultaneously.

When Extended SPI Mode (MCR[XSPI]) is enabled, every CMD FIFO entry can have multiple TX FIFO entries attached to it. Thus a single CMD FIFO entry can be used to transmit multiple TX FIFO entries. The CTARE[DTCP] field decides the number of SPI Data Frames having frame size as {FMSZE, FMSZ} to be transmitted using the current Command Entry. The CTAR/CTARE registers pointed by the CTAS field in the Command frame gives the FMSZ and FMSZE fields respectively. The time for which a command entry is in use is known as a Command Cycle. The Busy Flag SR[BSYF] is asserted for the duration of the Command Cycle except for the last SPI frame in the Command Cycle.

The CMD FIFO Counter field (CMDCTR) in the SPI Status Register (SR) indicates the number of valid entries in the CMD FIFO. The CMDCTR field is updated every time a 8- or 16-bit write takes place on the lower half of SPI_PUSHR or SPI data is transferred into the shift register from the TX FIFO.

The TXFRn Registers are invalid in the Extended SPI Mode, since the TX FIFO and CMD FIFO can be used independently. The CMDNXTPTR field indicates which CMD FIFO Entry will be used during the next command cycle. The CMDNXTPTR field is incremented every time the last SPI data in the command cycle is transferred from the TX FIFO to the shift register and it rolls over after reaching the maximum.

30.5.2.5 Receive First In First Out (RX FIFO) buffering mechanism

The RX FIFO functions as a buffer for data received on the SIN pin. The RX FIFO holds 16 received SPI data frames. The number of entries in the RX FIFO is device-specific. SPI data is added to the RX FIFO at the completion of a transfer when the received data in the shift register is transferred into the RX FIFO. SPI data are removed (popped) from the RX FIFO by reading the module POP RX FIFO Register (POPR). RX FIFO entries can only be removed from the RX FIFO by reading the POPR or by flushing the RX FIFO.

The RX FIFO Counter field (RXCTR) in the module's Status Register (SR) indicates the number of valid entries in the RX FIFO. The RXCTR is updated every time the POPR is read or SPI data is copied from the shift register to the RX FIFO.

The POPNXTPTR field in the SR points to the RX FIFO entry that is returned when the POPR is read. The POPNXTPTR contains the positive offset from RXFR0 in a number of 32-bit registers. For example, POPNXTPTR equal to two means that the RXFR2 contains the received SPI data that will be returned when the POPR is read. The POPNXTPTR field is incremented every time the POPR is read. The maximum value of the field is equal to the maximum implemented RXFR number and it rolls over after reaching the maximum.

30.5.2.5.1 Filling the RX FIFO

The RX FIFO is filled with the received SPI data from the shift register. While the RX FIFO is not full, SPI frames from the shift register are transferred to the RX FIFO. Every time an SPI frame is transferred to the RX FIFO, the RX FIFO Counter is incremented by one.

If the RX FIFO and shift register are full and a transfer is initiated, the RFOF bit in the SR is set indicating an overflow condition. Depending on the state of the ROOE bit in the MCR, the data from the transfer that generated the overflow is either ignored or shifted in to the shift register. If the ROOE bit is set, the incoming data is shifted in to the shift register. If the ROOE bit is cleared, the incoming data is ignored.

30.5.2.5.2 Draining the RX FIFO

Host CPU or a DMA can remove (pop) entries from the RX FIFO by reading the module POP RX FIFO Register (POPR). A read of the POPR decrements the RX FIFO Counter by one. Attempts to pop data from an empty RX FIFO are ignored and the RX FIFO Counter remains unchanged. The data, read from the empty RX FIFO, is undetermined.

When the RX FIFO is not empty, the RX FIFO Drain Flag (RFDF) in the SR is set. The RFDF bit is cleared when the RX_FIFO is empty and the DMA controller indicates that a read from POPR is complete or by writing a 1 to it.

30.5.3 Module baud rate and clock delay generation

The SCK frequency and the delay values for serial transfer are generated by dividing the system clock frequency by a prescaler and a scaler with the option for doubling the baud rate. The following figure shows conceptually how the SCK signal is generated.

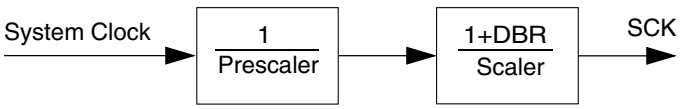


Figure 30-4. Communications clock prescalers and scalers

30.5.3.1 Baud rate generator

The baud rate is the frequency of the SCK. The protocol clock is divided by a prescaler (PBR) and scaler (BR) to produce SCK with the possibility of halving the scaler division. The DBR, PBR, and BR fields in the CTARs select the frequency of SCK by the formula in the BR field description. The following table shows an example of how to compute the baud rate.

Table 30-9. Baud rate computation example

f_P	PBR	Prescaler	BR	Scaler	DBR	Baud rate
100 MHz	0b00	2	0b0000	2	0	25 Mb/s
20 MHz	0b00	2	0b0000	2	1	10 Mb/s

NOTE

The clock frequencies mentioned in the preceding table are given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

30.5.3.2 PCS to SCK Delay (t_{CSC})

The PCS to SCK delay is the length of time from assertion of the PCS signal to the first SCK edge. See Figure 30-6 for an illustration of the PCS to SCK delay. The PCSSCK and CSSCK fields in the CTAR*x* registers select the PCS to SCK delay by the formula in the CSSCK field description. The following table shows an example of how to compute the PCS to SCK delay.

Table 30-10. PCS to SCK delay computation example

f_{SYS}	PCSSCK	Prescaler	CSSCK	Scaler	PCS to SCK Delay
100 MHz	0b01	3	0b0100	32	0.96 µs

NOTE

The clock frequency mentioned in the preceding table is given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

30.5.3.3 After SCK Delay (t_{ASC})

The After SCK Delay is the length of time between the last edge of SCK and the negation of PCS. See Figure 30-6 and Figure 30-7 for illustrations of the After SCK delay. The PASC and ASC fields in the CTAR*x* registers select the After SCK Delay by the formula in the ASC field description. The following table shows an example of how to compute the After SCK delay.

Table 30-11. After SCK Delay computation example

f_P	PASC	Prescaler	ASC	Scaler	After SCK Delay
100 MHz	0b01	3	0b0100	32	0.96 µs

NOTE

The clock frequency mentioned in the preceding table is given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

30.5.3.4 Delay after Transfer (t_{DT})

The Delay after Transfer is the minimum time between negation of the PCS signal for a frame and the assertion of the PCS signal for the next frame. See Figure 30-6 for an illustration of the Delay after Transfer. The PDT and DT fields in the CTAR*x* registers select the Delay after Transfer by the formula in the DT field description. The following table shows an example of how to compute the Delay after Transfer.

Table 30-12. Delay after Transfer computation example

f_P	PDT	Prescaler	DT	Scaler	Delay after Transfer
100 MHz	0b01	3	0b1110	32768	0.98 ms

NOTE

The clock frequency mentioned in the preceding table is given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

When in Non-Continuous Clock mode the t_{DT} delay is configured according to the equation specified in the CTAR[DT] field description. When in Continuous Clock mode, the delay is fixed at 1 SCK period.

30.5.3.5 Peripheral Chip Select Strobe Enable (PCSS_b)

The PCSS_b signal provides a delay to allow the PCS signals to settle after a transition occurs thereby avoiding glitches. When the Module is in Master mode and the PCSSE bit is set in the MCR, PCSS_b provides a signal for an external demultiplexer to decode peripheral chip selects other than PCS5 into glitch-free PCS signals. The following figure shows the timing of the PCSS_b signal relative to PCS signals.

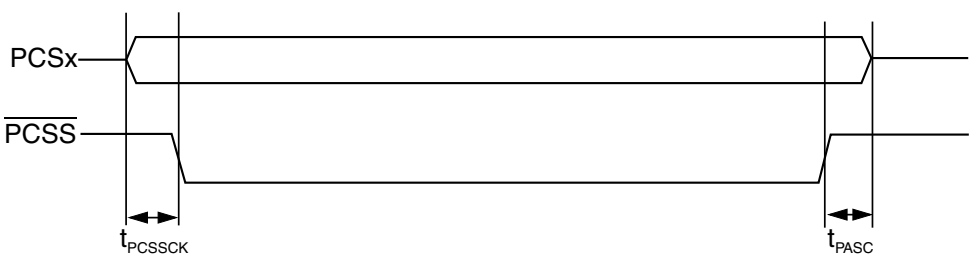


Figure 30-5. Peripheral Chip Select Strobe timing

The delay between the assertion of the PCS signals and the assertion of PCSS_b is selected by the PCSSCK field in the CTAR based on the following formula:

$$t_{\mathrm{PCSSCK}} = \frac{1}{f_{\mathrm{P}}} \times \mathrm{PCSSCK}$$

At the end of the transfer, the delay between PCSS_b negation and PCS negation is selected by the PASC field in the CTAR based on the following formula:

$$t_{\mathrm{PASC}} = \frac{1}{f_{\mathrm{P}}} \times \mathrm{PASC}$$

The following table shows an example of how to compute the t_{pcssck} delay.

Table 30-13. Peripheral Chip Select Strobe Assert computation example

f_P	PCSSCK	Prescaler	Delay before Transfer
100 MHz	0b11	7	70.0 ns

The following table shows an example of how to compute the t_{pasc} delay.

Table 30-14. Peripheral Chip Select Strobe Negate computation example

f_P	PASC	Prescaler	Delay after Transfer
100 MHz	0b11	7	70.0 ns

The PCSS_b signal is not supported when Continuous Serial Communication SCK mode is enabled.

NOTE

The clock frequency mentioned in the preceding tables is given as an example. Refer to the clocking chapter for the frequency used to drive this module in the device.

30.5.4 Transfer formats

The SPI serial communication is controlled by the Serial Communications Clock (SCK) signal and the PCS signals. The SCK signal provided by the master device synchronizes shifting and sampling of the data on the SIN and SOUT pins. The PCS signals serve as enable signals for the slave devices.

In Master mode, the CPOL and CPHA bits in the Clock and Transfer Attributes Registers (CTARn) select the polarity and phase of the serial clock, SCK.

- CPOL - Selects the idle state polarity of the SCK
- CPHA - Selects if the data on SOUT is valid before or on the first SCK edge

The module supports two different transfer formats:

- Classic SPI with CPHA=0
- Classic SPI with CPHA=1

In the interface configurations, the module provides the option of keeping the PCS signals asserted between frames. See Continuous Selection Format for details.

30.5.4.1 Classic SPI Transfer Format (CPHA = 0)

The transfer format shown in following figure is used to communicate with peripheral SPI slave devices where the first data bit is available on the first clock edge. In this format, the master and slave sample their SIN pins on the odd-numbered SCK edges and change the data on their SOUT pins on the even-numbered SCK edges.

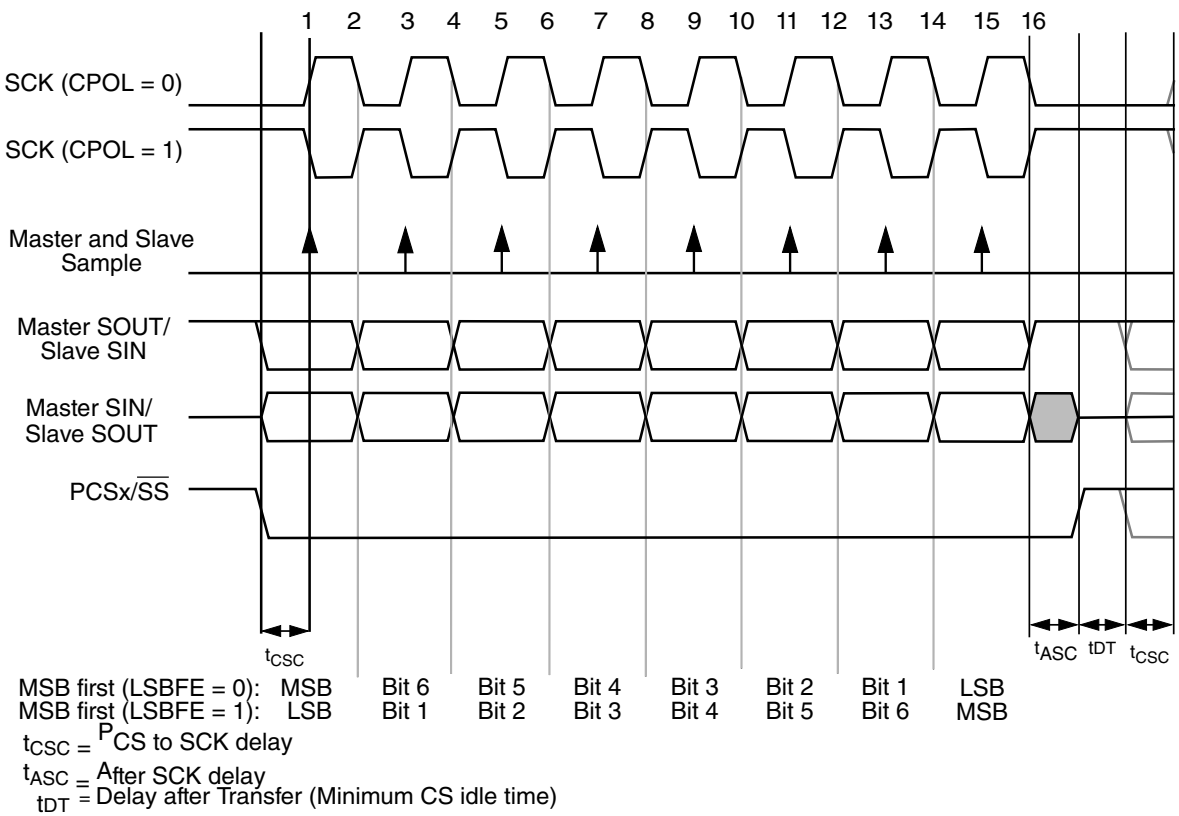


Figure 30-6. Module transfer timing diagram (CPHA=0, FMSZ=8)

The master initiates the transfer by placing its first data bit on the SOUT pin and asserting the appropriate peripheral chip select signals to the slave device. The slave responds by placing its first data bit on its SOUT pin. After the t_{CSC} delay elapses, the master outputs the first edge of SCK. The master and slave devices use this edge to sample the first input data bit on their serial data input signals. At the second edge of the SCK, the master and slave devices place their second data bit on their serial data output signals. For the rest of

the frame the master and the slave sample their SIN pins on the odd-numbered clock edges and changes the data on their SOUT pins on the even-numbered clock edges. After the last clock edge occurs, a delay of t_{ASC} is inserted before the master negates the PCS signals. A delay of t_{DT} is inserted before a new frame transfer can be initiated by the master.

30.5.4.2 Classic SPI Transfer Format (CPHA = 1)

This transfer format shown in the following figure is used to communicate with peripheral SPI slave devices that require the first SCK edge before the first data bit becomes available on the slave SOUT pin. In this format, the master and slave devices change the data on their SOUT pins on the odd-numbered SCK edges and sample the data on their SIN pins on the even-numbered SCK edges.

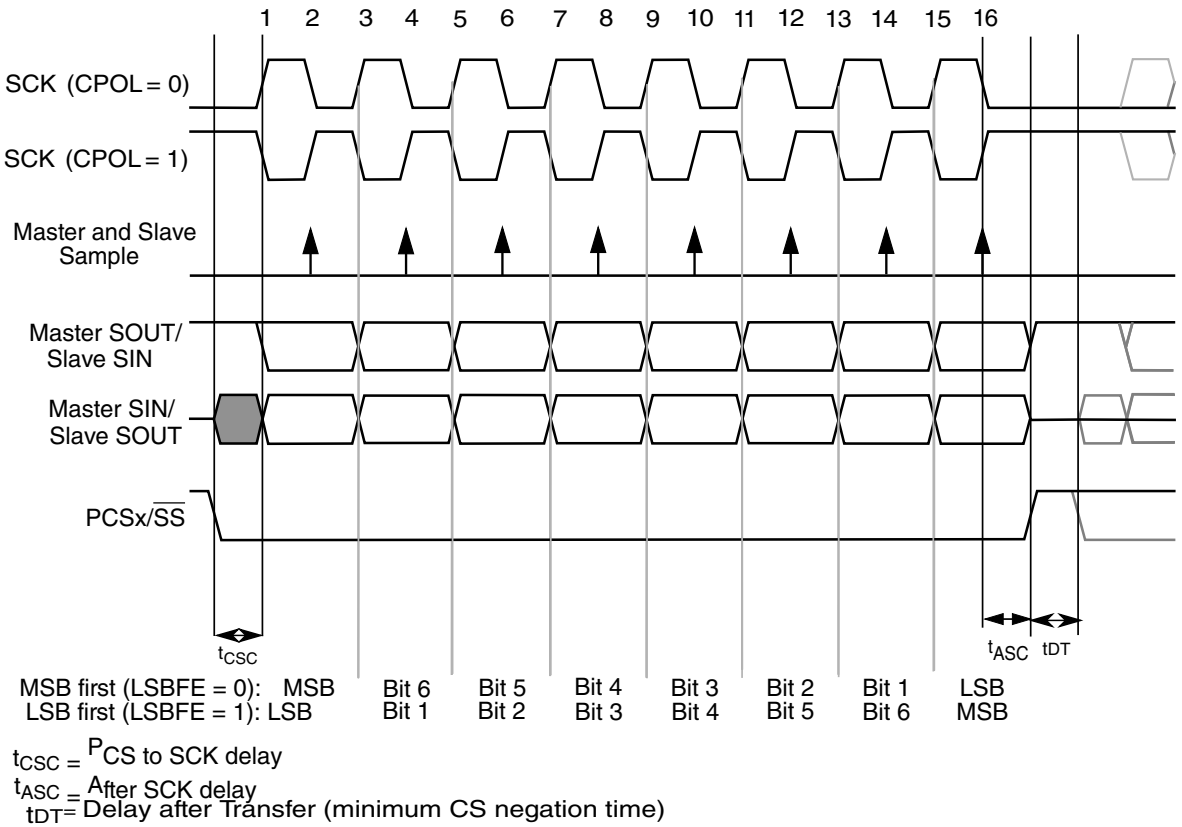


Figure 30-7. Module transfer timing diagram (CPHA=1, FMSZ=8)

The master initiates the transfer by asserting the PCS signal to the slave. After the t_{CSC} delay has elapsed, the master generates the first SCK edge and at the same time places valid data on the master SOUT pin. The slave responds to the first SCK edge by placing its first data bit on its slave SOUT pin.

At the second edge of the SCK the master and slave sample their SIN pins. For the rest of the frame the master and the slave change the data on their SOUT pins on the odd-numbered clock edges and sample their SIN pins on the even-numbered clock edges. After the last clock edge occurs, a delay of t_{ASC} is inserted before the master negates the PCS signal. A delay of t_{DT} is inserted before a new frame transfer can be initiated by the master.

30.5.4.3 Continuous Selection Format

Some peripherals must be deselected between every transfer. Other peripherals must remain selected between several sequential serial transfers. The Continuous Selection Format provides the flexibility to handle the following case. The Continuous Selection Format is enabled for the SPI configuration by setting the CONT bit in the SPI command.

When the CONT bit = 0, the module drives the asserted Chip Select signals to their idle states in between frames. The idle states of the Chip Select signals are selected by the PCSISn bits in the MCR. The following timing diagram is for two four-bit transfers with CPHA = 1 and CONT = 0.

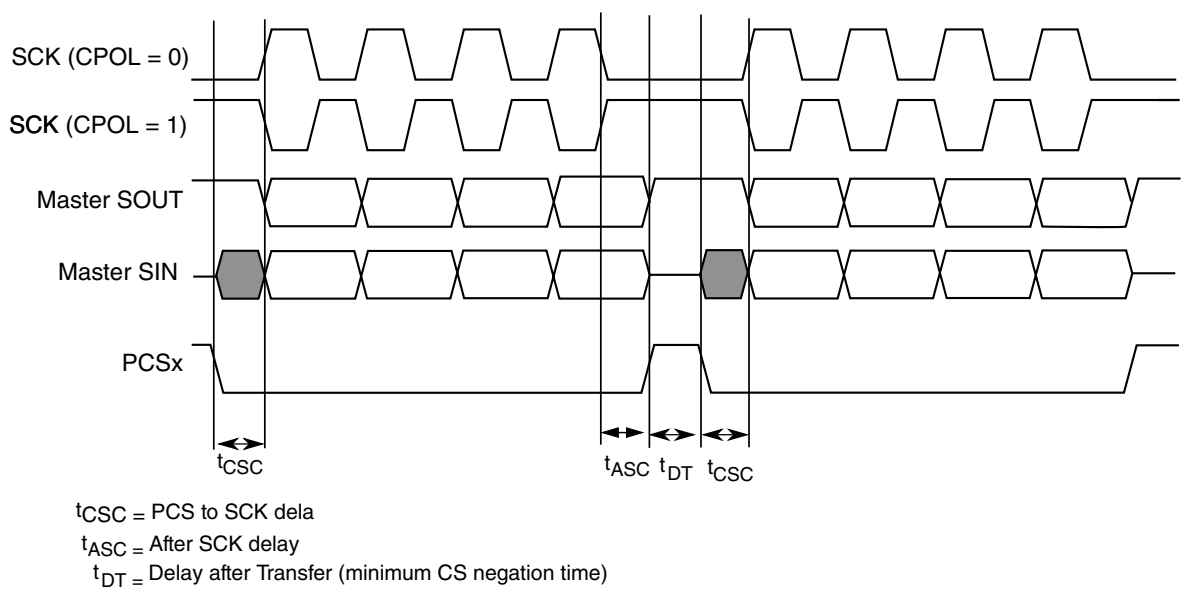


Figure 30-8. Example of non-continuous format (CPHA=1, CONT=0)

When the CONT bit = 1, the PCS signal remains asserted for the duration of the two transfers. The Delay between Transfers (t_{DT}) is not inserted between the transfers. The following figure shows the timing diagram for two four-bit transfers with CPHA = 1 and CONT = 1.

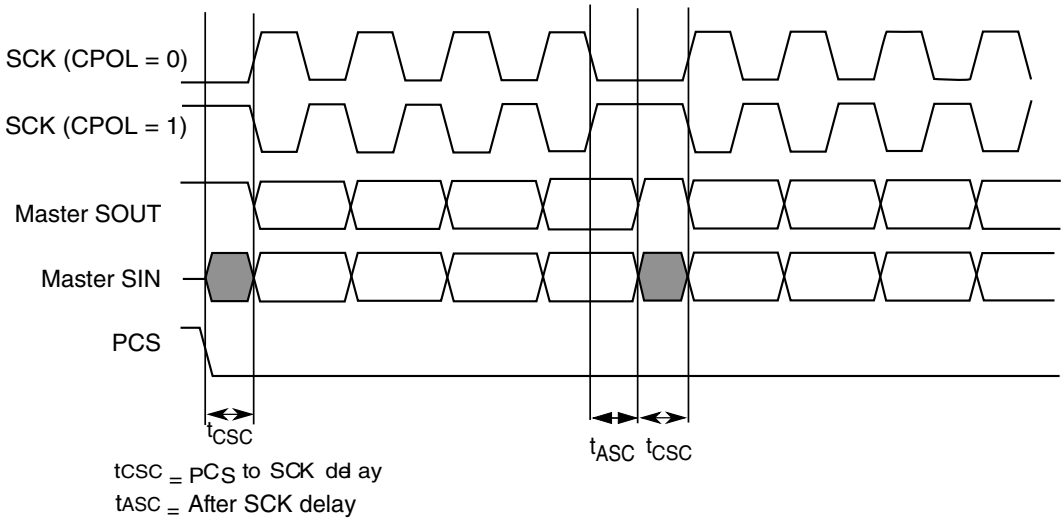


Figure 30-9. Example of continuous transfer (CPHA=1, CONT=1)

When using the module with continuous selection follow these rules:

- All transmit commands must have the same PCSn bits programming.
- The CTARs, selected by transmit commands, must be programmed with the same transfer attributes. Only FMSZ field can be programmed differently in these CTARs.
- When transmitting multiple frames in this mode, the user software must ensure that the last frame has the PUSHR[CONT] bit deasserted in Master mode and the user software must provide sufficient frames in the TX_FIFO to be sent out in Slave mode and the master deasserts the PCSn at end of transmission of the last frame.
- PUSHR[CONT] must be deasserted before asserting MCR[HALT] in master mode. This will make sure that the PCSn signals are deasserted. Asserting MCR[HALT] during continuous transfer will cause the PCSn signals to remain asserted and hence Slave Device cannot transition from Running to Stopped state.

NOTE

User must fill the TX FIFO with the number of entries that will be concatenated together under one PCS assertion for both master before the TX FIFO becomes empty.

30.5.5 Continuous Serial Communications Clock

The module provides the option of generating a Continuous SCK signal for slave peripherals that require a continuous clock.

Continuous SCK is enabled by setting the CONT_SCKE bit in the MCR. Enabling this bit generates the Continuous SCK only if MCR[HALT] bit is low. Continuous SCK is valid in all configurations.

Continuous SCK is only supported for CPHA=1. Clearing CPHA is ignored if the CONT_SCKE bit is set. .

Clock and transfer attributes for the Continuous SCK mode are set according to the following rules:

- When the module is in SPI configuration, CTAR0 is used initially. At the start of each SPI frame transfer, the CTAR specified by the CTAS for the frame is used.
- In all configurations, the currently selected CTAR remains in use until the start of a frame with a different CTAR specified, or the Continuous SCK mode is terminated.

It is recommended to keep the baud rate the same while using the Continuous SCK. Switching clock polarity between frames while using Continuous SCK can cause errors in the transfer. Continuous SCK operation is not guaranteed if the module is put into the External Stop mode or Module Disable mode.

Enabling Continuous SCK disables the PCS to SCK delay and the Delay after Transfer (t_{DT}) is fixed to one SCK cycle. The following figure is the timing diagram for Continuous SCK format with Continuous Selection disabled.

NOTE

In Continuous SCK mode, for the SPI transfer CTAR0 should always be used, and the TX FIFO must be cleared using the MCR[CLR_TXF] field before initiating transfer.

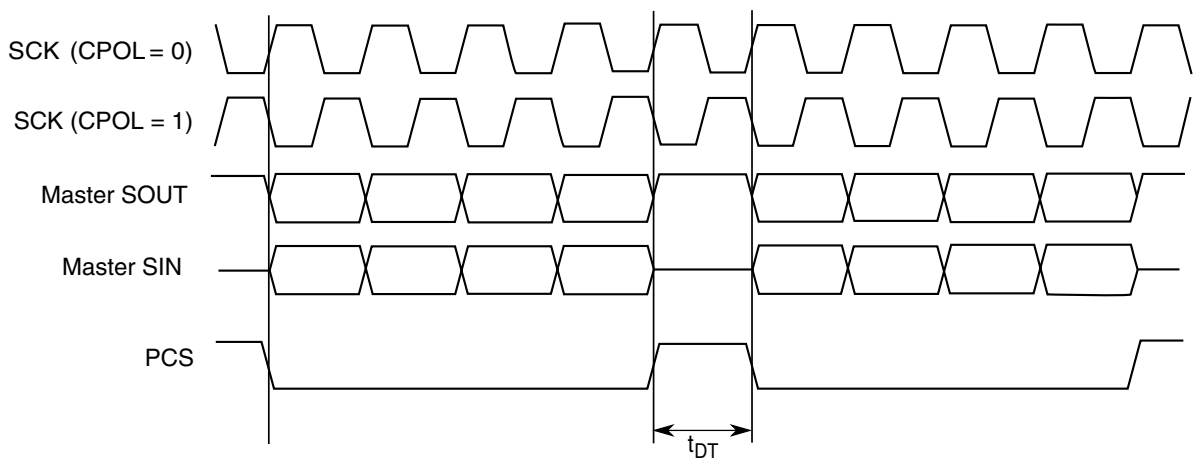


Figure 30-10. Continuous SCK Timing Diagram (CONT=0)

If the CONT bit in the TX FIFO entry is set, PCS remains asserted between the transfers. Under certain conditions, SCK can continue with PCS asserted, but with no data being shifted out of SOUT, that is, SOUT pulled high. This can cause the slave to receive incorrect data. Those conditions include:

- Continuous SCK with CONT bit set, but no data in the TX FIFO.
- Continuous SCK with CONT bit set and entering Stopped state (refer to Start and Stop of module transfers).
- Continuous SCK with CONT bit set and entering Stop mode or Module Disable mode.

The following figure shows timing diagram for Continuous SCK format with Continuous Selection enabled.

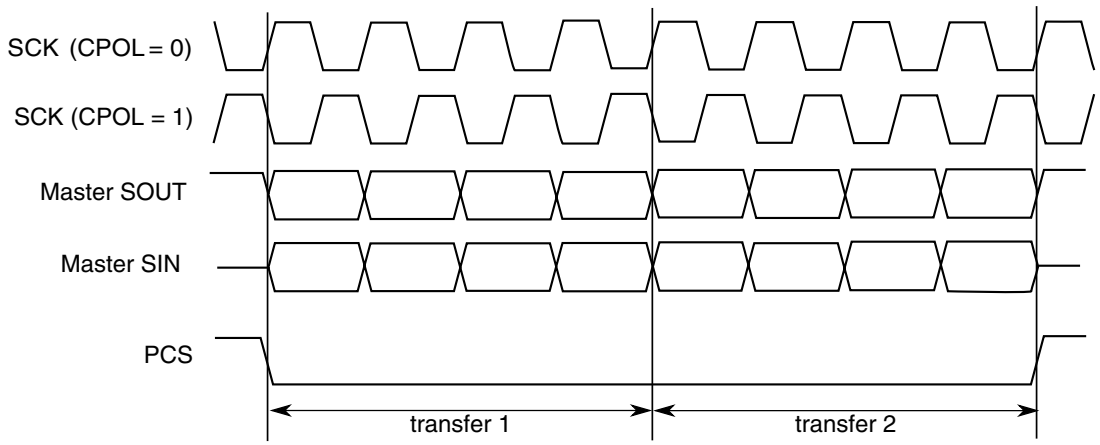


Figure 30-11. Continuous SCK timing diagram (CONT=1)

30.5.6 Parity Generation and Check

The module can generate and check parity in the serial frame. The parity bit replaces the last transmitted bit in the frame. The parity is calculated for all transmitted data bits in frame, not including the last data bit that would be transmitted. The parity generation/control is done on frame basis. The registers field setting frame size defines the total number of bits in the frame, including the parity bit. Thus, to transmit/receive the same number of data bits with parity check, increase the frame size by one versus the same data size frame without the parity check.

Parity can be selected as odd or even. Parity Errors in the received frame set Parity Error flags in the Status register. The Parity Error Interrupt Requests are generated if enabled. The module can be programmed to stop frame transmission in case of a frame reception with parity error.

30.5.6.1 Parity for SPI Frames

When the module is in the master mode the parity generation is controlled by PE and PP bits of the CMD FIFO entries (PUSHR). Setting the PE bit enables parity generation for transmitted SPI frames and parity check for received frames. PP bit defines polarity of the parity bit.

When continuous PCS selection is used to transmit SPI data, two parity generation scenarios are available:

- Generate/check parity for the whole frame
- Generate/check parity for each sub-frame separately.

To generate/check parity for the whole frame set PE bit only in the last command/TX FIFO entry, forming this frame (with the PUSHR register).

To generate/check parity for each sub-frame set PE bit in each command/TX FIFO entry, forming this frame.

If the parity error occurs for received SPI frame, the SR[SPEF] bit is set. If MCR[PES] bit is set, the module stops SPI frames transmission. To resume SPI operation clear the SR[SPEF] or the MCR[PES] bits.

30.5.7 Interrupts/DMA requests

The module has several conditions that can generate only interrupt requests and two conditions that can generate interrupt or DMA requests. The following table lists these conditions.

Table 30-15. Interrupt and DMA request conditions

Condition	Flag	Interrupt	DMA
End of Queue (EOQ)	EOQF	Yes	-
TX FIFO Fill	TFFF	Yes	Yes
CMD FIFO Fill	CMDFFF	Yes	Yes
TX FIFO Invalid Write	TFIWF	Yes	-
Transfer Complete	TCF	Yes	-
CMD Transfer Complete	CMDTCF	Yes	-
TX FIFO Underflow	TFUF	Yes	-
RX FIFO Drain	RFDF	Yes	Yes
RX FIFO Overflow	RFOF	Yes	-
SPI Parity Error	SPEF	Yes	-

Each condition has a flag bit in the module Status Register (SR) and a Request Enable bit in the DMA/Interrupt Request Select and Enable Register (RSER). Certain flags (as shown in above table) generate interrupt requests or DMA requests depending on configuration of RSER register.

The module also provides a global interrupt request line, which is asserted when any of individual interrupt requests lines is asserted.

30.5.7.1 End Of Queue interrupt request

The End Of Queue (EOQ) interrupt request indicates that the end of a transmit queue is reached. The module generates the interrupt request when EOQ interrupt requests are enabled (RSER[EOQF_RE]) and the EOQ bit in the executing SPI command is 1.

When Extended SPI mode is enabled (MCR[XSPI]) and the EOQ bit in the executing SPI command is 1, the module generates the EOQ interrupt request when the last bit of the last data frame in the command cycle has been transmitted.

When Extended SPI mode is disabled (MCR[XSPI]), the module generates the interrupt request when the last bit of the SPI frame with EOQ bit set is transmitted.

30.5.7.2 Transmit FIFO Fill Interrupt or DMA Request

The Transmit FIFO Fill Request indicates that the TX FIFO is not full. The Transmit FIFO Fill Request is generated when the number of entries in the TX FIFO is less than the maximum number of possible entries, and the TFFF_RE bit in the RSER is set. The TFFF_DIRS bit in the RSER selects whether a DMA request or an interrupt request is generated.

NOTE

TFFF flag clears automatically when DMA is used to fill TX FIFO. Configure the DMA to fill only one FIFO location per transfer.

To clear TFFF when not using DMA, follow these steps for every PUSH performed using CPU to fill TX FIFO:

1. Wait until TFFF = 1.
2. Write data to PUSHR using CPU.
3. Clear TFFF by writing a 1 to its location. If TX FIFO is not full, this flag will not clear.

30.5.7.3 Command FIFO Fill Interrupt or DMA Request

The Command FIFO Fill Request indicates that the CMD FIFO is not full. The Command FIFO Fill Request is generated when the number of entries in the CMD FIFO is less than the maximum number of possible entries, and the CMDFFF_RE bit in the RSER is set. The CMDFFF_DIRS bit in the RSER selects whether a DMA request or an interrupt request is generated.

This Request is useful when MCR[XSPI] is enabled, since TX FIFO and CMD FIFO can be filled independantly. If MCR[XSPI] is disabled, then 'TX FIFO Fill Interrupt or DMA Request' will suffice to fill both FIFO's since both FIFO's must be filled simultaneously.

NOTE

CMDFFF flag clears automatically when DMA is used to fill CMDFIFO.

To clear CMDFFF when not using DMA, follow these steps for every PUSH performed using CPU to fill CMDFIFO:

1. Wait until CMDFFF = 1
2. Write data to PUSHR using CPU.
3. Clear CMDFFF by writing a 1 to its location. If CMD FIFO is not full, this flag will not clear.

30.5.7.4 Transmit FIFO Invalid Write Interrupt Request

The Transmit FIFO Invalid Write Request is valid only when MCR[XSPI] is enabled. This Request indicates that Data exists in the TX FIFO while the CMD FIFO is empty. Since no Command Fields are associated with the Data present in TX FIFO, this data is considered invalid until a Command Entry becomes available. The Transmit FIFO Invalid Write Request is generated for the above condition when TFIWF_RE bit is set in the RSER.

30.5.7.5 Transfer Complete Interrupt Request

The Transfer Complete Request indicates the end of the transfer of a serial frame. The Transfer Complete Request is generated at the end of each frame transfer when the TCF_RE bit is set in the RSER.

30.5.7.6 Command Transfer Complete Interrupt Request

The Command Transfer Complete Request indicates the end of transfer of the last SPI frame in a Command Cycle. The Transfer Complete Request is generated for the above condition when the CMDTCF_RE bit is set in the RSER.

30.5.7.7 Receive FIFO Drain Interrupt or DMA Request

The Receive FIFO Drain Request indicates that the RX FIFO is not empty. The Receive FIFO Drain Request is generated when the number of entries in the RX FIFO is not zero, and the RFDF_RE bit in the RSER is set. The RFDF_DIRS bit in the RSER selects whether a DMA request or an interrupt request is generated. Configure the DMA to drain only one FIFO location per transfer.

30.5.7.8 Receive FIFO Overflow Interrupt Request

The Receive FIFO Overflow Request indicates that an overflow condition in the RX FIFO has occurred. A Receive FIFO Overflow request is generated when RX FIFO and shift register are full and a transfer is initiated. The RFOF_RE bit in the RSER must be set for the interrupt request to be generated.

Depending on the state of the ROOE bit in the MCR, the data from the transfer that generated the overflow is either ignored or shifted in to the shift register. If the ROOE bit is set, the incoming data is shifted in to the shift register. If the ROOE bit is cleared, the incoming data is ignored.

30.5.7.9 SPI Frame Parity Error Interrupt Request

The SPI Frame Parity Error Flag indicates that a SPI frame with parity error had been received. The SPEF_RE bit in the RSER must be set for the interrupt request to be generated.

30.5.8 Power saving features

The module supports following power-saving strategies:

- External Stop mode
- Module Disable mode – Clock gating of non-memory mapped logic

30.5.8.1 Stop mode (External Stop mode)

This module supports the Stop mode protocol. When a request is made to enter External Stop mode, the module acknowledges the request . If a serial transfer is in progress, then this module waits until it reaches the frame boundary before it is ready to have its clocks shut off . While the clocks are shut off, this module's memory-mapped logic is not accessible. This also puts the module in STOPPED state. The SR[TXRXS] bit is cleared to indicate STOPPED state. The states of the interrupt and DMA request signals cannot be changed while in External Stop mode.

30.5.8.2 Module Disable mode

Module Disable mode is a block-specific mode that the module can enter to save power. Host CPU can initiate the Module Disable mode by setting the MDIS bit in the MCR. The Module Disable mode can also be initiated by hardware.

When the MDIS bit is set, the module negates the Clock Enable signal at the next frame boundary. Once the Clock Enable signal is negated, it is said to have entered Module Disable Mode. This also puts the module in STOPPED state. The SR[TXRXS] bit is cleared to indicate STOPPED state.If implemented, the Clock Enable signal can stop the clock to the non-memory mapped logic. When Clock Enable is negated, the module is in a dormant state, but the memory mapped registers are still accessible. Certain read or write operations have a different effect when the module is in the Module Disable mode. Reading the RX FIFO Pop Register does not change the state of the RX FIFO. Similarly, writing to the PUSHR Register does not change the state of the TX FIFO or CMD FIFO. Clearing either of the FIFOs has no effect in the Module Disable mode. Changes to the DIS_TXF and DIS_RXF fields of the MCR have no effect in the Module Disable mode. In the Module Disable mode, all status bits and register flags in the module return the correct values when read, but writing to them has no effect. Writing to the TCR during Module Disable mode has no effect. Interrupt and DMA request signals cannot be cleared while in the Module Disable mode.

30.6 Initialization/application information

This section describes how to initialize the module.

30.6.1 How to manage queues

The queues are not part of the module, but it includes features in support of queue management. Queues are primarily supported in SPI configuration.

1. When module executes last command word from a queue, the EOQ bit in the command word is set to indicate it that this is the last entry in the queue.
2. At the end of the transfer, corresponding to the command word with EOQ set is sampled, the EOQ flag (EOQF) in the SR is set.
3. The setting of the EOQF flag disables serial transmission and reception of data, putting the module in the Stopped state. The TXRXS bit is cleared to indicate the Stopped state.
4. The DMA can continue to fill TX FIFO until it is full or step 5 occurs.
5. Disable DMA transfers by disabling the DMA enable request for the DMA channel assigned to TX FIFO (and CMD FIFO) and RX FIFO. This is done by clearing the corresponding DMA enable request bits in the DMA Controller.
6. Ensure all received data in RX FIFO has been transferred to memory receive queue by reading the RXCNT in SR or by checking RFDF in the SR after each read operation of the POPR.
7. Modify DMA descriptor of TX and RX channels for new queues
8. Flush TX FIFO (and CMD FIFO) by writing a 1 to the CLR_TXF bit in the MCR. Flush RX FIFO by writing a '1' to the CLR_RXF bit in the MCR.
9. Clear transfer count either by setting CTCNT bit in the command word of the first entry in the new queue or via CPU writing directly to SPI_TCNT field in the TCR.
10. Enable DMA channel by enabling the DMA enable request for the DMA channel assigned to the module TX FIFO, (and CMD FIFO) and RX FIFO by setting the corresponding DMA set enable request bit.
11. Enable serial transmission and serial reception of data by clearing the EOQF bit.

30.6.2 Initializing Module in Master Mode

Once the appropriate mode in MCR[MSTR] is configured, the module is enabled by clearing MCR[HALT]. It should be ensured that module Slave is enabled before enabling it's Master. This ensures the Slave is ready to be communicated with, before Master initializes communication.

30.6.3 Baud rate settings

The following table shows the baud rate that is generated based on the combination of the baud rate prescaler PBR and the baud rate scaler BR in the CTARs. The values calculated assume a 100 MHz protocol frequency and the double baud rate DBR bit is cleared.

NOTE

The clock frequency mentioned above is given as an example in this chapter. See the clocking chapter for the frequency used to drive this module in the device.

Table 30-16. Baud rate values (bps)

		Baud rate divider prescaler values			
		2	3	5	7
Baud Rate Scaler Values	2	25.0M	16.7M	10.0M	7.14M
	4	12.5M	8.33M	5.00M	3.57M
	6	8.33M	5.56M	3.33M	2.38M
	8	6.25M	4.17M	2.50M	1.79M
	16	3.12M	2.08M	1.25M	893k
	32	1.56M	1.04M	625k	446k
	64	781k	521k	312k	223k
	128	391k	260k	156k	112k
	256	195k	130k	78.1k	55.8k
	512	97.7k	65.1k	39.1k	27.9k
	1024	48.8k	32.6k	19.5k	14.0k
	2048	24.4k	16.3k	9.77k	6.98k
	4096	12.2k	8.14k	4.88k	3.49k
	8192	6.10k	4.07k	2.44k	1.74k
	16384	3.05k	2.04k	1.22k	872
	32768	1.53k	1.02k	610	436

30.6.4 Delay settings

The following table shows the values for the Delay after Transfer (t_{DT}) and CS to SCK Delay (T_{CSC}) that can be generated based on the prescaler values and the scaler values set in the CTARs. The values calculated assume a 100 MHz protocol frequency.

NOTE

The clock frequency mentioned above is given as an example in this chapter. See the clocking chapter for the frequency used to drive this module in the device.

Table 30-17. Delay values

		Delay prescaler values			
		1	3	5	7
Delay scaler values	2	20.0 ns	60.0 ns	100.0 ns	140.0 ns
	4	40.0 ns	120.0 ns	200.0 ns	280.0 ns
	8	80.0 ns	240.0 ns	400.0 ns	560.0 ns
	16	160.0 ns	480.0 ns	800.0 ns	1.1 µs
	32	320.0 ns	960.0 ns	1.6 µs	2.2 µs
	64	640.0 ns	1.9 µs	3.2 µs	4.5 µs
	128	1.3 µs	3.8 µs	6.4 µs	9.0 µs
	256	2.6 µs	7.7 µs	12.8 µs	17.9 µs
	512	5.1 µs	15.4 µs	25.6 µs	35.8 µs
	1024	10.2 µs	30.7 µs	51.2 µs	71.7 µs
	2048	20.5 µs	61.4 µs	102.4 µs	143.4 µs
	4096	41.0 µs	122.9 µs	204.8 µs	286.7 µs
	8192	81.9 µs	245.8 µs	409.6 µs	573.4 µs
	16384	163.8 µs	491.5 µs	819.2 µs	1.1 ms
	32768	327.7 µs	983.0 µs	1.6 ms	2.3 ms
	65536	655.4 µs	2.0 ms	3.3 ms	4.6 ms

30.6.5 Calculation of FIFO pointer addresses

Complete visibility of the FIFO contents is available through the FIFO registers, and valid entries can be identified through a memory-mapped pointer and counter for each FIFO. The pointer to the first-in entry in each FIFO is memory mapped. For the TX FIFO the first-in pointer is the Transmit Next Pointer (TXNXTPTR). For the CMD FIFO the first-in pointer is the Command Next Pointer (CMDNXTPTR). For the RX FIFO the first-in pointer is the Pop Next Pointer (POPNXTPTR). The following figure illustrates the concept of first-in and last-in FIFO entries along with the FIFO Counter. The TX FIFO is chosen for the illustration, but the concepts carry over. See Transmit First In First

Out (TX FIFO) buffering mechanism, Command First In First Out (CMD FIFO) Buffering Mechanism and Receive First In First Out (RX FIFO) buffering mechanism for details on the FIFO operation.

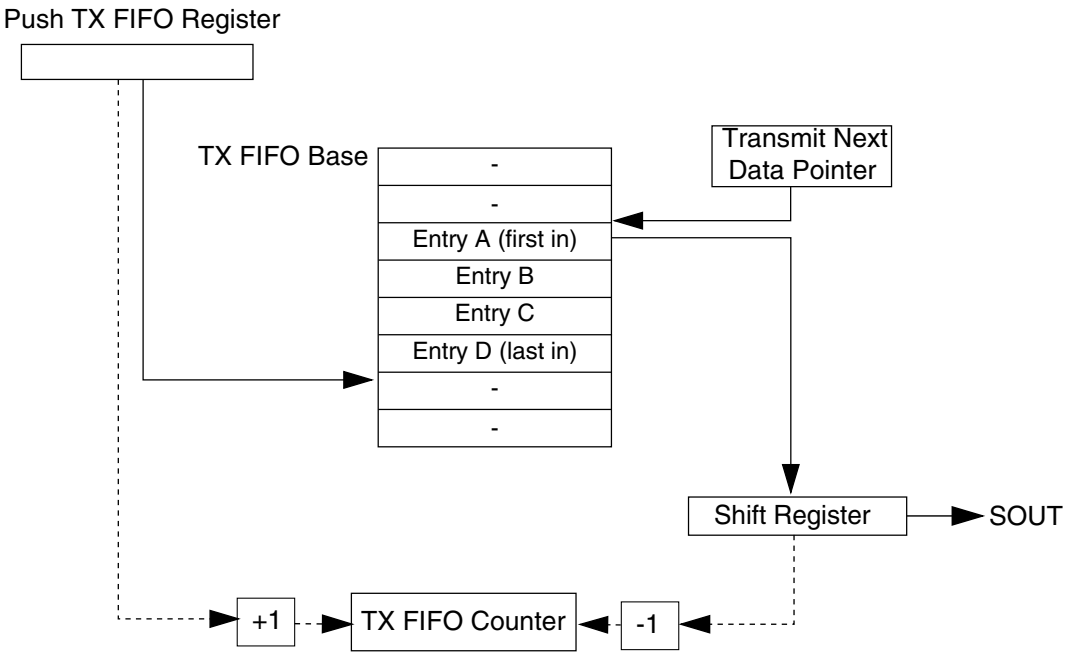


Figure 30-12. TX FIFO pointers and counter

30.6.5.1 Address Calculation for the First-in Entry and Last-in Entry in the TX FIFO

The memory address of the first-in entry in the TX FIFO is computed by the following equation:

$$\text{First-in EntryAddress} = \text{TXFIFOBase} + \left(4 \times \text{TXNXTPTR}\right)$$

The memory address of the last-in entry in the TX FIFO is computed by the following equation:

$$\text{Last-inEntryaddress} = \text{TXFIFOBase} + 4 \times \left(\text{TXCTR} + \text{TXNXTPTR} - 1\right) \bmod \left(\text{TXFIFOdepth}\right)$$

TX FIFO Base - Base address of TX FIFO
TXCTR - TX FIFO Counter
TXNXTPTR - Transmit Next Pointer
TX FIFO Depth - Transmit FIFO depth, implementation specific

30.6.5.2 Address Calculation for the First-in Entry and Last-in Entry in the CMD FIFO

The memory address of the first-in entry in the CMD FIFO is computed by the following equation:

$$\text{First-in EntryAddress} = \text{TXFIFOBase} + (4 \times \text{TXNXTPTR})$$

The memory address of the last-in entry in the CMD FIFO is computed by the following equation:

$$\text{Last-inEntryaddress} = \text{TXFIFOBase} + 4 \times (\text{TXCTR} + \text{TXNXTPTR} - 1) \bmod (\text{TXFIFOdepth})$$

CMD FIFO Base - Base address of CMD FIFO
CMDCTR - CMD FIFO Counter
CMDNXTPTR - Command Next Pointer
CMD FIFO Depth - Command FIFO depth, implementation specific

30.6.5.3 Address Calculation for the First-in Entry and Last-in Entry in the RX FIFO

The memory address of the first-in entry in the RX FIFO is computed by the following equation:

$$\text{First-in EntryAddress} = \text{RX FIFOBase} + (4 \times \text{POPNXTPTR})$$

The memory address of the last-in entry in the RX FIFO is computed by the following equation:

$$\text{Last-inEntryaddress} = \text{RX FIFO Base} + 4 \times (\text{RXCTR} + \text{POPNXTPTR} - 1) \bmod (\text{RXFIFOdepth})$$

RX FIFO Base - Base address of RX FIFO
RXCTR - RX FIFO counter
POPNXTPTR - Pop Next Pointer
RX FIFO Depth - Receive FIFO depth, implementation specific

Chapter 31
Thermal Monitoring Unit (TMU)

31.1 The TMU module as implemented on the chip

This section provides details about how the TMU module is implemented on the chip.

31.1.1 Local Temperature Sensor Placement

The table below shows the placement of the local temperature sensor:

Table 31-1. Local Temperature Sensor Placement in Chip

Temperature Sensor ID	Placement
0	Near A53 core

31.1.2 Initialization Information

The TMU calculates temperature by reading one of the temperature sensor sites on the chip. The temperature is calculated from a calibration table, which is programmed either by the pre-boot loader or initialization of calibration table through software. Failure to properly initialize the calibration table may lead to an undefined behavior. Calibration determines specific sensor reading translated in degrees Celsius as shown in the following tables.

Table 31-2. Temperature calibration points

Register Name	Value
TTR0CR, 12 points at 0°C	
Point 0 at 0°C	
TTCFGR	0x0000_0000

Table continues on the next page...

Table 31-2. Temperature calibration points (continued)

Register Name	Value
TSCFGR	0x0000_0025
Point 1 at 4°C	
TTCFGR	0x0000_0001
TSCFGR	0x0000_002C
Point 2 at 8°C	
TTCFGR	0x0000_0002
TSCFGR	0x0000_0032
Point 3 at 12°C	
TTCFGR	0x0000_0003
TSCFGR	0x0000_0039
Point 4 at 16°C	
TTCFGR	0x0000_0004
TSCFGR	0x0000_003F
Point 5 at 20°C	
TTCFGR	0x0000_0005
TSCFGR	0x0000_0046
Point 6 at 24°C	
TTCFGR	0x0000_0006
TSCFGR	0x0000_004C
Point 7 at 28°C	
TTCFGR	0x0000_0007
TSCFGR	0x0000_0053
Point 8 at 32°C	
TTCFGR	0x0000_0008
TSCFGR	0x0000_0059
Point 9 at 36°C	
TTCFGR	0x0000_0009
TSCFGR	0x0000_005F
Point 10 at 40°C	
TTCFGR	0x0000_000A
TSCFGR	0x0000_0066
Point 11 at 44°C	
TTCFGR	0x0000_000B
TSCFGR	0x0000_006C
TTR1CR, 10 points at 42°C	
Point 0 at 42°C	
TTCFGR	0x0000_0000
TSCFGR	0x0000_0026
Point 1 at 46°C	

Table continues on the next page...

Table 31-2. Temperature calibration points (continued)

Register Name	Value
TTCFGR	0x0000_0001
TSCFGR	0x0000_002D
Point 2 at 50°C	
TTCFGR	0x0000_0002
TSCFGR	0x0000_0035
Point 3 at 54°C	
TTCFGR	0x0000_0003
TSCFGR	0x0000_003D
Point 4 at 58°C	
TTCFGR	0x0000_0004
TSCFGR	0x0000_0045
Point 5 at 62°C	
TTCFGR	0x0000_0005
TSCFGR	0x0000_004D
Point 6 at 66°C	
TTCFGR	0x0000_0006
TSCFGR	0x0000_0055
Point 7 at 70°C	
TTCFGR	0x0000_0007
TSCFGR	0x0000_005D
Point 8 at 74°C	
TTCFGR	0x0000_0008
TSCFGR	0x0000_0065
Point 9 at 78°C	
TTCFGR	0x0000_0009
TSCFGR	0x0000_006D
TTR2CR, 7 points at 76°C	
Point 0 at 76°C	
TTCFGR	0x0000_0000
TSCFGR	0x0000_0026
Point 1 at 80°C	
TTCFGR	0x0000_0001
TSCFGR	0x0000_0030
Point 2 at 84°C	
TTCFGR	0x0000_0002
TSCFGR	0x0000_003A
Point 3 at 88°C	
TTCFGR	0x0000_0003
TSCFGR	0x0000_0044

Table continues on the next page...

Table 31-2. Temperature calibration points (continued)

Register Name	Value
Point 4 at 92°C	
TTCFGR	0x0000_0004
TSCFGR	0x0000_004E
Point 5 at 96°C	
TTCFGR	0x0000_0005
TSCFGR	0x0000_0059
Point 6 at 100°C	
TTCFGR	0x0000_0006
TSCFGR	0x0000_0063
TTR3CR, 7 points at 98°C	
Point 0 at 98°C	
TTCFGR	0x0000_0000
TSCFGR	0x0000_0014
Point 1 at 102°C	
TTCFGR	0x0000_0001
TSCFGR	0x0000_0021
Point 2 at 106°C	
TTCFGR	0x0000_0002
TSCFGR	0x0000_002E
Point 3 at 110°C	
TTCFGR	0x0000_0003
TSCFGR	0x0000_003A

The process for programming the calibration table is described below:

1. Disable the monitoring mode, TMR[ME]=0 (default).
2. Program TTR0CR=0x000B_0000, TTR1CR=0x0009_002A, TTR2CR=0x0006_004C and TTR3CR=0x0003_0062.
3. Write the temperature configuration register TTCFGR with value from table.
4. Write the sensor configuration register, TSCFGR, with the sensor value from table.
5. Repeat steps 2-3 for all temperatures as defined in the above table.

31.2 Thermal Monitoring Unit Introduction

The Thermal Monitoring Unit (TMU) monitors and reports the temperature from one or more remote temperature measurement sites located on chip.

31.2.1 TMU Overview

The TMU has access to multiple temperature measurement sites strategically located on the chip. It monitors these sites and can signal an alarm if a programmed threshold is ever exceeded. The upper and lower temperature range is continuously captured. A set of reporting registers allow for reading the current temperature at monitored sites.

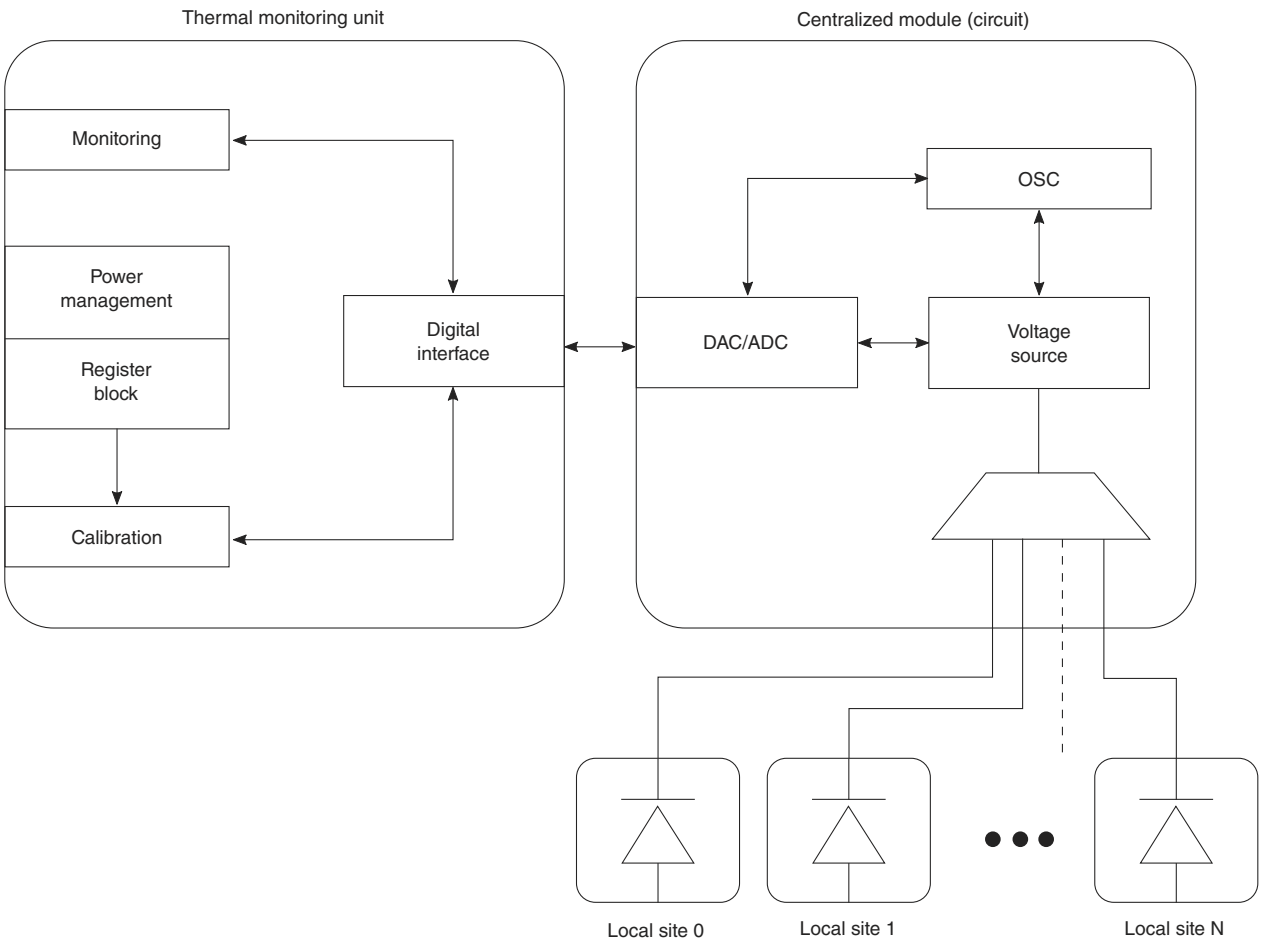


Figure 31-1. Thermal Monitoring Unit Block Diagram

31.2.2 Features

The temperature management unit features:

- Temperature measurement range 0-110°C.
- Calibration
  - Calibration table loaded from boot code ROM using pre-boot loader *or* initialization of calibration table through software
- Monitoring
  - Single-, or multi-site monitoring

  - Programmable monitoring interval
  - Out-of-range indication
  - High/low temperature range monitoring
  - Immediate and average temperature monitoring
  - Average temperature monitoring programmable low-pass filtering
  - Programmable monitoring thresholds for normal and critical alarm
- Reporting
  - Immediate and average temperature reporting for all monitor sites

31.2.3 Modes of Operation

The TMU has one mode of operation:

- Monitoring

The mode register monitoring enable bit, TMR[ME], determines if the unit is in active monitoring mode or in power saving mode.

The table below describes bit settings required for each TMU mode of operation.

Table 31-3. TMU Mode Bit Setting

Modes with Features	TMR[ME]
Monitoring mode disabled	0
Monitoring mode enabled	1

31.3 TMU register descriptions

The table shows the memory map for management of the TMU resources.

31.3.1 TMU Memory map

TMU base address: 1F0_0000h

Offset	Register	Width (In bits)	Access	Reset value
0h	TMU mode register (TMR)	32	RW	0000_0000h

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
4h	TMU status register (TSR)	32	RO	0000_0000h
8h	TMU monitor temperature measurement interval register (TMTMIR)	32	RW	0000_0000h
20h	TMU interrupt enable register (TIER)	32	RW	0000_0000h
24h	TMU interrupt detect register (TIDR)	32	W1C	0000_0000h
28h	TMU interrupt site capture register (TISCR)	32	RW	0000_0000h
2Ch	TMU interrupt critical site capture register (TICSCR)	32	RW	0000_0000h
40h	TMU monitor high temperature capture register (TMHTCR)	32	RO	0000_0000h
44h	TMU monitor low temperature capture register (TMLTCR)	32	RO	0000_0000h
50h	TMU monitor high temperature immediate threshold register (TMHTITR)	32	RW	0000_0000h
54h	TMU monitor high temperature average threshold register (TMHTATR)	32	RW	0000_0000h
58h	TMU monitor high temperature average critical threshold register (TMHTACTR)	32	RW	0000_0000h
80h	TMU temperature configuration register (TTCFGR)	32	RW	0000_0000h
84h	TMU sensor configuration register (TSCFGR)	32	RW	0000_0000h
100h	TMU report immediate temperature site register 0 (TRITSR0)	32	RO	0000_0000h
104h	TMU report average temperature site register 0 (TRATSR0)	32	RO	0000_0000h
F10h	TMU temperature range 0 control register (TTR0CR)	32	RW	000B_0000h
F14h	TMU temperature range 1 control register (TTR1CR)	32	RW	000A_0026h
F18h	TMU temperature range 2 control register (TTR2CR)	32	RW	0008_0048h
F1Ch	TMU temperature range 3 control register (TTR3CR)	32	RW	0007_0061h

31.3.2 TMU mode register (TMR)

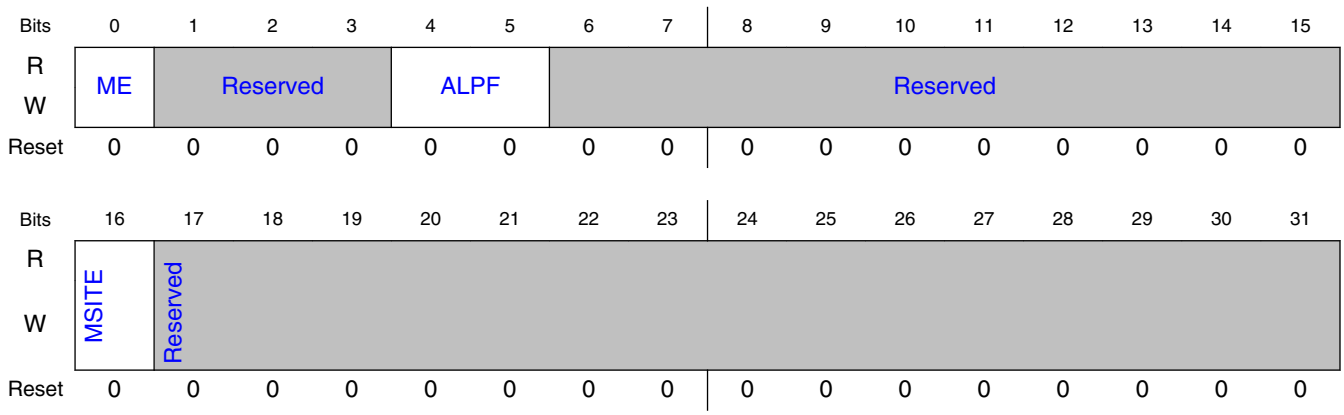
31.3.2.1 Offset

Register	Offset
TMR	0h

31.3.2.2 Function

The TMU mode register allows software to control the operation of the thermal monitoring.

31.3.2.3 Diagram



31.3.2.4 Fields

Field	Function
0 ME	Monitoring mode enable. 0 No monitoring. 1 Monitoring of sites as defined by field MSITE. Before enabling the TMU for monitoring, the TMU must be configured, see section Initialization Information. Failure to properly initialize the configuration table may result in boundedly undefined behavior.
1-3 —	Reserved.
4-5 ALPF	Average low pass filter setting. 00 1.0 01 0.5 10 0.25 11 0.125 The average temperature is calculated as: ALPF x Current_Temp + (1 - ALPF) x Average_Temp. If no previous (average) temperature is valid, current temperature is used. For proper operation, this field should only change when monitoring is disabled.
6-15 —	Reserved.
16 MSITE	Monitoring site select 0. By setting the select bit for a temperature sensor site, it is enabled and included in all monitoring functions. For proper operation, this field should only change when monitoring is disabled. If no site is selected, site 0 is monitored by default.
17-31 —	Reserved.

31.3.3 TMU status register (TSR)

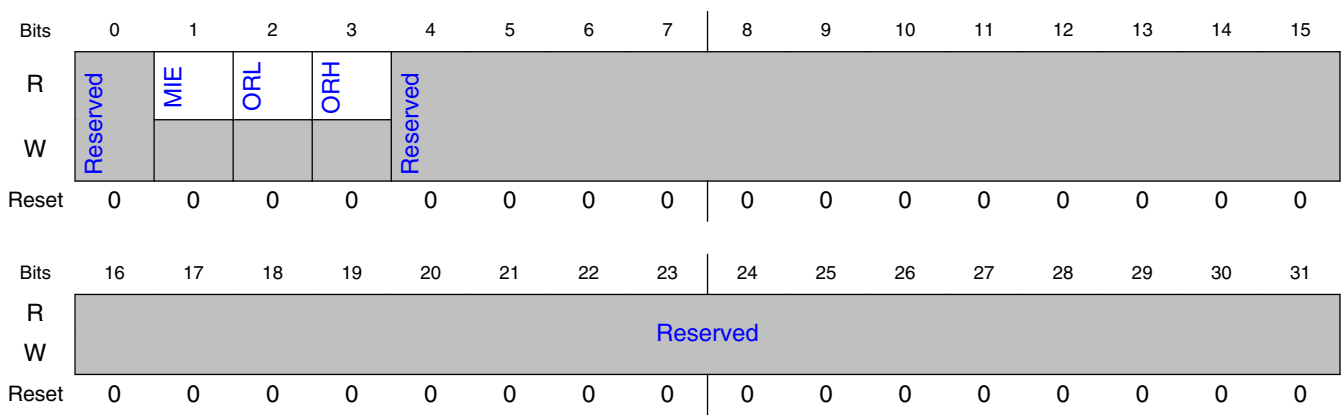
31.3.3.1 Offset

Register	Offset
TSR	4h

31.3.3.2 Function

The TMU status register reports the monitoring and calibration status during operation.

31.3.3.3 Diagram



31.3.3.4 Fields

Field	Function
0 —	Reserved.
1 MIE	Monitoring interval exceeded. 0 Monitoring interval not exceeded. 1 Monitoring interval exceeded. The time required to perform measurement of all monitored sites has exceeded the monitoring interval as defined by TMTMIR.

Table continues on the next page...

Field	Function
	This bit will clear automatically when TMU monitoring is (re-)enabled or the monitoring interval register, TMTMIR, is written.
2 ORL	Out-of-range low temperature measurement detected. A temperature sensor detected a temperature reading below the lowest measurable temperature of 0 degrees Celsius. This bit will clear automatically when TMU monitoring is (re-)enabled.
3 ORH	Out-of-range high temperature measurement detected. A temperature sensor detected a temperature reading above the highest measurable temperature of 110 °C. This bit will clear automatically when TMU monitoring is (re-)enabled.
4-31 —	Reserved.

31.3.4 TMU monitor temperature measurement interval register (TMTMIR)

31.3.4.1 Offset

Register	Offset
TMTMIR	8h

31.3.4.2 Function

The TMU monitor temperature measurement interval register determines at what frequency temperature sensors are read. All enabled monitored sites are read once in the duration of the time interval. The status bit TSR[MIE] will be set if the temperature measument takes longer than the set interval. Software should consider increasing the interval or reducing the number of active sites if the interval is exceeded. Disabling the interval allows for continuous monitoring.

NOTE

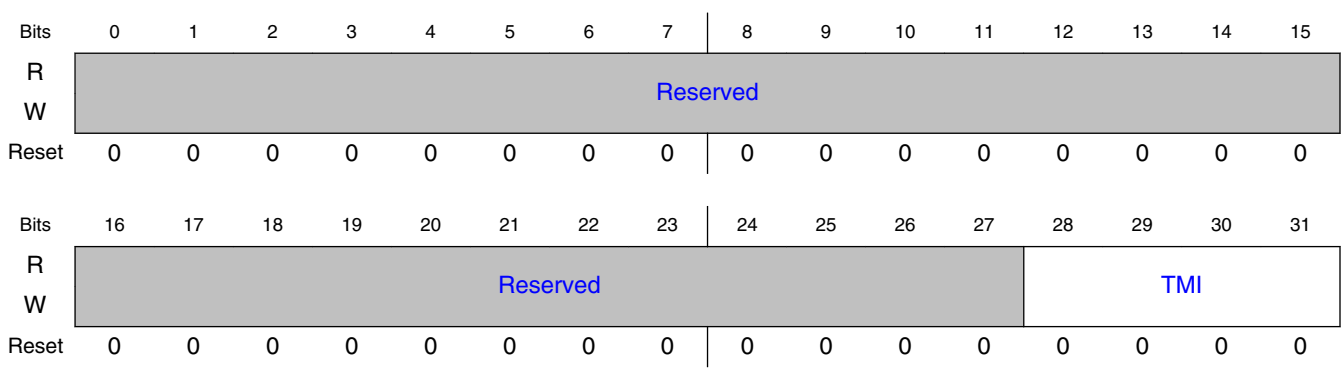
The time it takes for one temperature measurement is dependent on the temperatures measured and mode settings, thus there is no fixed time advertised for a single temperature measurements.

The following table lists temperature measurement intervals.

Table 31-4. Temperature measurement interval

TMI bit field settings	Platform clock frequency
	250 MHz
0000	0.034 s
0001	0.067 s
0010	0.134 s
0011	0.27 s
0100	0.54 s
0101	1.07 s
0110	2.15 s
0111	4.3 s
1000	8.6 s
1001	17 s
1010	34 s
1011	68 s
1100	137 s
1101	275 s
1110	550 s
1111	Disabled

31.3.4.3 Diagram



31.3.4.4 Fields

Field	Function
0-27 —	Reserved.

Table continues on the next page...

Field	Function
28-31 TMI	Temperature monitoring interval in seconds. For proper operation, this field should only change when monitoring is disabled, TMR[ME]=0. For lower platform speeds, the time increases proportionally, while the opposite is true for higher platform speeds.

31.3.5 TMU interrupt enable register (TIER)

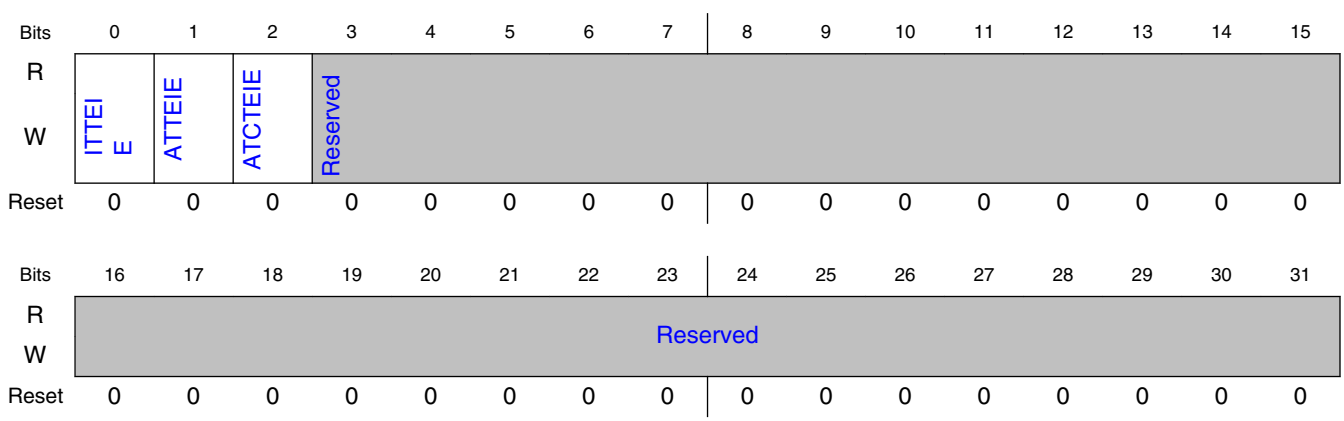
31.3.5.1 Offset

Register	Offset
TIER	20h

31.3.5.2 Function

The TMU interrupt enable register determines if a detected status condition should cause a system interrupt. A system interrupt occurs if a bit in this register is set and the corresponding bit in the interrupt detect register is also set. To clear the interrupt, write a 1 to the interrupt detect register.

31.3.5.3 Diagram



31.3.5.4 Fields

Field	Function
0 ITTEIE	Immediate temperature threshold exceeded interrupt enable. 0 Disabled. 1 Interrupt enabled. Generate an interrupt if TIDR[ITTE] is set.
1 ATTEIE	Average temperature threshold exceeded interrupt enable. 0 Disabled. 1 Interrupt enabled. Generate an interrupt if TIDR[ATTE] is set.
2 ATCTEIE	Average temperature critical threshold exceeded interrupt enable. 0 Disabled. 1 Interrupt enabled. Generate an interrupt if TIDR[ATCTE] is set.
3-31 —	Reserved.

31.3.6 TMU interrupt detect register (TIDR)

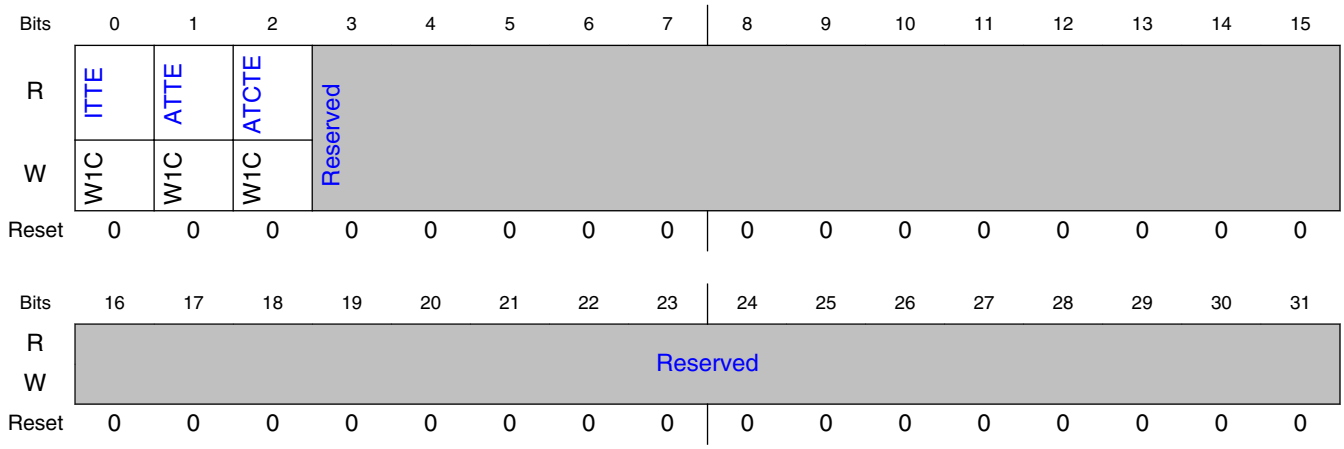
31.3.6.1 Offset

Register	Offset
TIDR	24h

31.3.6.2 Function

The TMU interrupt detect register indicates if an status condition was detected that could generate an interrupt. Write 1 to clear the detected condition and the interrupt, if enabled.

31.3.6.3 Diagram



31.3.6.4 Fields

Field	Function
0 ITTE	Immediate temperature threshold exceeded. Write 1 to clear. 0 No threshold exceeded. 1 Immediate temperature threshold, as defined by TMHTITR, has been exceeded by one or more monitored sites. This includes an out-of-range measured temperature above 110 °C. The sites which has exceeded the threshold are captured in TISCR[ISITE].
1 ATTE	Average temperature threshold exceeded. Write 1 to clear. 0 No threshold exceeded. 1 Average temperature threshold, as defined by TMHTATR, has been exceeded by one or more monitored sites. The sites which has exceeded the threshold are captured in TISCR[ASITE].
2 ATCTE	Average temperature critical threshold exceeded. Write 1 to clear. 0 No threshold exceeded. 1 Average temperature critical threshold, as defined by TMHTACTR, has been exceeded by one or more monitored sites. The sites which has exceeded the threshold are captured in TICSCR[CASITE].
3-31 —	Reserved.

31.3.7 TMU interrupt site capture register (TISCR)

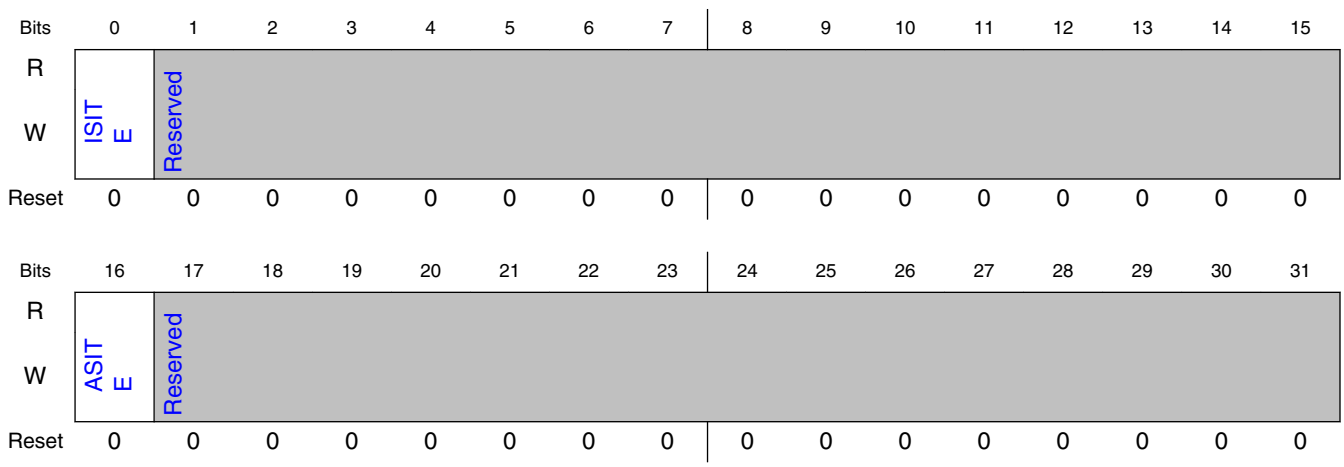
31.3.7.1 Offset

Register	Offset
TISCR	28h

31.3.7.2 Function

The TMU interrupt site capture register holds information about the temperature sensor site associated with a detected interrupt event.

31.3.7.3 Diagram



31.3.7.4 Fields

Field	Function
0 ISITE	Temperature sensor site associated with the setting of TIDR[ITTE]. This field has the same bit representation as TMR[MSITE]. Software should clear this field after handling the detected interrupt events ITTE.
1-15 —	Reserved.
16 ASITE	Temperature sensor site associated with the setting of TIDR[ATTE] . This field has the same bit representation as TMR[MSITE]. Software should clear this field after handling the detected interrupt event ATTE.
17-31 —	Reserved.

31.3.8 TMU interrupt critical site capture register (TICSCR)

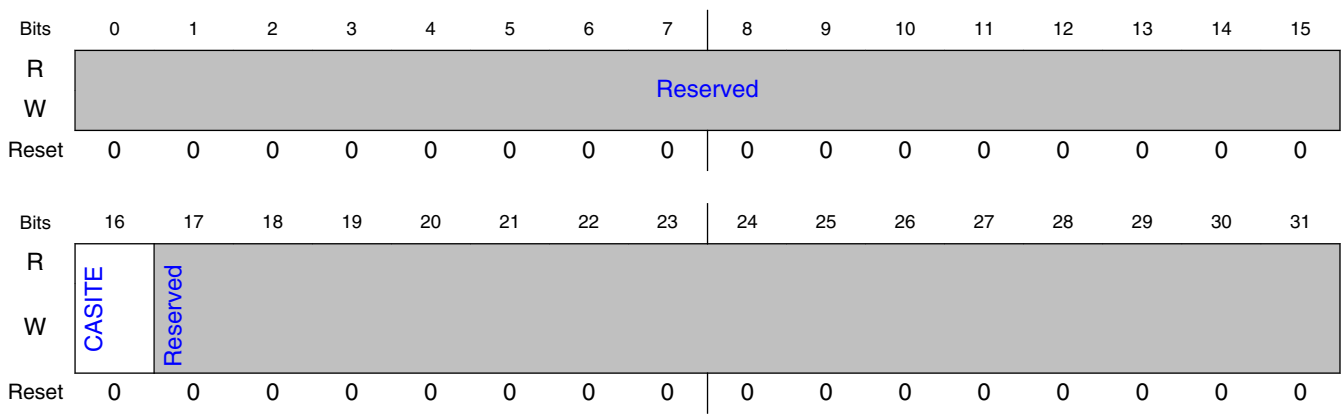
31.3.8.1 Offset

Register	Offset
TICSCR	2Ch

31.3.8.2 Function

The TMU interrupt critical site capture register holds information about the temperature sensor site associated with a detected critical interrupt event.

31.3.8.3 Diagram



31.3.8.4 Fields

Field	Function
0-15 —	Reserved.
16 CASITE	Temperature sensor site associated with the setting of TIDR[ATCTE] . This field has the same bit representation as TMR[MSITE]. Software should clear this field after handling the detected critical interrupt event ATCTE.
17-31	Reserved.

Field	Function
—	

31.3.9 TMU monitor high temperature capture register (TMHTCR)

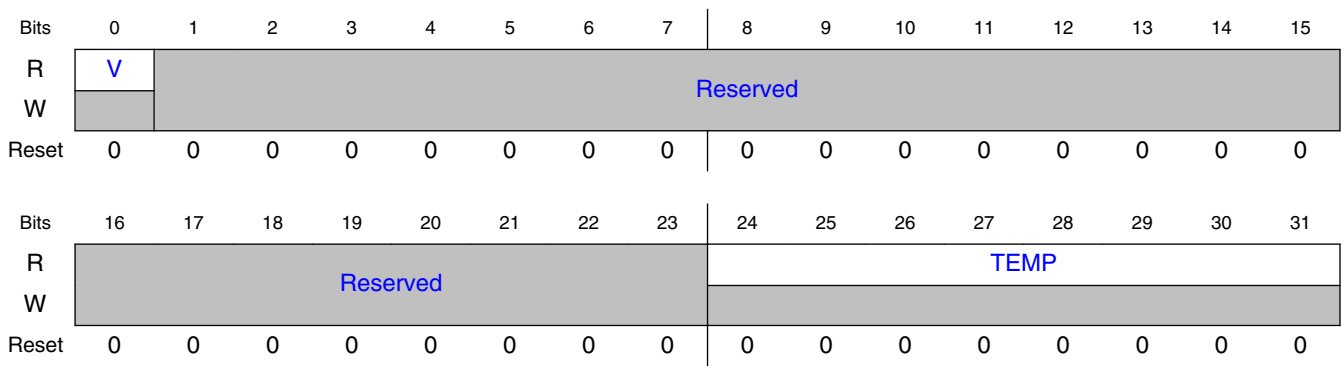
31.3.9.1 Offset

Register	Offset
TMHTCR	40h

31.3.9.2 Function

This TMU monitor register captures and record the highest temperature reached for any one enabled monitored site within temperature sensor range.

31.3.9.3 Diagram



31.3.9.4 Fields

Field	Function
0	Valid reading.
V	0 Temperature reading is not valid due to no measured temperature within the sensor range of 0-110 °C for an enabled monitored site. 1 Temperature reading is valid.

Table continues on the next page...

Field	Function
	(Re-)enabling the TMU will automatically clear this bit and start a new search.
1-23 —	Reserved.
24-31 TEMP	Highest temperature recorded in degrees Celcius by any enabled monitored site. Valid when V=1. 0-110 °C Sensor range 111-255 °C Reserved

31.3.10 TMU monitor low temperature capture register (TMLTCR)

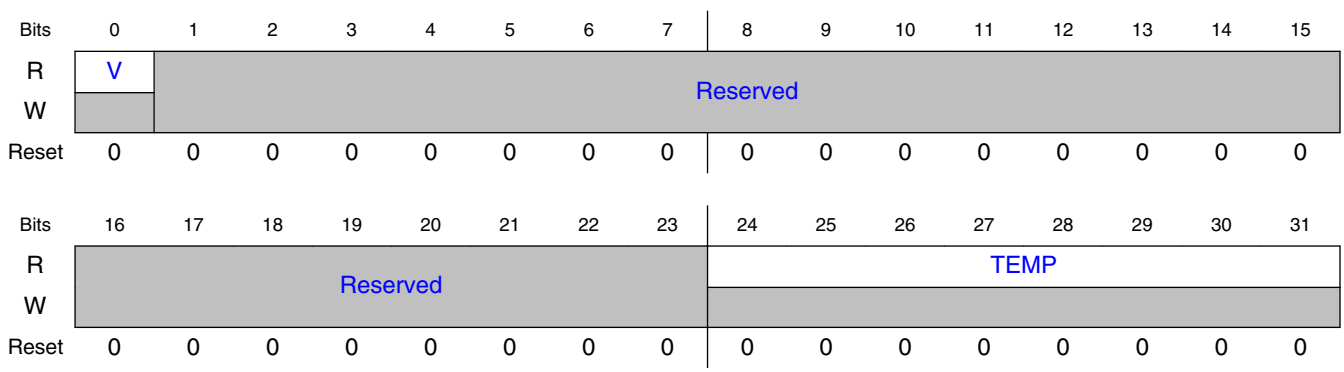
31.3.10.1 Offset

Register	Offset
TMLTCR	44h

31.3.10.2 Function

This TMU monitor register captures and record the lowest temperature reached for any one enabled monitored site within temperature sensor range.

31.3.10.3 Diagram



31.3.10.4 Fields

Field	Function
0 V	Valid reading. 0 Temperature reading is not valid due to no measured temperature within the sensor range of 0-110 °C for an enabled monitored site. 1 Temperature reading is valid. (Re-)enabling the TMU will automatically clear this bit and start a new search.
1-23 —	Reserved.
24-31 TEMP	Lowest temperature recorded in degrees Celcius by any enabled monitored site. Valid when V=1. 0-110 °C Sensor range 111-255 °C Reserved

31.3.11 TMU monitor high temperature immediate threshold register (TMHTITR)

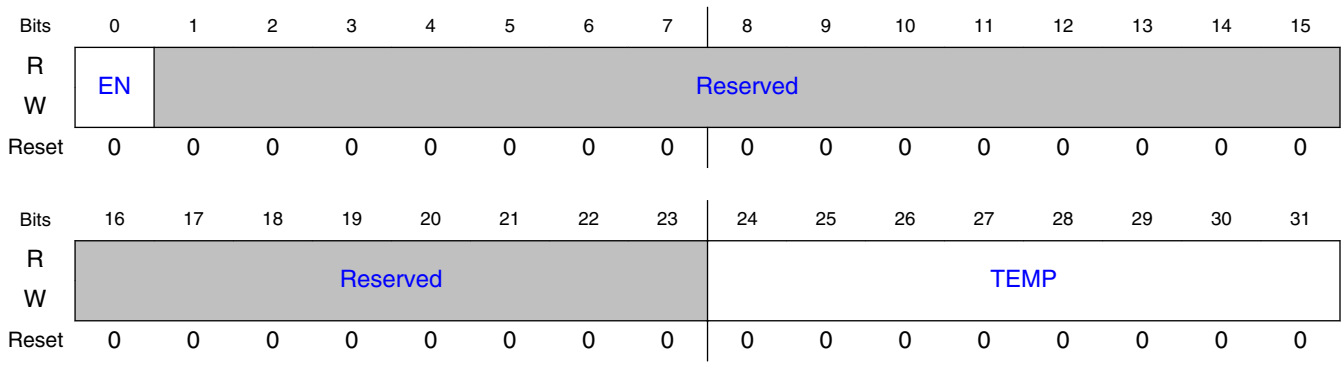
31.3.11.1 Offset

Register	Offset
TMHTITR	50h

31.3.11.2 Function

This TMU monitor register determines the high current temperature threshold for generating the TIDR[ITTE] event.

31.3.11.3 Diagram



31.3.11.4 Fields

Field	Function
0 EN	Enable threshold. 0 Disabled. 1 Threshold enabled.
1-23 —	Reserved.
24-31 TEMP	High temperature immediate threshold value. Determines the current upper temperature threshold, for any enabled monitored site, that if exceeded will cause TIDR[ITTE] to be set when EN=1. 0-110 °C Sensor range 111-255 °C Reserved

31.3.12 TMU monitor high temperature average threshold register (TMHTATR)

31.3.12.1 Offset

Register	Offset
TMHTATR	54h

31.3.12.2 Function

This TMU monitor register determines the high average temperature threshold for generating the TIDR[ATTE] event. The low-pass filter setting, TMR[ALPF], determines the function for calculating average temperature.

31.3.12.3 Diagram

Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	EN	Reserved														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

31.3.12.4 Fields

Field	Function
0 EN	Enable threshold. 0 Disabled. 1 Threshold enabled.
1-23 —	Reserved.
24-31 TEMP	High temperature average threshold value. Determines the average upper temperature threshold, for any enabled monitor site, that if exceeded will cause TIDR[ATTE] to be set when EN=1. 0-110 °C Sensor range 111-255 °C Reserved

31.3.13 TMU monitor high temperature average critical threshold register (TMHTACTR)

31.3.13.1 Offset

Register	Offset
TMHTACTR	58h

31.3.13.2 Function

This TMU monitor register determines the high average critical temperature threshold for generating the TIDR[ATCTE] event. The low-pass filter setting, TMR[ALPF], determines the function for calculating average temperature.

31.3.13.3 Diagram

Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	EN	Reserved														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

31.3.13.4 Fields

Field	Function
0 EN	Enable threshold. 0 Disabled. 1 Threshold enabled.
1-23 —	Reserved.
24-31 TEMP	High temperature average critical threshold value. Determines the average upper critical temperature threshold, for any enabled monitored site, that if exceeded will cause TIDR[ATCTE] to be set when EN=1. 0-110 °C Sensor range 111-255 °C Reserved

31.3.14 TMU temperature configuration register (TTCFGR)

31.3.14.1 Offset

Register	Offset
TTCFGR	80h

31.3.14.2 Function

The TMU temperature configuration register, in conjunction with the sensor configuration register, is used to initialize the internal sensor translation table used during monitoring. This register pair defines indirect access to the table. See the section Initialization Information.

31.3.14.3 Diagram

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

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

31.3.14.4 Fields

Field	Function
0-31 DATA	Sensor data.

31.3.15 TMU sensor configuration register (TSCFGR)

31.3.15.1 Offset

Register	Offset
TSCFGR	84h

31.3.15.2 Function

The TMU sensor configuration register, in conjunction with the temperature configuration register, is used to initialize the internal sensor translation table used during monitoring. This register pair defines indirect access to the table. Reading this register will return the data from the translation table as defined by TTCFGR. See the section Initialization Information.

31.3.15.3 Diagram

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

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

31.3.15.4 Fields

Field	Function
0-31 DATA	Sensor data.

31.3.16 TMU report immediate temperature site register 0 (TRITSR0)

31.3.16.1 Offset

Register	Offset
TRITSR0	100h

31.3.16.2 Function

This TMU report register returns the last measured temperature at site. The site must be part of the list of enabled monitored sites as defined by TMR[MSITE].

31.3.16.3 Diagram

Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	V	Reserved														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

31.3.16.4 Fields

Field	Function
0 V	Valid measured temperature. 0 Not valid. Temperature out of sensor range or first measurement still pending. 1 Valid.
1-23 —	Reserved.
24-31	Last temperature reading at site when V=1.

Field	Function
TEMP	

31.3.17 TMU report average temperature site register 0 (TRATSR0)

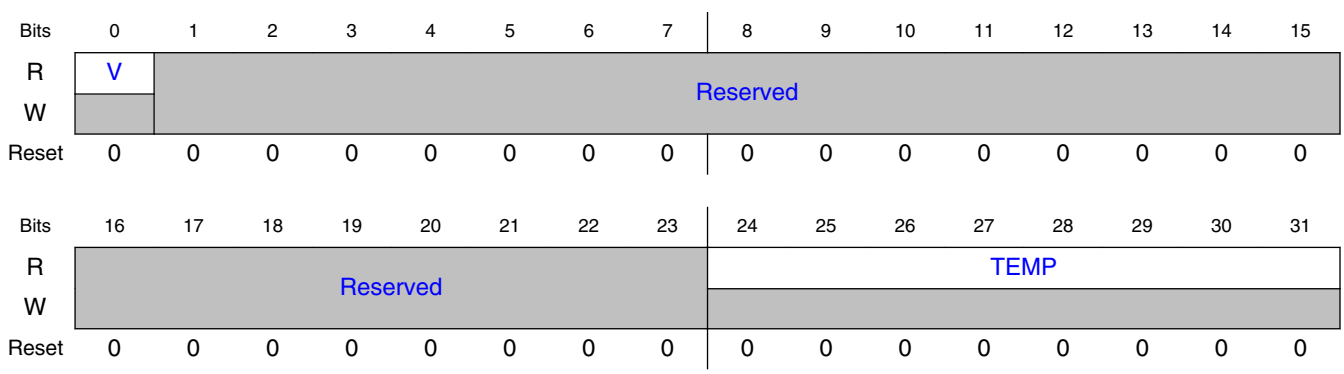
31.3.17.1 Offset

Register	Offset
TRATSR0	104h

31.3.17.2 Function

This TMU report register returns the average measured temperature at site. The site must be part of the list of enabled monitored sites as defined by TMR[MSITE].

31.3.17.3 Diagram



31.3.17.4 Fields

Field	Function
0	Valid measured temperature.
V	0 Not valid. Temperature out of sensor range or first measurement still pending.

Table continues on the next page...

Field	Function
	1 Valid.
1-23 —	Reserved.
24-31 TEMP	Average temperature reading at site when V=1.

31.3.18 TMU temperature range 0 control register (TTR0CR)

31.3.18.1 Offset

Register	Offset
TTR0CR	F10h

31.3.18.2 Function

The TMU temperature range control registers allow for defining the four temperature ranges that each sensor covers. A starting temperature range and number of calibration points, up to 16, is given. The temperature ranges may overlap to remove possible sample error, when searching for an accurate sensor reading. The default temperature calibration points are shown below.

Table 31-5. Default Temperature Configuration Points

Configuration Range	Starting Temperature (Celsius)	Temperature Configuration Points (Celsius)	Number of Points
0	0	0, 4, 8, 12, 16, 20, 24, 28, 32, 36, 40, 44	12
1	42	42, 46, 50, 54, 58, 62, 66, 70, 74, 78	10
2	76	76, 80, 84, 88, 92, 96, 100	7
3	98	98, 102, 106, 110	4

31.3.18.3 Diagram

Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Reserved												CAL_PTR			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1

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

31.3.18.4 Fields

Field	Function
0-11 —	Reserved.
12-15 CAL_PTR	Temperature configuration points. 0000 Reserved 0001 2 points 0010 3 points ... 1111 16 points
16-23 —	Reserved.
24-31 TEMP	Starting temperature in Celsius for range. In this temperature range each point is an integer multiple of 4C increments from this starting point. Therefore, any point temp in the range is = TTRnCR[TEMP]+ i * 4C , 0 ≤ i < TTRnCR[CAL_PTR].

31.3.19 TMU temperature range 1 control register (TTR1CR)

31.3.19.1 Offset

Register	Offset
TTR1CR	F14h

31.3.19.2 Function

The TMU temperature range control registers allow for defining the four temperature ranges that each sensor covers. A starting temperature range and number of calibration points, up to 16, is given. The temperature ranges may overlap to remove possible sample error, when searching for an accurate sensor reading. The default temperature calibration points are shown below.

Table 31-6. Default Temperature Configuration Points

Configuration Range	Starting Temperature (Celsius)	Temperature Configuration Points (Celsius)	Number of Points
0	0	0, 4, 8, 12, 16, 20, 24, 28, 32, 36, 40, 44	12
1	42	42, 46, 50, 54, 58, 62, 66, 70, 74, 78	10
2	76	76, 80, 84, 88, 92, 96, 100	7
3	98	98, 102, 106, 110	4

31.3.19.3 Diagram

Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Reserved												CAL_PTR			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0

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

31.3.19.4 Fields

Field	Function
0-11 —	Reserved.
12-15 CAL_PTR	Temperature configuration points. 0000 Reserved 0001 2 points

Table continues on the next page...

Field	Function
	0010 3 points ... 1111 16 points
16-23 —	Reserved.
24-31 TEMP	Starting temperature in Celsius for range. In this temperature range each point is an integer multiple of 4C increments from this starting point. Therefore, any point temp in the range is = TTRnCR[TEMP]+ i * 4C , 0 ≤ i < TTRnCR[CAL_PTR].

31.3.20 TMU temperature range 2 control register (TTR2CR)

31.3.20.1 Offset

Register	Offset
TTR2CR	F18h

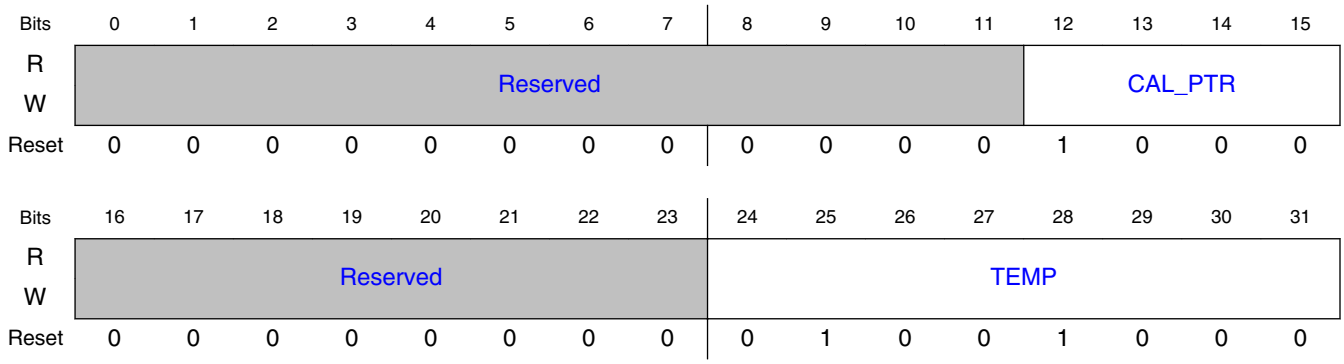
31.3.20.2 Function

The TMU temperature range control registers allow for defining the four temperature ranges that each sensor covers. A starting temperature range and number of calibration points, up to 16, is given. The temperature ranges may overlap to remove possible sample error, when searching for an accurate sensor reading. The default temperature calibration points are shown below.

Table 31-7. Default Temperature Configuration Points

Configuration Range	Starting Temperature (Celsius)	Temperature Configuration Points (Celsius)	Number of Points
0	0	0, 4, 8, 12, 16, 20, 24, 28, 32, 36, 40, 44	12
1	42	42, 46, 50, 54, 58, 62, 66, 70, 74, 78	10
2	76	76, 80, 84, 88, 92, 96, 100	7
3	98	98, 102, 106, 110	4

31.3.20.3 Diagram



31.3.20.4 Fields

Field	Function
0-11 —	Reserved.
12-15 CAL_PTR	Temperature configuration points. 0000 Reserved 0001 2 points 0010 3 points ... 1111 16 points
16-23 —	Reserved.
24-31 TEMP	Starting temperature in Celsius for range. In this temperature range each point is an integer multiple of 4C increments from this starting point. Therefore, any point temp in the range is = TTRnCR[TEMP]+ i * 4C , 0 ≤ i < TTRnCR[CAL_PTR].

31.3.21 TMU temperature range 3 control register (TTR3CR)

31.3.21.1 Offset

Register	Offset
TTR3CR	F1Ch

31.3.21.2 Function

The TMU temperature range control registers allow for defining the four temperature ranges that each sensor covers. A starting temperature range and number of calibration points, up to 16, is given. The temperature ranges may overlap to remove possible sample error, when searching for an accurate sensor reading. The default temperature calibration points are shown below.

Table 31-8. Default Temperature Configuration Points

Configuration Range	Starting Temperature (Celsius)	Temperature Configuration Points (Celsius)	Number of Points
0	0	0, 4, 8, 12, 16, 20, 24, 28, 32, 36, 40, 44	12
1	42	42, 46, 50, 54, 58, 62, 66, 70, 74, 78	10
2	76	76, 80, 84, 88, 92, 96, 100	7
3	98	98, 102, 106, 110	4

31.3.21.3 Diagram

Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Reserved												CAL_PTR			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

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

31.3.21.4 Fields

Field	Function
0-11 —	Reserved.
12-15 CAL_PTR	Temperature configuration points. 0000 Reserved 0001 2 points

Table continues on the next page...

Field	Function
	0010 3 points ... 1111 16 points
16-23 —	Reserved.
24-31 TEMP	Starting temperature in Celsius for range. In this temperature range each point is an integer multiple of 4C increments from this starting point. Therefore, any point temp in the range is = TTRnCR[TEMP]+ i * 4C , 0 ≤ i < TTRnCR[CAL_PTR].

31.4 Functional Description

The following sections describe the functionality of the TMU in details.

31.4.1 Monitoring

Monitoring is the process of reading enabled temperature sensor sites on-chip at regular intervals and taking appropriate actions, such as alarming the user when the temperature exceeds a programmed temperature threshold.

During monitoring, all enabled temperature sensor sites are periodically measured for temperature starting with site 0 and ending with site 0. The interval at which the sensors are read is set in TMTMIR and should reflect the maximum interval time required to accurately capture temperature changes. If the measurement interval has not expired after the last active site has been read, the sensor logic enters low-power mode. If the interval has expired when the last active site is read, the measurement interval exceeded bit, TSR[MIE], is set and the next active site is read immediately. If the interval has been exceeded, user may opt to reduce number of sites monitored or increase the interval, if possible.

For each site the current and average temperature is logged. The average temperature is calculated based on the low-pass filter function in the mode register. If any of the set temperature threshold registers are exceeded, the corresponding interrupt detect bit is set in TIDR. Interrupts are enabled through the interrupt enable register, TIER.

Process for enabling monitoring mode:

1. Clear the interrupt detect register, TIDR.
2. Clear interrupt site capture register (TISCR) and interrupt critical site capture register (TICSCR).
3. Enable interrupt handling by setting the appropriate bits in TIER.

4. Set the temperature threshold registers TMHTITR, TMHTATR and TMHTACTR.
5. Set the monitoring interval register, TMTMIR.
6. Enable monitor mode by setting TMR[ME]=1. Sites to monitor are controlled by setting TMR[MSITE]. There should be at least one active site enabled. Set other mode control bits as needed.
7. If the monitoring interval is too short as indicated by TSR[MIE], the interval may need to be increased or number of sites reduced.

31.4.2 Reporting

The TMU can directly report the current and average temperature for a particular temperature sensor site during monitoring mode by reading one of the report registers per site. The report uses the last measurement done by the monitoring process and requires a site to be actively monitored for accurate reading. Reading a site which has last measured an invalid temperature outside the sensor range of 0-110 degrees Celsius, will have the valid bit cleared.

If monitoring is disabled, the last temperature measurement remains for the site(s) previously monitored and can still be read using the report registers knowing that the temperature reported is no longer accurate. This method can be used to capture the temperature at multiple sites in time, but does not allow for continuous monitoring.

Chapter 32
Universal Serial Bus Interface 2.0

32.1 Overview

This chapter describes the universal serial bus (USB) interface of the chip.

The USB interface implements many industry standards. However, it is beyond the scope of this document to document the intricacies of these standards. Instead, it is left to the reader to refer to the governing specifications.

The following documents are available from the USB Implementers Forum web page at *http://www.usb.org/developers/docs/*.

- *Universal Serial Bus Revision 2.0 Specification*

The following documents are available from the Intel USB Specifications web page at *http://www.intel.com/technology/usb/spec.htm*.

- *Enhanced Host Controller Interface (EHCI) Specification for Universal Serial Bus, Revision 1.0*

The following documents are available from the ULPI web page at *http://www.ulpi.org/*

- *UTMI+ Specification, Revision 1.0*
- *UTMI Low Pin-Count Interface (ULPI) Specification, Revision 1.0*

The device implements a dual-role (DR) USB module. This module may be connected to an external port. Collectively the module and external port are called the USB interface.

The USB DR module is a USB 2.0-compliant serial interface engine for implementing a USB interface. The registers and data structures are based on the *Enhanced Host Controller Interface Specification for Universal Serial Bus* (EHCI) from Intel Corporation. Each DR module is either a device or host controller.

The DR module supports the required signaling for transceivers (PHYs).

The USB interface is shown in the figure below.

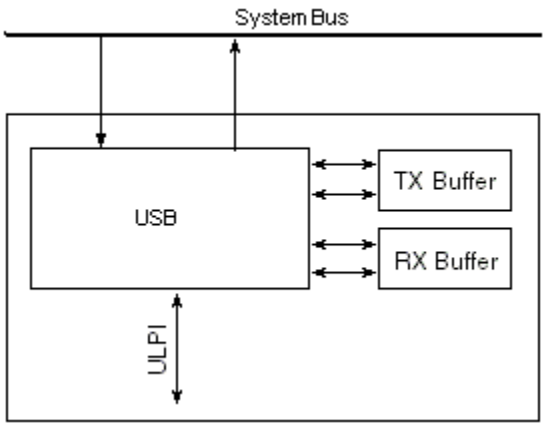


Figure 32-1. USB interface block diagram

NOTE

- The USB 2.0 operates only in host mode.
- There should not be any unaligned accesses from USB 2.0.

32.1.1 USB features summary

The USBDR module includes the following features:

- Complies with *USB Specification Rev 2.0*
- Supports operation as a standalone USB host controller
  - Supports enhanced host controller interface (EHCI)
- Supports high-speed (480 Mbps), full-speed (12 Mbps), and low-speed (1.5 Mbps) operation. Low speed is only supported in host mode
- Supports external PHY with ULPI (UTMI + low-pin interface)
- Supports operation as a standalone USB device
  - Supports one upstream facing port
  - Supports six bidirectional USB Endpoints
- Host and device support

32.1.2 Modes of operation

The USB DR module operates in two modes:

- Host (HS/FS/LS)
- Device (HS/FS/LS)

NOTE

Only high-speed and full-speed operations are supported in device mode.

32.2 USB external signals

This section contains detailed description of all the USB controller signals.

32.2.1 ULPI

The ULPI (UTMI+ low pin interface) is a reduced pin-count (12 signals) extension of the UTMI+ specification.

Pin count is reduced by converting relatively static signals to register bits, and providing a bidirectional, generic data bus that carries USB and register data. This interface minimizes pin count requirements for external PHYs. The table below describes the signals for the ULPI interface.

Table 32-1. ULPI signal descriptions

Signal	I/O	Description	
USB*n*_DIR	I	Direction. USB*n*_DIR controls the direction of the data bus. When the PHY has data to transfer to USB port, it drives USB*n*_DIR high to take ownership of the bus. When the PHY has no data to transfer it drives USB*n*_DIR low and monitors the bus for link activity. The PHY pulls USB*n*_DIR high whenever the interface cannot accept data from the link.	
		State Meaning	Asserted-PHY has data to transfer to the link. Negated-PHY has no data to transfer.
		Timing	Synchronous to PHY_CLK.
USB*n*_NXT	I	Next data. The PHY asserts USB*n*_NXT to throttle the data. When USB port is sending data to the PHY, USB*n*_NXT indicates when the current byte has been accepted by the PHY. The USB port places the next byte on the data bus in the following clock cycle. When the PHY is sending data to USB port, USB*n*_NXT indicates when a new byte is available for USB port to consume.	
		State Meaning	Asserted-PHY is ready to transfer byte. Negated-PHY is not ready.
		Timing	Synchronous to PHY_CLK.
USB*n*_STP	O	Stop. USB*n*_STP indicates the end of a transfer on the bus.	
		State Meaning	Asserted-USB asserts this signal for 1 clock cycle to stop the data stream currently on the bus. If USB port is sending data to the PHY, USB*n*_STP indicates the last byte of data was previously on the bus. If the PHY is sending data to USB port, USB*n*_STP forces the PHY to end its transfer, negate USB*n*_DIR and relinquish control of the data bus to the USB port. Negated-Indicates normal operation.

Table continues on the next page...

Table 32-1. ULPI signal descriptions (continued)

Signal	I/O	Description	
		Timing	Synchronous to PHY_CLK.
USB*n*_D[7:0]	I/O	Data bit *n*. USB*n*_D*n* is bit *n* of the 8-bit (USB*n*_D7-USB*n*_D0), uni-directional data bus used to carry USB, register, and interrupt data between the PHY and the USB controller.	
		State Meaning	Asserted-Data bit *n* is 1. Negated-Data bit *n* is 0.
		Timing	Synchronous to PHY_CLK.

NOTE

USB2.0 PWRFAULT signal is tied to 0 by default. If required, this signal can be shared from USB 3.0.

32.2.2 PHY clocks

The USB*n*_CLK input is the clock from the external ULPI PHY to the controller .

The clock is 60 MHz. Detailed clock specifications are given in the chip data sheet document.

NOTE

A write to registers in the USB controller memory map may cause the system to hang if PORTSC[PHCD]=0 and there is no clock from the ULPI PHY.

32.3 USB register descriptions

This section provides the memory map and detailed descriptions of all USB interface registers.

The following sections provide details about the registers in the USB memory map.

NOTE

Memory may be viewed from either a big-endian or little-endian byte ordering perspective depending on the processor configuration. In big-endian mode, the most-significant byte of word 0 is located at address 0 and the least-significant byte of word 0 is located at address 3. In little-endian mode, the least-significant byte of word 0 is located at address 0 and the most-

significant byte of word 0 is located at address 3. Within registers, bits are numbered within a word starting with bit 31 as the most-significant bit. By convention, USB registers use little-endian byte ordering. In the USB DR module, these are the registers from offsets 0x00 to 0x1FF. The registers associated with the internal system interface (0x400 and above) use big-endian byte ordering.

32.3.1 USB Memory map

USB base address: 860_0000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Identification register (USB_ID)	32	RO	See description.
80h	General purpose timer load n (GPTIMER0LD)	32	RW	0000_0000h
84h	General purpose timer control n (GPTIMER0CTRL)	32	RW	0000_0000h
88h	General purpose timer load n (GPTIMER1LD)	32	RW	0000_0000h
8Ch	General purpose timer control n (GPTIMER1CTRL)	32	RW	0000_0000h
100h	Capability Register Length (CAPLENGTH)	8	RO	40h
102h	Host Controller Interface Version Number (HCIVERSION)	16	RO	0100h
104h	Host Controller Structural Parameters (HCSPARAMS)	32	RO	0001_0011h
108h	Host Controller Capability Parameters (HCCPARAMS)	32	RO	0000_0006h
120h	Device Controller Interface Version Number (DCIVERSION)	16	RO	0001h
124h	Device Controller Capability Parameters (DCCPARAMS)	32	RO	0000_0186h
140h	USB Command (USBCMD)	32	RW	See description.
144h	USB Status (USBSTS)	32	W1C	0000_0000h
148h	USB Interrupt Enable (USBINTR)	32	RW	0000_0000h
14Ch	USB Frame Index (FRINDEX)	32	RW	0000_0000h
154h	USB Device Address [device mode] (DEVICEADDR)	32	RW	0000_0000h
154h	Periodic Frame List Base Address [host mode] (PERIODICLIST BASE)	32	RW	0000_0000h
158h	Next Asynchronous List Addr [host mode] (ASYNCLISTADDR)	32	RW	0000_0000h
158h	Address at Endpoint List [device mode] (ENDPOINTLISTADDR)	32	RW	0000_0000h
160h	Master Interface Data Burst Size (BURSTSIZE)	32	RW	0000_1010h
164h	Transmit FIFO Tuning Controls (TXFILLTUNING)	32	RW	See description.
170h	ULPI Register Access (ULPI_VIEWPORT)	32	RW	0800_0000h
178h	Endpoint NAK Indicaton Register (ENDPTNAK)	32	RW	0000_0000h
17Ch	Endpoint NAK Indication Enable Register (ENDPTNAKEN)	32	RW	0000_0000h

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
180h	Configured Flag Register (CONFIGFLAG)	32	RO	0000_0001h
184h	Port Status/Control (PORTSC)	32	RW	See description.
1A8h	USB Device Mode (USBMODE)	32	RW	0000_5000h
1ACh	Endpoint Setup Status (ENDPTSETUPSTAT)	32	W1C	0000_0000h
1B0h	Endpoint Initialization (ENDPOINTPRIME)	32	RW	0000_0000h
1B4h	Endpoint Flush (ENDPTFLUSH)	32	RW	0000_0000h
1B8h	Endpoint Status (ENDPTSTATUS)	32	RO	0000_0000h
1BCh	Endpoint Complete (ENDPTCOMPLETE)	32	W1C	0000_0000h
1C0h	Endpoint Control 0 (ENDPTCTRL0)	32	RW	0080_0080h
1C4h - 1D4h	Endpoint Control n (ENDPTCTRL1 - ENDPTCTRL5)	32	RW	0000_0000h
400h - 404h	Snoop n (SNOOP1 - SNOOP2)	32	RW	0000_0000h
408h	Age Count Threshold (AGE_CNT_THRESH)	32	RW	0000_0000h
40Ch	Priority Control (PRI_CTRL)	32	RW	0000_0000h
410h	System Interface Control (SI_CTRL)	32	RW	0000_0000h
500h	Control (CONTROL)	32	RW	0000_0000h

32.3.2 Identification register (USB_ID)

32.3.2.1 Offset

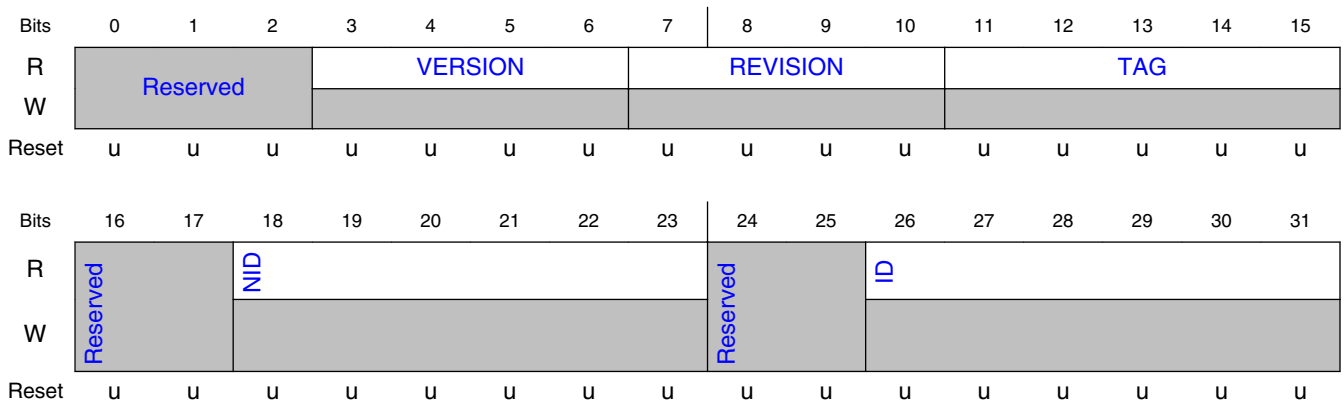
Register	Offset
USB_ID	0h

32.3.2.2 Function

The identification register is used to declare the slave interface presence and include table of the hardware configuration parameters. It is not defined by the EHCI specification.

The ID register provides a simple way to determine if the USB DR module is provided in the system. The ID register identifies the USB DR and its revision (E481_FA05).

32.3.2.3 Diagram



32.3.2.4 Fields

Field	Function
0-2 —	- Reserved
3-6 VERSION	VERSION Identifies the version of the module.
7-10 REVISION	REVISION Identifies the revision number of the module.
11-15 TAG	TAG Identifies the tag of the USB core.
16-17 —	- Reserved
18-23 NID	NID Ones complement version of ID bit.
24-25 —	- Reserved
26-31 ID	ID Configuration number. This number is set to 0x05 and indicates that the peripheral is the USB_DR.

32.3.3 General purpose timer load n (GPTIMER0LD - GPTIMER1 LD)

32.3.3.1 Offset

For a = 0 to 1:

Register	Offset
GPTIMERaLD	80h + (a × 8h)

32.3.3.2 Function

Host/device controller drivers can measure time related activities using these timer registers. These registers are not part of standard EHCI controller.

32.3.3.3 Diagram

Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Reserved								GPTLD							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

32.3.3.4 Fields

Field	Function
0-7 —	- Reserved
8-31 GPTLD	GPTLD **General Purpose Timer Load Value** . This field is the value loaded into the GPTCNT countdown timer on a reset action. This value represents the time in microseconds minus one for the timer duration. Note: Maximum value is 0xFFFFFF or 16.777215 seconds.

32.3.4 General purpose timer control n (GPTIMER0CTRL - GPTIMER1CTRL)

32.3.4.1 Offset

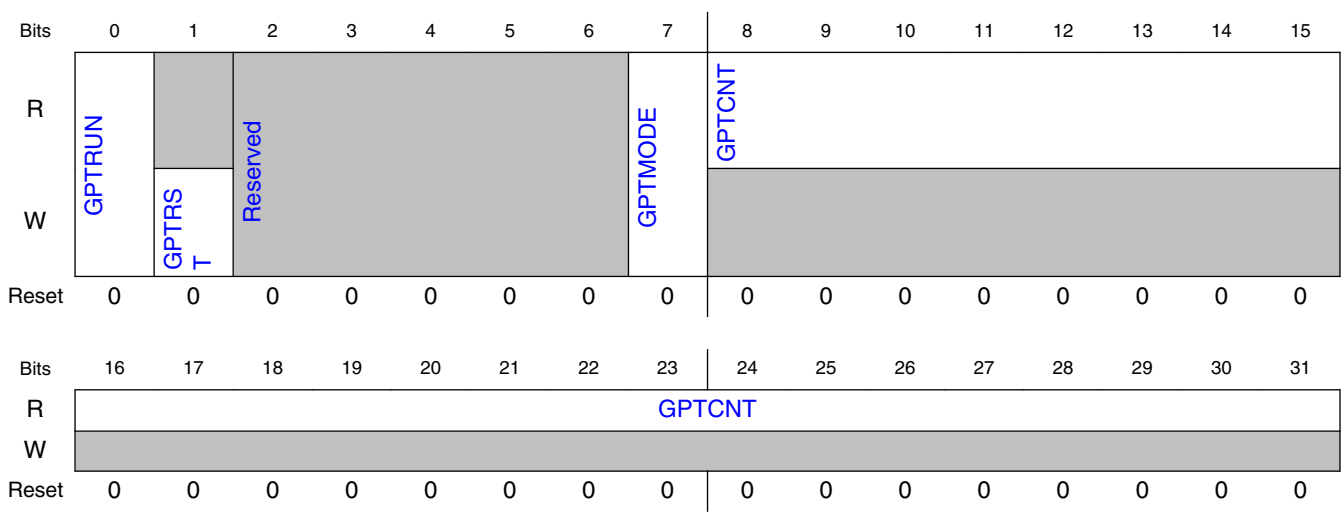
For a = 0 to 1:

Register	Offset
GPTIMERaCTRL	84h + (a × 8h)

32.3.4.2 Function

Host/device controller drivers can measure time related activities using these timer registers. These registers are not part of standard EHCI controller.

32.3.4.3 Diagram



32.3.4.4 Fields

Field	Function
0 GPTRUN	GPTRUN **General Purpose Timer Run** . This bit enables the general purpose timer to run. Setting or clearing this bit does not have any effect on GPTCNT value.

Table continues on the next page...

Field	Function
	0b - Timer Stop 1b - Timer Run
1 GPTRST	GPTRST **General Purpose Timer Reset** . Writing 1 to this bit reloads GPTCNT with value in GPTLD. 0b - No Action 1b - Reload Counter
2-6 —	- Reserved
7 GPTMODE	GPTMODE **General Purpose Timer Mode** . This bit selects between single timer countdown and a looped countdown. In one-shot mode, the timer counts down to zero, generates an interrupt and stops until software resets the counter. In repeat mode, the timer counts down to zero, generates an interrupt and automatically reloads the counter to begin again. 0b - One-shot 1b - Repeat
8-31 GPTCNT	GPTCNT **General Purpose Timer Counter** . Contains the value of runnig counter.

32.3.5 Capability Register Length (CAPLENGTH)

32.3.5.1 Offset

Register	Offset
CAPLENGTH	100h

32.3.5.2 Function

The capability registers specify the software limits, restrictions, and capabilities of the host/device controller implementation. Most of these registers are defined by the EHCI specification. Registers that are not defined by the EHCI specification are noted in their descriptions.

CAPLENGTH is used as an offset to add to the register base address to find the beginning of the operational register space, that is, the location of the USBCMD register.

32.3.5.3 Diagram

Bits	0	1	2	3	4	5	6	7
R	CAPLENGTH							
W								
Reset	0	1	0	0	0	0	0	0

32.3.5.4 Fields

Field	Function
0-7 CAPLENGTH	CAPLENGTH Capability registers length. It indicates which offset to add to the register base address at the beginning of the operational register. Value is 0x40.

32.3.6 Host Controller Interface Version Number (HCIVERSION)

32.3.6.1 Offset

Register	Offset
HCIVERSION	102h

32.3.6.2 Function

HCIVERSION contains a BCD encoding of the EHCI revision number supported by this host controller. The most-significant byte of the register represents major revision and the least-significant byte is minor revision.

32.3.6.3 Diagram

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

32.3.6.4 Fields

Field	Function
0-15 HCIVERSION	HCIVERSION EHCI revision number. Value is 0x0100 indicating version 1.0.

32.3.7 Host Controller Structural Parameters (HCSPARAMS)

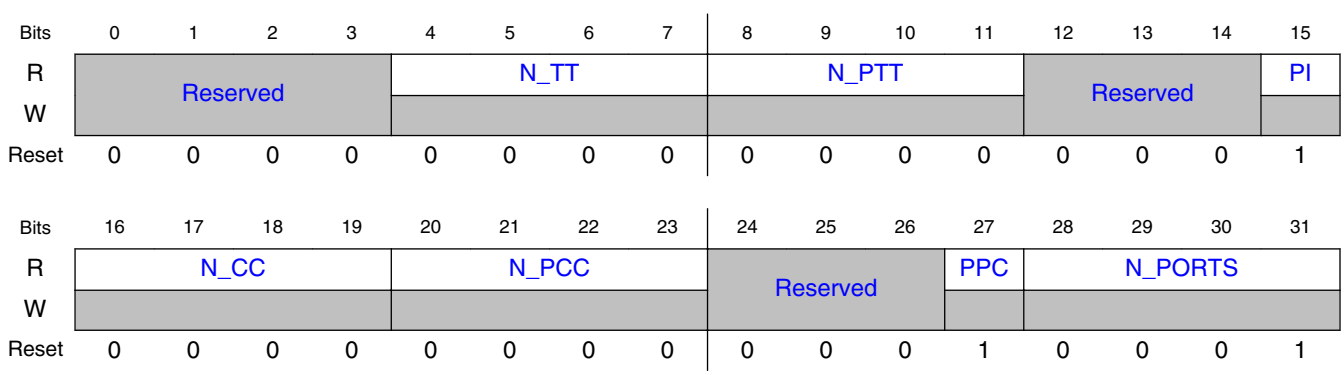
32.3.7.1 Offset

Register	Offset
HCSPARAMS	104h

32.3.7.2 Function

HCSPARAMS contains structural parameters such as the number of downstream ports.

32.3.7.3 Diagram



32.3.7.4 Fields

Field	Function
0-3 —	- Reserved, should be cleared.
4-7 N_TT	N_TT Number of transaction translators. This is a non-EHCI field. This field indicates the number of embedded transaction translators associated the module. Always 1.
8-11 N_PTT	N_PTT Ports per transaction translator. This is a non-EHCI field. The number of ports assigned to each transaction translator. This is equal to N_PORTS.
12-14 —	- Reserved, should be cleared.
15 PI	PI Port indicators. Indicates whether the ports support port indicator control. Always 1. 1b - The port status and control registers include a R/W field for controlling the state of the port indicator.
16-19 N_CC	N_CC Number of companion controllers associated with the DR controller. Always 0.
20-23 N_PCC	N_PCC Number ports per CC. This field indicates the number of ports supported per internal companion controller. Always 0.
24-26 —	- Reserved, should be cleared.
27 PPC	PPC Power port control. Indicates whether the host controller supports port power control. ULPI Mode: 0b - USBDR will write 0 for DRVVBUS bit of OTG Control register in PHY. 1b - USBDR will write 1 for DRVVBUS bit of OTG Control register in PHY. The OTG cntrol register is defined in ULPI specification.
28-31 N_PORTS	N_PORTS Number of ports. Number of physical downstream ports implemented for host applications. The value of this field determines how many port registers are addressable in the operational register. Always 1.

32.3.8 Host Controller Capability Parameters (HCCPARAMS)

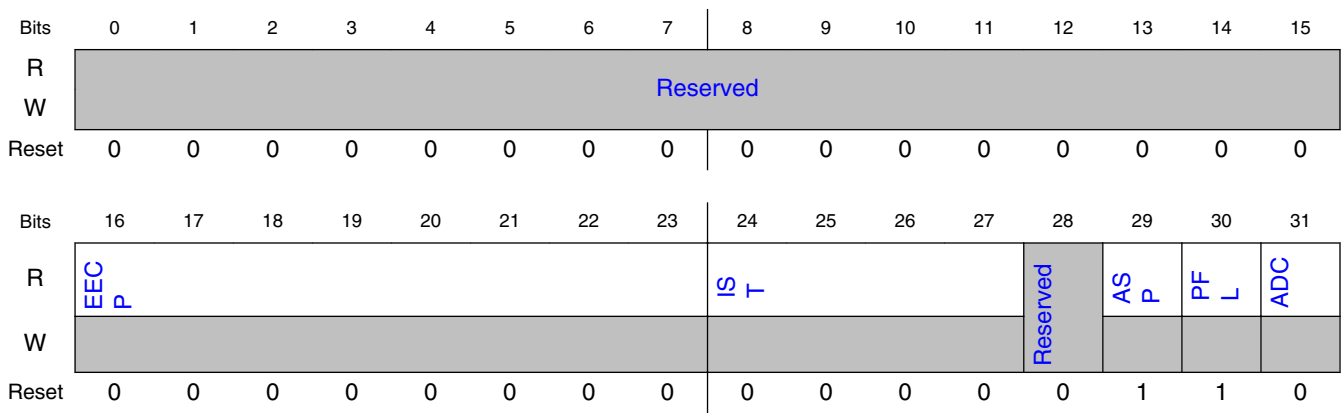
32.3.8.1 Offset

Register	Offset
HCCPARAMS	108h

32.3.8.2 Function

HCCPARAMS identifies multiple mode control (time-base bit functionality) addressing capability.

32.3.8.3 Diagram



32.3.8.4 Fields

Field	Function
0-15 —	- Reserved, should be cleared.
16-23 EECP	EECP EHCI extended capabilities pointer. Indicates the existence of a capabilities list. A value of 0x00 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 0x40 or greater if implemented to maintain the consistency of the PCI header defined for this class of device. This field is always 0.
24-27 IST	IST

Table continues on the next page...

Field	Function
	Isochronous scheduling threshold. 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 microframes a host controller can hold a set of isochronous data structures (one or more) before flushing the state. When bit 7 is a one, then host software assumes the host controller may cache an isochronous data structure for an entire frame. This field is always 0.
28 —	- Reserved, should be cleared.
29 ASP	ASP Asynchronous schedule park capability. Indicates whether the USB DR module supports the park feature for high-speed queue heads in the asynchronous schedule. The feature can be disabled or enabled and set to a specific level by using the *asynchronous schedule park mode enable* and *asynchronous schedule park mode count* fields in the USBCMD register. This field is always 1 (park feature supported).
30 PFL	PFL Programmable frame list flag. Indicates whether system software can specify and use a frame list length less that 1024 elements. Frame list size is configured via the USBCMD register frame list size field. The frame list must always be aligned on a 4-K page boundary. This requirement ensures that the frame list is always physically contiguous. This field is always 1.
31 ADC	ADC 64-bit addressing capability. Always 0; 64-bit addressing is not supported. 0b - Data structures use 32-bit address memory pointers

32.3.9 Device Controller Interface Version Number (DCIVERSION)

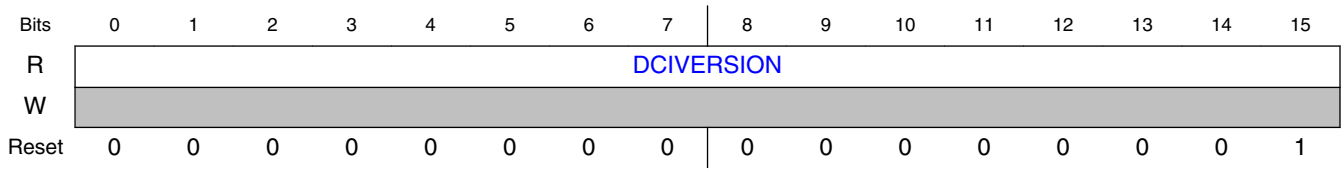
32.3.9.1 Offset

Register	Offset
DCIVERSION	120h

32.3.9.2 Function

This register is not defined in the EHCI specification. DCIVERSION is a two-byte register containing a BCD encoding of the device controller interface. The most-significant byte of the register represents major revision and the least-significant byte is minor revision.

32.3.9.3 Diagram



32.3.9.4 Fields

Field	Function
0-15 DCIVERSION	DCIVERSION Device interface revision number.

32.3.10 Device Controller Capability Parameters (DCCPARAMS)

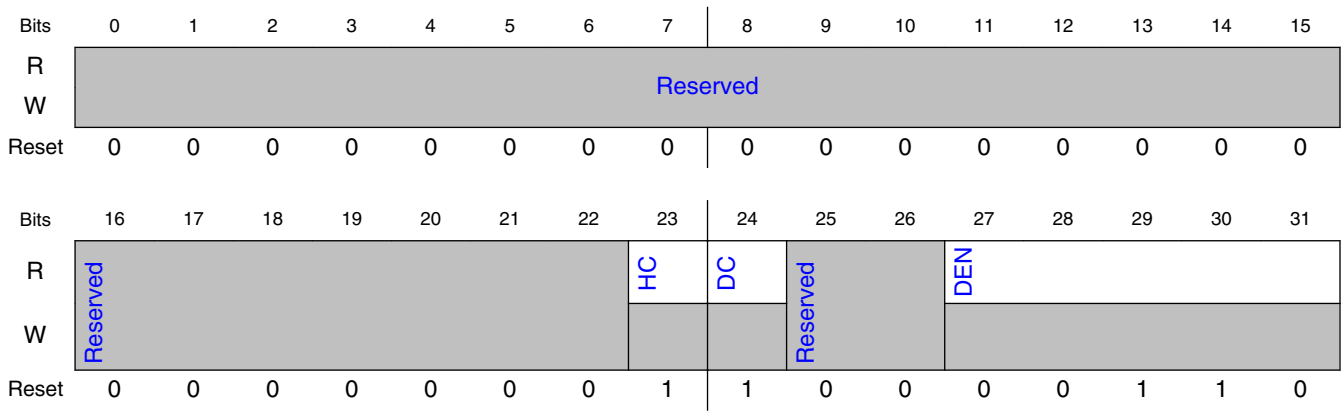
32.3.10.1 Offset

Register	Offset
DCCPARAMS	124h

32.3.10.2 Function

This register is not defined in the EHCI specification. This register describes the overall host/device capability of the DR module.

32.3.10.3 Diagram



32.3.10.4 Fields

Field	Function
0-22 —	- Reserved, should be cleared.
23 HC	HC Host capable. Always 1, indicating the USB DR controller can operate as an EHCI compatible USB 2.0 host
24 DC	DC Device capable. Always 1, indicating the USB DR controller can operate as an USB 2.0 device. 0b - No device capability (host only). 1b - Device capability.
25-26 —	- Reserved, should be cleared.
27-31 DEN	DEN Device Endpoint number. Indicates the number of Endpoints built into the device controller. Always 0x6.

32.3.11 USB Command (USBCMD)

32.3.11.1 Offset

Register	Offset
USBCMD	140h

32.3.11.2 Function

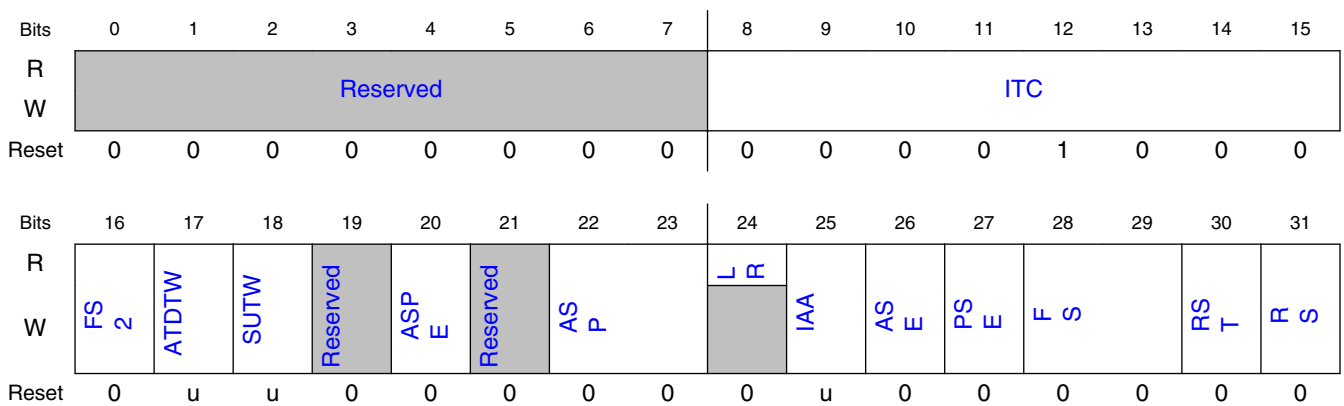
The operational registers are comprised of dynamic control or status registers that may be read-only, read/write, or read/write-1-to-clear. The following registers define the operational registers.

The module executes the command indicated in this register.

NOTE

Since the CONTROL register has a default setting for UTMI mode, the reset value of USBCMD register is read as 0x80002. When the CONTROL register is programmed to enter into the ULPI mode, the reset value of the USBCMD register becomes 0x80000.

32.3.11.3 Diagram



32.3.11.4 Fields

Field	Function
0-7 —	- Reserved, should be cleared.
8-15 ITC	ITC Interrupt threshold control. The system software uses this field to set the maximum rate at which the USB DR module issues interrupts. ITC contains the maximum interrupt interval measured in microframes. Valid values are shown below.

Table continues on the next page...

Field	Function
	00000000b - Immediate (no threshold) 00000001b - 1 microframe 00000010b - 2 microframes 00000100b - 4 microframes 00001000b - 8 microframes 00010000b - 16 microframes 00100000b - 32 microframes 01000000b - 40 microframes
16 FS2	FS2 See bits 3-2 below. This is a non-EHCI bit.
17 ATDTW	ATDTW Add dTD TripWire. This is a non-EHCI bit. Used as a semaphore when a dTD is added to an active (primed) Endpoint. This bit is set and cleared by software. This bit shall also be cleared by hardware when its state machine is in hazard region where adding a dTD to a primed Endpoint may go unrecognized.
18 SUTW	SUTW Setup tripwire. This is a non-EHCI bit. Used as a semaphore when the 8 bytes of setup data read extracted from a QH by the DCD. If the setup lockout mode is off (See USB Device Mode (USBMODE)) then there exists a hazard when new setup data arrives and the DCD is copying setup from the QH for a previous setup packet. This bit is set and cleared by software and will be cleared by hardware when a hazard exists.
19 —	- Reserved, should be cleared.
20 ASPE	ASPE Asynchronous schedule park mode enable. This bit defaults to a 1 and is R/W. Software uses this bit to enable or disable park mode. 0b - Disabled 1b - Enabled
21 —	- Reserved, should be cleared.
22-23 ASP	ASP Asynchronous schedule park mode count. It contains a count of the number of successive transactions the host controller is allowed to execute from a high-speed queue head on the Asynchronous schedule before continuing traversal of the Asynchronous schedule. Valid values are 0x1H to 0x3H. Software must not write a zero to this field when ASPE is set as this results in undefined behavior.
24 LR	LR Light host/device controller reset (OPTIONAL). Not implemented. Always 0.
25 IAA	IAA Interrupt on async advance doorbell. Used as a doorbell by software to tell the USB DR 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 controller has evicted all appropriate cached schedule states, it sets USBSTS[AAI]. If USBINTR[AAE] is set, the host controller asserts an interrupt at the next interrupt threshold. The controller clears this bit after it has set USBSTS[AAI]. Software should not set this bit when the asynchronous schedule is inactive. Doing so yields undefined results. This bit is only used in host mode. Setting this bit when the USB DR module is in device mode is selected results in undefined results.

Table continues on the next page...

Field	Function
26 ASE	ASE Asynchronous schedule enable. Controls whether the controller skips processing the asynchronous schedule. Only used in host mode. 0b - Do not process the asynchronous schedule 1b - Use the ASYNCLISTADDR register to access the asynchronous schedule.
27 PSE	PSE Periodic schedule enable. Controls whether the controller skips processing the periodic schedule. Only used in host mode. 0b - Do not process the periodic schedule. 1b - Use the PERIODICLISTBASE register to access the periodic schedule.
28-29 FS	FS Frame list size. Together with bit 15 these bits make the FS[2:0] field. This field is read/write only if programmable frame list flag in the HCCPARAMS registers is set to 1. This field specifies the size of the frame list that controls which bits in FRINDEX should be used for the frame list current index. Only used in host mode. Note that values below 256 elements are not defined in the EHCI specification. 00b - If FS2=0, 1024 elements (4096 bytes). If FS2=1, 64 elements (256 bytes). 01b - If FS2=0, 512 elements (2048 bytes). If FS2=1, 32 elements (128 bytes). 10b - If FS2=0, 256 elements (1024 bytes). If FS2=1, 16 elements (64 bytes). 11b - If FS2=0, 128 elements (512 bytes). If FS2=1, 8 elements (32 bytes).
30 RST	RST Controller reset. Software uses this bit to reset the controller. This bit is cleared by the controller when the reset process is complete. Software cannot terminate the reset process early by writing a zero to this register. • Host Mode: When software sets this bit, the host controller resets its internal pipelines, timers, counters, state machines, and so on 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 when USBSTS[HCH] is a zero. Attempting to reset an actively running host controller results in undefined behavior. • Device Mode: When software writes a one to this bit, the USB_DR resets its internal pipelines, timers, counters, state machines, and so on to their initial value. Any transaction currently in progress on USB is immediately terminated. Writing a one to this bit when the device is in the attached state is not recommended, since the effect on an attached host is undefined. In order to ensure that the device is not in an attached state before initiating a controller reset, all primed Endpoints should be flushed and the USBCMD[RS] bit should be set to 0. **NOTE:** • Once set, this bit is high for 63 clock cycles of system clock and then it transitions back to 0. Software lets the previous reset complete before setting this bit again. On setting this bit, the USBDR also resets the PHY. Software ensures that PHY also comes out reset before setting this bit again.
31 RS	RS Run/Stop. Host mode: • When this bit is set, the controller proceeds with the execution of the schedule. The controller continues execution as long as this bit is set. When this bit is set to 0, the host controller completes the current transaction on the USB and then halts. The USBSTS[HCH] bit indicates when the USB DR 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, USBSTS[HCH] is a one). Device mode:

Field	Function
	• Setting this bit causes the USB DR 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 is disabled upon transitioning into high-speed mode. Software should use this bit to prevent an attach event before the controller has been properly initialized. Clearing this bit causes a detach event. 0b - Stop 1b - Run

32.3.12 USB Status (USBSTS)

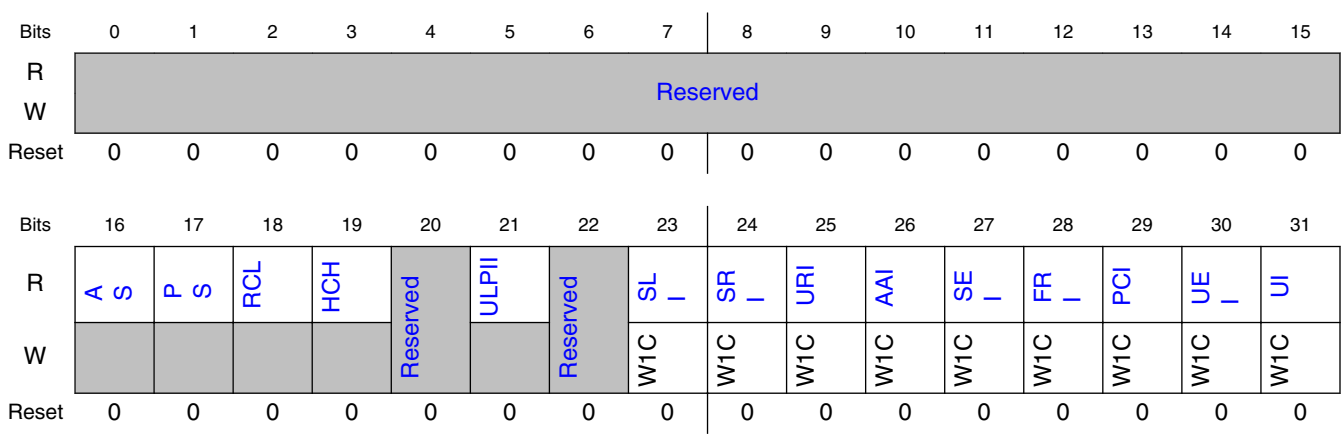
32.3.12.1 Offset

Register	Offset
USBSTS	144h

32.3.12.2 Function

The USB status register indicates various states of the USB DR module and any pending interrupts. This register does not indicate status resulting from a transaction on the serial bus. Software clears certain bits in this register by writing a 1 to them (indicated by a w1c in the bit's W cell).

32.3.12.3 Diagram



32.3.12.4 Fields

Field	Function
0-15	-
—	Reserved, should be cleared.
16	AS
AS	Asynchronous schedule status. Reports the current real status of the asynchronous schedule. The USB DR controller is not required to immediately disable or enable the asynchronous schedule when software transitions USBCMD[ASE]. When this bit and USBCMD[ASE] have the same value, the asynchronous schedule is either enabled (1) or disabled (0). Only used in host mode. 0b - Disabled 1b - Enabled
17	PS
PS	Periodic schedule status. Reports the current real status of the periodic schedule. The USB DR controller is not required to immediately disable or enable the periodic schedule when software transitions USBCMD[PSE]. When this bit and USBCMD[PSE] have the same value, the periodic schedule is either enabled (1) or disabled (0). Only used in host mode. 0b - Disabled 1b - Enabled
18	RCL
RCL	Reclamation. Used to detect an empty asynchronous schedule. Only used by the host mode. 0b - Non-empty asynchronous schedule 1b - Empty asynchronous schedule
19	HCH
HCH	HC halted. This bit is a zero whenever USBCMD[RS] is a one. The USB DR controller sets this bit to one after it has stopped executing because of USBCMD[RS] being cleared, either by software or by the host controller hardware (for example, internal error). Only used in host mode. 0b - Running 1b - Halted
20	-
—	Reserved, should be cleared.
21	ULPII
ULPII	ULPI interrupt. An event completion to the viewport register sets this bit. If the ULPI enables the USBINTR[ULPIE] to be set, the USB interrupt (UI) occurs.
22	-
—	Reserved, should be cleared.
23	SLI
SLI	DC Suspend. This is a non-EHCI bit. When a device controller enters a suspend state from an active state, this bit is set. This bit is only cleared by software by writing a 1 to it. Only used by the device controller. 0b - Active 1b - Suspended
24	SRI
SRI	Host mode:

Table continues on the next page...

Field	Function
	• This is a non-EHCI status bit. In host mode, this bit is set every 125 µs, provided the PHY clock is present and running (for example, the port is NOT suspended), and can be used by the host controller driver as a time base. Device mode: • SOF received. When the USB DR controller detects a Start Of (Micro)Frame, this bit is set. When a SOF is extremely late, the DR controller automatically sets this bit to indicate that an SOF was expected. Therefore, this bit is set roughly every 1- ms in device FS mode and every 125 µs in HS mode and is synchronized to the actual SOF that is received. Because the controller is initialized to FS before connect, this bit is set at an interval of 1 ms during the prelude to the connect and chirp. Software writes a 1 to this bit to clear it.
25 URI	URI USB reset received. This is a non-EHCI bit. When the USB DR controller detects a USB reset and enters the default state, this bit is set. Software can write a 1 to this bit to clear the USB reset received status bit. Only used by the device mode. 0b - No reset received 1b - Reset received
26 AAI	AAI Interrupt on async advance. System software can force the controller to issue an interrupt the next time the USB DR controller advances the asynchronous schedule by writing a one to USBCMD[IAA]. This status bit indicates the assertion of that interrupt source. Only used by the host mode. 0b - No async advance interrupt 1b - Async advance interrupt
27 SEI	SEI System error. This bit is set whenever an error is detected on the system bus. If USBINTR[SEE] is set, an interrupt is generated. The interrupt and status bits remain asserted until cleared by writing a 1 to this bit. Additionally, when in host mode, USBCMD[RS] is cleared, effectively disabling the USB DR controller. For the USB DR controller in device mode, an interrupt is generated, but no other action is taken. 0b - Normal operation 1b - Error
28 FRI	FRI Frame list rollover. The 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 USBCMD[FS]) is 1024, FRINDEX rolls over every time FRINDEX [13] toggles. Similarly, if the size is 512, the USB DR controller sets this bit to a one every time FRINDEX [12] toggles. Only used by the host mode.
29 PCI	PCI Port change detect. Host mode: • The controller sets this bit when a connect status occurs on any port, a port enable/disable change occurs, an over current change occurs, or PORTSC[FPR] is set as the result of a J-K transition on the suspended port. Device mode: • The USB DR controller sets this bit when it 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 USBSTS[URI] and USBSTS[SLI], respectively.

Table continues on the next page...

Field	Function
	The controller also sets this bit when the VBUS de-asserts (occurs when the host disables port power or the device has been disconnected from the bus). This bit is not EHCI compatible.
30 UEI	UEI USB error interrupt (USBERRINT). When completion of a USB transaction results in an error condition, this bit is set by the controller. This bit is set along with the UI, if the TD on which the error interrupt occurred also had its interrupt on complete (IOC) bit set. See Section 4.15.1 in EHCI for a complete list of host error interrupt conditions. For the USB DR controller in device mode, only resume signaling is detected, all others are ignored. 0b - No error 1b - Error detected
31 UI	UI USB interrupt (USBINT). This bit is set by the 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 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.

32.3.13 USB Interrupt Enable (USBINTR)

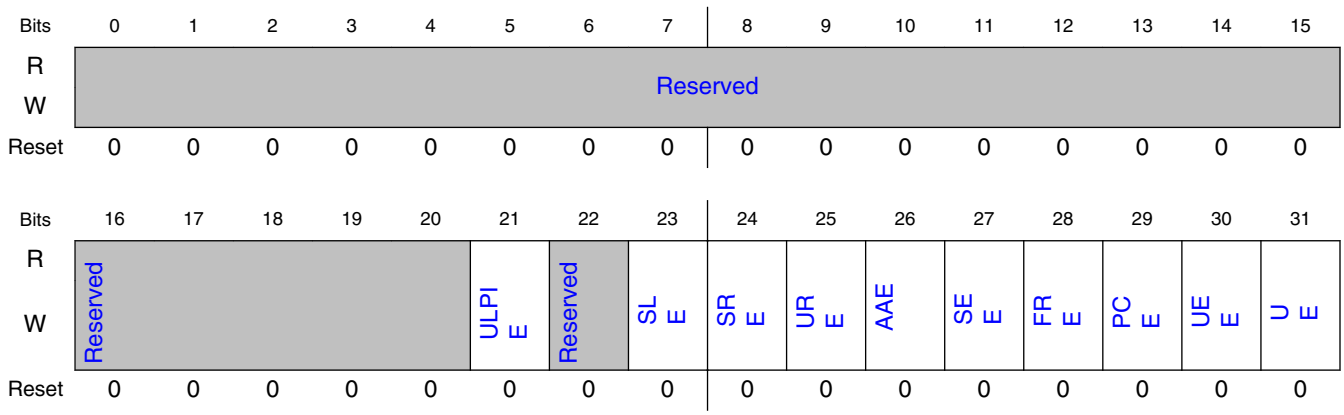
32.3.13.1 Offset

Register	Offset
USBINTR	148h

32.3.13.2 Function

The interrupts to software are enabled with the USB interrupt enable register. An interrupt is generated when a bit is set and the corresponding interrupt is active. The USB status register (USBSTS) still shows interrupt sources even if they are disabled by the USBINTR register, allowing polling of interrupt events by the software.

32.3.13.3 Diagram



32.3.13.4 Fields

Field	Function
0-20 —	- Reserved, should be cleared.
21 ULPIE	ULPIE ULPI interrupt enable. An event completion to the viewport register sets the USBSTS[ULPII]. If the ULPI enables ULPIE bit to be set, then the USBINT (USBSTS[UI]) occurs. 0b - Disable 1b - Enable
22 —	- Reserved, should be cleared.
23 SLE	SLE Sleep enable. This is a non-EHCI bit. When this bit is a one, and USBSTS[SLI] transitions, the USB DR controller issues an interrupt. The interrupt is acknowledged by software writing a one to USBSTS[SLI]. Only used in device mode. 0b - Disable 1b - Enable
24 SRE	SRE SOF received enable. This is a non-EHCI bit. When this bit is a one, and USBSTS[SRI] is a one, the controller issues an interrupt. The interrupt is acknowledged by software clearing USBSTS[SRI]. 0b - Disable 1b - Enable
25 URE	URE USB reset enable. This is a non-EHCI bit. When this bit is a one, and USBSTS[URI] is a one, the device controller issues an interrupt. The interrupt is acknowledged by software clearing USBSTS[URI] bit. Only used in device mode. 0b - Disable 1b - Enable

Table continues on the next page...

Field	Function
26 AAE	AAE Interrupt on async advance enable. When this bit is a one, and USBSTS[AAI] is a one, the controller issues an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing USBSTS[AAI]. Only used in host mode. 0b - Disable 1b - Enable
27 SEE	SEE System error enable. When this bit is a one, and USBSTS[SEI] is a one, the controller issues an interrupt. The interrupt is acknowledged by software clearing USBSTS[SEI]. 0b - Disable 1b - Enable
28 FRE	FRE Frame list rollover enable. When this bit is a one, and USBSTS[FRI] is a one, the controller issues an interrupt. The interrupt is acknowledged by software clearing USBSTS[FRI]. Only used by the host mode. 0b - Disable 1b - Enable
29 PCE	PCE Port change detect enable. When this bit is a one, and USBSTS[PCI] is a one, the controller issues an interrupt. The interrupt is acknowledged by software clearing USBSTS[PCI]. 0b - Disable 1b - Enable
30 UEE	UEE USB error interrupt enable. When this bit is a one, and USBSTS[UEI] is a one, the controller issues an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing USBSTS[UEI]. 0b - Disable 1b - Enable
31 UE	UE USB interrupt enable. When this bit is a one, and USBSTS[UI] is a one, the DR controller issues an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing USBSTS[UI]. 0b - Disable 1b - Enable

32.3.14 USB Frame Index (FRINDEX)

32.3.14.1 Offset

Register	Offset
FRINDEX	14Ch

32.3.14.2 Function

In host mode, the frame index register is used by the controller to index the periodic frame list. The register updates every 125 microseconds (once each microframe). 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 USBCMD[FS].

This register must be written as a DWord. Byte writes produce-undefined results. This register cannot be written unless the USB DR controller is in the Halted state as indicated by the USBSTS[HCH]. A write to this register while USBCMD[RS] is set produces undefined results. Writes to this register also affect the SOF value.

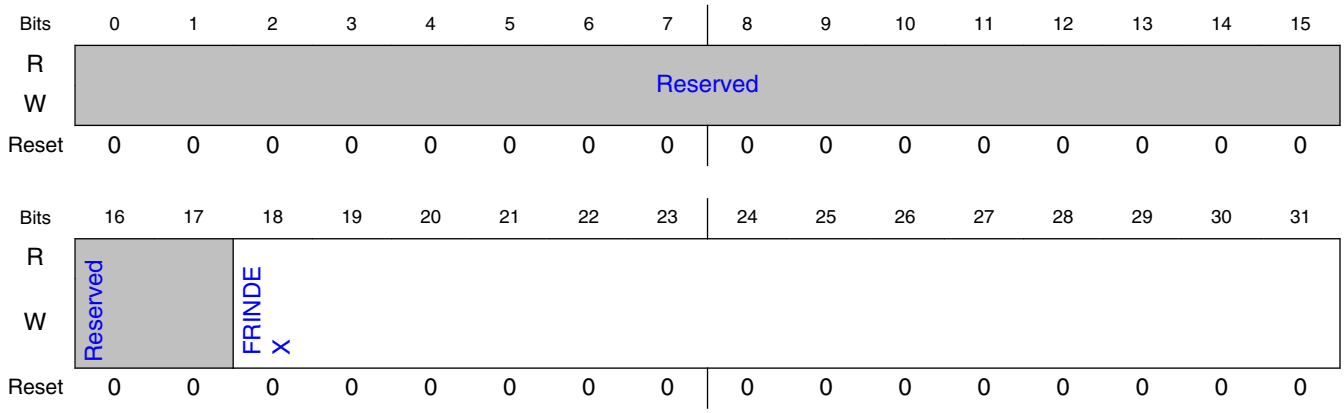
In device mode, this register is read-only and the USB DR controller updates the FRINDEX[13-3] register from the frame number indicated by the SOF marker. Whenever a SOF is received by the USB bus, FRINDEX[13-3] is checked against the SOF marker. If FRINDEX[13-3] is different from the SOF marker, FRINDEX[13-3] is set to the SOF value and FRINDEX[2-0] is cleared (that is, SOF for 1 ms frame). If FRINDEX[13-3] is equal to the SOF value, FRINDEX[2-0] is incremented (that is, SOF for 125-µsec microframe).

The table below illustrates values of N based on the value of the Frame List Size in the USBCMD register, when used in host mode.

Table 32-2. FRINDEX N Values

USBCMD[FS]	Frame List Size	FRINDEX N value
000	1024 elements (4096 bytes)	12
001	512 elements (2048 bytes)	11
010	256 elements (1024 bytes)	10
011	128 elements (512 bytes)	9
100	64 elements (256 bytes)	8
101	32 elements (128 bytes)	7
110	16 elements (64 bytes)	6
111	8 elements (32 bytes)	5

32.3.14.3 Diagram



32.3.14.4 Fields

Field	Function
0-17 —	- Reserved, should be cleared.
18-31 FRINDEX	FRINDEX Frame index. The value in this register increments at the end of each time frame (for example, microframe). 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 microframes) before moving to the next index. 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.

32.3.15 USB Device Address [device mode] (DEVICEADDR)

32.3.15.1 Offset

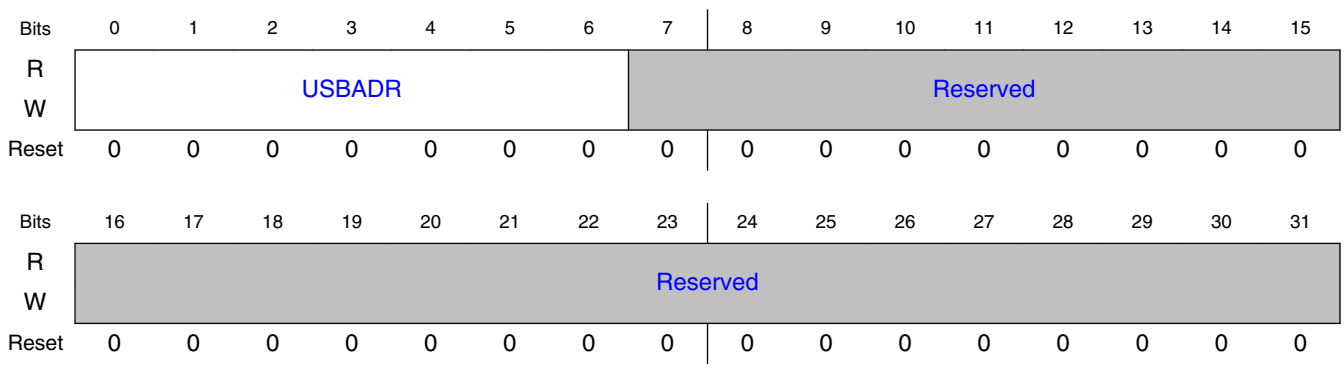
Register	Offset
DEVICEADDR	154h

32.3.15.2 Function

The device address register is not defined in the EHCI specification. In device mode, the upper seven bits of this register represent the device address. After any controller reset or a USB reset, the device address is set to the default address (0). The default address will match all incoming addresses. Software reprograms the address after receiving a SET_ADDRESS descriptor.

This register is shared between the host and device mode functions. In device mode, it is the DEVICEADDR register; in host mode, it is the PERIODICLISTBASE register. See Periodic Frame List Base Address [host mode] (PERIODICLISTBASE) , for more information.

32.3.15.3 Diagram



32.3.15.4 Fields

Field	Function
0-6 USBADR	USBADR Device address. This field corresponds to the USB device address.
7-31 —	- Reserved, should be cleared.

32.3.16 Periodic Frame List Base Address [host mode] (PERIODICLISTBASE)

32.3.16.1 Offset

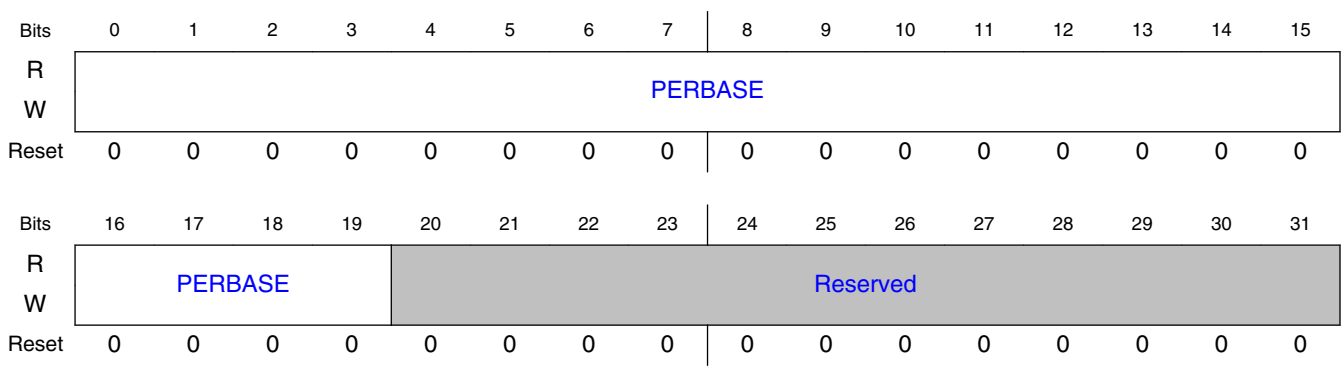
Register	Offset
PERIODICLISTBASE	154h

32.3.16.2 Function

This register contains the beginning address of the periodic frame list in the system memory. The host controller driver loads this register prior to starting the schedule execution by the controller. The memory structure referenced by this physical memory pointer is assumed to be 4 KB aligned. The contents of this register are combined with the frame index register (FRINDEX) to enable the controller to step through the periodic frame list in sequence.

This register is shared between the host and device mode functions. In host mode, it is the PERIODICLISTBASE register; in device mode, it is the DEVICEADDR register. See USB Device Address [device mode] (DEVICEADDR) , for more information.

32.3.16.3 Diagram



32.3.16.4 Fields

Field	Function
0-19 PERBASE	PERBASE Base address. Correspond to memory address signal [31:12]. Only used in the host mode.
20-31 —	- Reserved, should be cleared.

32.3.17 Next Asynchronous List Addr [host mode] (ASYNCLISTADDR)

32.3.17.1 Offset

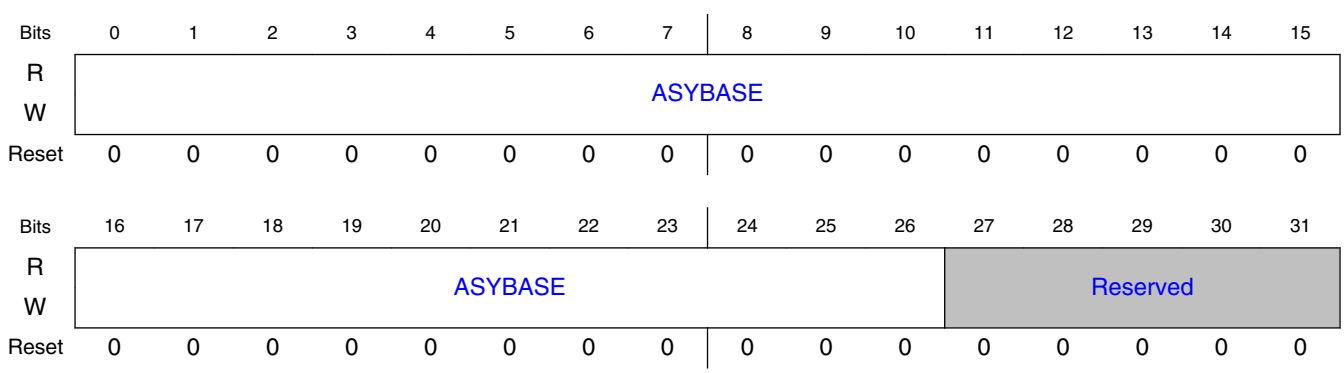
Register	Offset
ASYNCLISTADDR	158h

32.3.17.2 Function

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 always return zeros when read.

Note that this register is shared between the host and device mode functions. In host mode, it is the ASYNCLISTADDR register; in device mode, it is the ENDPOINTLISTADDR register. See Address at Endpoint List [device mode] (ENDPOINTLISTADDR) , for more information.

32.3.17.3 Diagram



32.3.17.4 Fields

Field	Function
0-26 ASYBASE	ASYBASE Link pointer low (LPL). These bits correspond to memory address signals [31:5]. This field may only reference a queue head (QH). Only used by the host controller.
27-31 —	- Reserved, should be cleared.

32.3.18 Address at Endpoint List [device mode] (ENDPOINTLISTADDR)

32.3.18.1 Offset

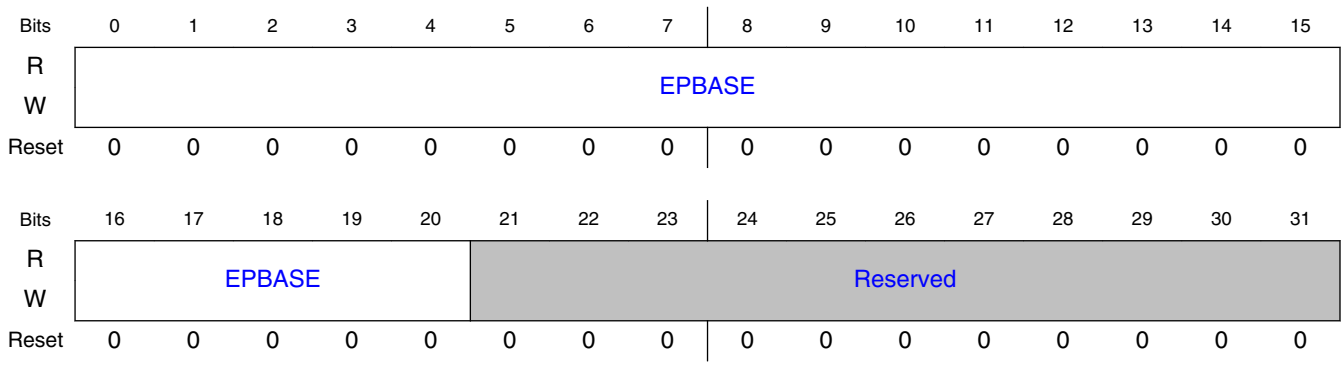
Register	Offset
ENDPOINTLISTADDR	158h

32.3.18.2 Function

The Endpoint list address register is not defined in the EHCI specification. 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 always return zeros when read. The memory structure referenced by this physical memory pointer is assumed to be 64-bytes. The queue head is actually a 48-byte structure, but must be aligned on 64-byte boundary. However, the ENDPOINTLISTADDR[EPBASE] has a granularity of 2 KB, so in practice the queue head should be 2 KB aligned.

This register is shared between the host and device mode functions. In device mode, it is the ENDPOINTLISTADDR register; in host mode, it is the ASYNCLISTADDR register. See Next Asynchronous List Addr [host mode] (ASYNCLISTADDR) , for more information.

32.3.18.3 Diagram



32.3.18.4 Fields

Field	Function
0-20 EPBASE	EPBASE Endpoint list address. Address of the top of the Endpoint list.
21-31 —	- Reserved, should be cleared.

32.3.19 Master Interface Data Burst Size (BURSTSIZE)

32.3.19.1 Offset

Register	Offset
BURSTSIZE	160h

32.3.19.2 Function

The master interface data burst size register is not defined in the EHCI specification. This register is used to control and dynamically change the burst size used during data movement on the initiator (master) interface.

32.3.19.3 Diagram

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

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

32.3.19.4 Fields

Field	Function
0-15 —	- Reserved, should be cleared.
16-23 TXPBURST	TXPBURST Programable TX burst length. This register represents the maximum length of a burst in 32-bit words while moving data from system memory to the USB bus. Must not be set to greater that 16.
24-31 RXPBURST	RXPBURST Programable RX burst length. This register represents the maximum length of a burst in 32-bit words while moving data from the USB bus to system memory. Must not be set to greater than 16.

32.3.20 Transmit FIFO Tuning Controls (TXFILLTUNING)

32.3.20.1 Offset

Register	Offset
TXFILLTUNING	164h

32.3.20.2 Function

The transmit FIFO tuning controls register is not defined in the EHCI specification. This register is used to control and dynamically change the burst size used during data movement on device DMA transfers. It is only used in host mode.

The fields in this register control performance tuning associated with how the USB DR module posts data to the TX latency FIFO before moving the data onto the USB bus. The specific areas of performance include 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_s = Total Packet Flight Time (send-only) packet ($Ts = T_0 + T_1$)

T_{ff} = Time to fetch packet into TX FIFO up to specified level.

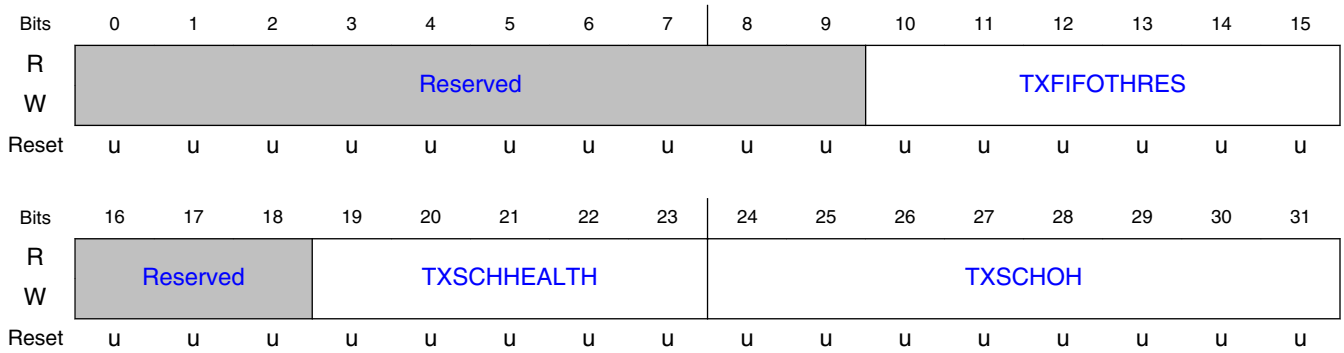
T_p = Total Packet Time (fetch and send) packet ($T_p = T_{ff} + T_s$)

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 any time 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 module eventually recovers, a mark is made in 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 TSCHHEALTH (T_{ff}) parameter described below.

NOTE

This register cannot be written before the controller is programmed to the host mode. The register read 0x0000_0000 before the controller is configured to the host mode.

32.3.20.3 Diagram



32.3.20.4 Fields

Field	Function
0-9 —	- Reserved, should be cleared.
10-15 TXFIFOTHRES	TXFIFOTHRES FIFO burst threshold. Control 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 USBMODE[SDIS] (stream disable bit) is set. When USBMODE[SDIS] is set, the host controller behaves as if TXFIFOTHRES is set to the maximum value.
16-18 —	- Reserved, should be cleared.
19-23 TXSCHHEALTH	TXSCHHEALTH Scheduler health counter. Increment 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 clears the counter and this counter stops counting after reaching the maximum of 31.
24-31 TXSCHOH	TXSCHOH Scheduler overhead. These bits add an additional fixed offset to the schedule time estimator described above as T_{ff}. 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 1.267µs when a device is connected in high-speed mode. • The time unit represented in this register is 6.333µs when a device is connected in low-/full-speed mode.

Field	Function
	For most applications, TXSCHOH can be set to 4 or less. A good value to begin with is: TXFIFOTHRES x (16 x 4 bytes-per-word) ÷ (40 x TimeUnit), always rounded to the next higher integer. TimeUnit is either 1.267 or 6.333 as noted earlier in this description. If this value of TXSCHOH results in a TXSCHHEALTH count of 0 per second, try lowering the value by 1 if optimizing performance is desired. If TXSCHHEALTH exceeds 10 per second, try raising the value by 1. If streaming mode is disabled via the USBMODE register, treat TXFIFOTHRES as the maximum value for purposes of the TXSCHOH calculation.

32.3.21 ULPI Register Access (ULPI_VIEWPORT)

32.3.21.1 Offset

Register	Offset
ULPI_VIEWPORT	170h

32.3.21.2 Function

The ULPI register access provides indirect access to the ULPI PHY register set. Although the controller modules perform access to the ULPI PHY register set, there may be extraordinary circumstances where software may need direct access. Be advised that writes to the ULPI through the ULPI viewport can substantially harm standard USB operations. Currently no usage model has been defined where software should need to execute writes directly to the ULPI. Note that executing read operations though the ULPI viewport should have no harmful side effects to standard USB operations. Also note that if the ULPI interface is not enabled, this register will always read zeros.

There are two operations that can be performed with the ULPI viewport, wakeup and read /write operations. The wakeup operation is used to put the ULPI interface into normal operation mode and re-enable the clock if necessary. A wakeup operation is required before accessing the registers when the ULPI interface is operating in low power mode, serial mode, or carkit mode. The ULPI state can be determined by reading the sync state bit (ULPISS). If this bit is set, then the ULPI interface is running in normal operation mode and can accept read/write operations. If the ULPISS is cleared, then read/write operations will not be able execute. Undefined behavior results if a read or write operation is performed when ULPISS is cleared. To execute a wakeup operation, write all

32-bits of the ULPI Viewport where ULPIPORT is constructed appropriately and the ULPIWU bit is set and the ULPIRUN bit is cleared. Poll the ULPI Viewport until ULPIWU is cleared for the operation to complete.

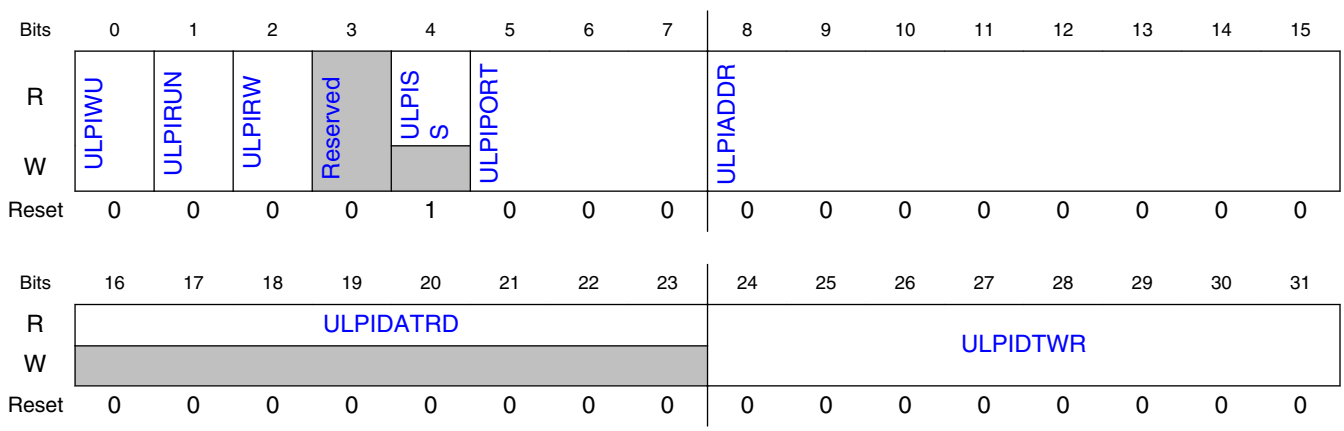
To execute a read or write operation, write all 32-bits of the ULPI Viewport where ULPIDATWR, ULPIADDR, ULPIPORT, ULPIRW are constructed appropriately and the ULPIRUN bit is set. Poll the ULPI Viewport until ULPIRUN is cleared for the operation to complete. For read operations, ULPIDATRD is valid once ULPIRUN is cleared.

The polling method above can be replaced with interrupts using the ULPI interrupt defined in the USBSTS and USBINTR registers. When a wakeup or read/write operation completes, the ULPI interrupt is set.

NOTE

Read or write to the ULPI PHY extended register set (address > 3Fh) is not supported.

32.3.21.3 Diagram



32.3.21.4 Fields

Field	Function
0 ULPIWU	ULPIWU ULPI Wake Up. Writing 1 to this bit begins the wakeup operation. This bit automatically transitions to 0 after the wakeup is complete. Once this bit is set, it cannot be cleared by software. **NOTE:** The driver must never execute a wakeup and a read/write operation at the same time.
1	ULPIRUN

Table continues on the next page...

Field	Function
ULPIRUN	ULPI Run. Writing 1 to this bit begins a read/write operation. This bit automatically transitions to 0 after the read/write is complete. Once this bit is set, it can not be cleared by software. **NOTE:** The driver must never execute a wakeup and a read/write operation at the same time.
2 ULPIRW	ULPIRW This bit selects between running a read or write operation to the ULPI. 0b - Read 1b - Write
3 —	- Reserved, should be cleared.
4 ULPISS	ULPISS This bit represents the state of the ULPI interface. Before reading this bit, the ULPIPORT field should be set accordingly if used with the multi-port host. Otherwise, this field should always remain 0. 0b - Any other state (that is, carkit, serial, low power). 1b - Normal Sync State.
5-7 ULPIPORT	ULPIPORT For wakeup or read/write operations this value selects the port number to which the ULPI PHY is attached. Valid values are 0 and 1.
8-15 ULPIADDR	ULPIADDR When a read or write operation is commanded, the address of the operation is written to this field.
16-23 ULPIDATRD	ULPIDATRD After a read operation completes, the result is placed in this field.
24-31 ULPIDTWR	ULPIDTWR When a write operation is commanded, the data to be sent is written to this field.

32.3.22 Endpoint NAK Indicaton Register (ENDPTNAK)

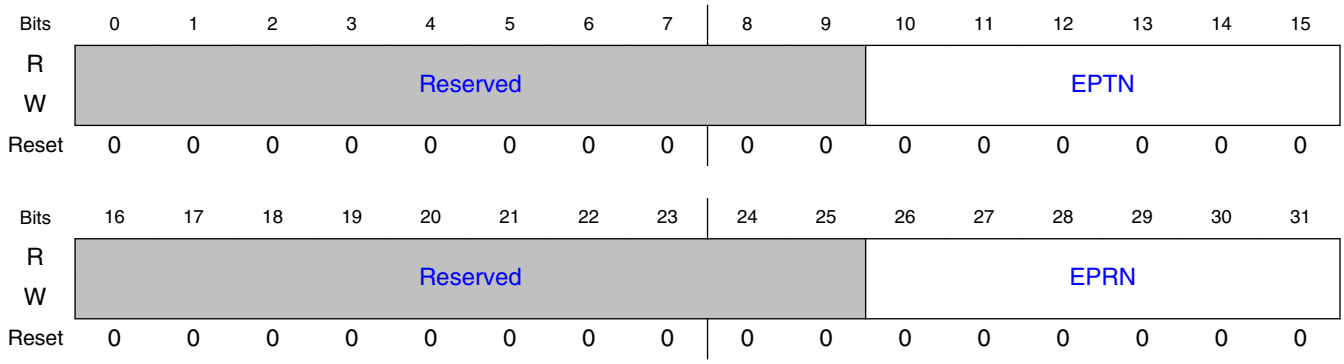
32.3.22.1 Offset

Register	Offset
ENDPTNAK	178h

32.3.22.2 Function

This register is only used in device mode.

32.3.22.3 Diagram



32.3.22.4 Fields

Field	Function
0-9 —	- Reserved.
10-15 EPTN	EPTN **TX Endpoint NAK** . Each TX Endpoint has one bit in this field. This bit is set when controller sends a NAK handshake on a received IN token for the corresponding Endpoint. EPTN[5] - Endpoint #5 ... EPTN[1] - Endpoint #1 EPTN[0] - Endpoint #0 Only used in device mode.
16-25 —	- Reserved.
26-31 EPRN	EPRN **RX Endpoint NAK** . Each RX Endpoint has one bit in this field. This bit is set when controller sends a NAK handshake on a received OUT or PING token for the corresponding Endpoint. EPRN[5] - Endpoint #5 ... EPRN[1] - Endpoint #1 EPRN[0] - Endpoint #0 Only used in device mode.

32.3.23 Endpoint NAK Indication Enable Register (ENDPTNAKEN)

32.3.23.1 Offset

Register	Offset
ENDPTNAKEN	17Ch

32.3.23.2 Function

This register is only used in device mode.

32.3.23.3 Diagram

Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Reserved										EPTNE					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

32.3.23.4 Fields

Field	Function
0-9 —	- Reserved.
10-15 EPTNE	EPTNE **TX Endpoint NAK Enable** . Each bit is an enable bit for corresponding TX Endpoint NAK bit. If this bit is set and corresponding TX Endpoint NAK bit is set, the NAK interrupt bit is set. EPTNE[5] - Endpoint #5 ... EPTNE[1] - Endpoint #1 EPTNE[0] - Endpoint #0

Table continues on the next page...

Field	Function
	Only used in device mode.
16-25 —	- Reserved.
26-31 EPRNE	EPRNE **RX Endpoint NAK Enable** . Each bit is an enable bit for corresponding RX Endpoint NAK bit. If this bit is set and corresponding RX Endpoint NAK bit is set, the NAK interrupt bit is set. EPRNE[5] - Endpoint #5 ... EPRNE[1] - Endpoint #1 EPRNE[0] - Endpoint #0 Only used in device mode.

32.3.24 Configured Flag Register (CONFIGFLAG)

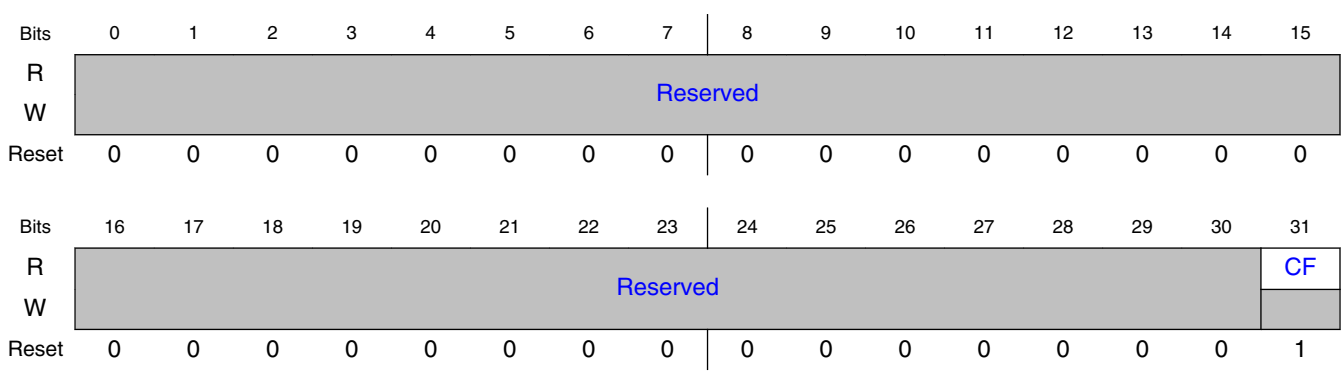
32.3.24.1 Offset

Register	Offset
CONFIGFLAG	180h

32.3.24.2 Function

This EHCI register is not used in this implementation. A read from this register returns a constant of a 0x0000_0001 to indicate that all port routings default to this host controller.

32.3.24.3 Diagram



32.3.24.4 Fields

Field	Function
0-30 —	- Reserved.
31 CF	CF Configure flag. Always 1. Indicating all port routings default to this host.

32.3.25 Port Status/Control (PORTSC)

32.3.25.1 Offset

Register	Offset
PORTSC	184h

32.3.25.2 Function

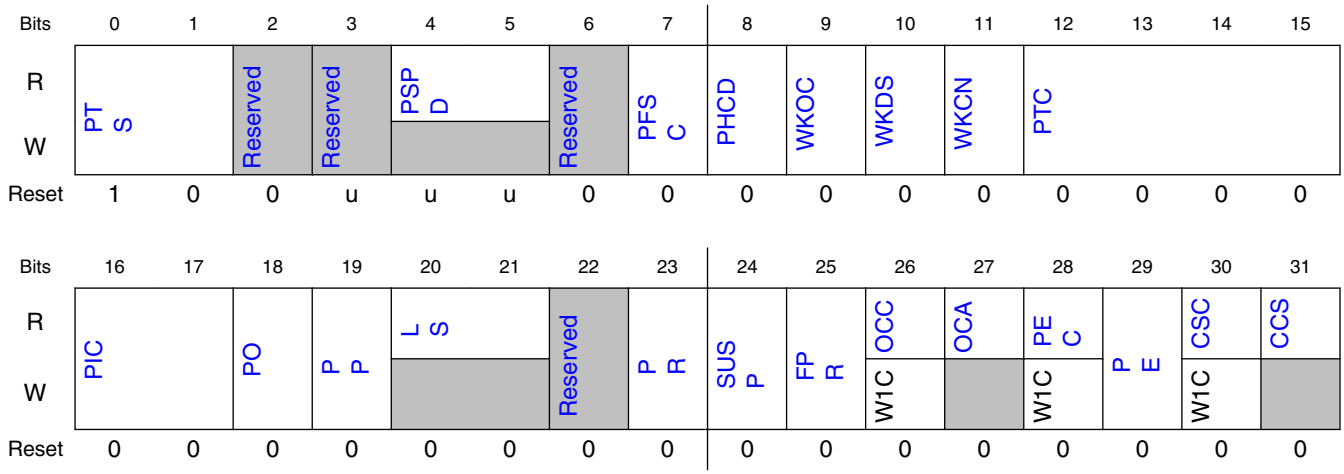
The port status and control (PORTSC) register is only reset when power is initially applied or in response to a controller reset. The initial conditions of a port are:

- No device connected
- Port disabled

If the port has port power control, this state remains until software applies power to the port by setting port power to one.

In device mode, the USB DR controller 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.

32.3.25.3 Diagram



32.3.25.4 Fields

Field	Function
0-1 PTS	PTS Port transceiver select. This register bit is used to control which parallel transceiver interface is selected. This bit is not defined in the EHCI specification. 00b - Reserved 01b - Reserved, should be cleared 10b - ULPI parallel interface 11b - Reserved
2 —	- Reserved, should be cleared
3 —	- Reserved, should be cleared
4-5 PSPD	PSPD Port speed. This read-only register field indicates the speed at which the port is operating. This bit is not defined in the EHCI specification. 00b - Full-speed 01b - Low-speed 10b - High-speed 11b - Undefined
6 —	- Reserved, should be cleared
7 PFSC	PFSC Port force full-speed connect. Used to disable the chirp sequence that allows the port to identify itself as a HS port. This is useful for testing FS configurations with a HS host, hub or device.

Table continues on the next page...

Field	Function
	This bit is not defined in the EHCI specification. This bit is for debugging purposes. 0b - Allow the port to identify itself as high speed. 1b - Force the port to only connect at full speed.
8 PHCD	PHCD PHY low power suspend. This bit is not defined in the EHCI specification. 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. Device mode: • The PHY can be put into low power suspend - when the device is not running (USBCMD[RS] = 0b) or suspend signaling is detected on the USB. Low power suspend is cleared automatically when the resume signaling has been detected or when forcing port resume. Reading this bit indicates the status of the PHY. **NOTE:** If there is no clock connected to the USB*n* _CLK signals, PHCD must be set and the following registers should not be written: DEVICE_ADDR/PERIODICLISTBASE, PORTSC, ENDPTCTRL0, ENDPTCTRL1, ENDPTCTRL2 , ENDPTCTRL3, ENDPTCTRL4, ENDPTCTRL5 . 0b - Normal PHY operation. 1b - Signal the PHY to enter low power suspend mode
9 WKOC	WKOC Wake on over-current enable. Writing this bit to a one enables the port to be sensitive to over-current conditions as wake-up events. This field is zero if Port Power (PP) is zero. This bit is used only in Host mode and used by an external power control circuit.
10 WKDS	WKDS Wake on disconnect enable. Writing this bit to a one enables the port to be sensitive to device disconnects as wake-up events. This field is zero if Port Power(PP) is zero or in device mode. This bit is used only in Host mode and used by an external power control circuit.
11 WKCN	WKCN Wake on connect enable. Writing this bit to a one enables the port to be sensitive to device connects as wake-up events. This field is zero if Port Power(PP) is zero or in device mode. This bit is used only in Host mode and used by an external power control circuit.
12-15 PTC	PTC Port test control. Any other value than zero indicates that the port is operating in test mode. Note that for K_STATE test mode (Test_K), a controller reset (Host mode) or power cycle (Device mode) is required after the test completes. Refer to Chapter 7 of the USB Specification Revision 2.0 [3] for details on each test mode. 0000b - Not Enabled 0001b - J_STATE 0010b - K_STATE

Table continues on the next page...

Field	Function
	0011b - SEQ_NAK 0100b - Packet 0101b - FORCE_ENABLE 0110-1111b - Reserved, should be cleared
16-17 PIC	PIC Port indicator control. Control the link indicator signals. These signals are valid for host mode only. This field is output from the controller for use by an external LED driving circuit. 00b - Off 01b - Amber 10b - Green 11b - Undefined
18 PO	PO Port owner. 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 module (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 hand-off is not implemented in this design, therefore this bit is always 0.
19 PP	PP Port power. 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, and so on. When an over-current condition is detected on a powered port, the PP bit in each affected port is transitioned by the host controller driver from a one to a zero (removing power from the port). This feature is implemented in the host controller (PPC = 1). In a device-only implementation port power control is not necessary, thus PPC and PP = 0. 0b - USBDR writes 0 for DRVVBUS bit of OTGControl register in PHY. 1b - USBDR writes 1 for DRVVBUS bit of OTGControl register in PHY. The OTG Control register is defined in ULPI specification.
20-21 LS	LS Line status. Reflect the current logical levels of the USB D+ (bit 11) and D- (bit 10) signal lines. The use of line status by the host controller driver is not necessary (unlike EHCI), because the connection of FS and LS is managed by hardware. 00b - SE0 01b - K-state 10b - J-state 11b - Undefined
22 —	- Reserved, should be cleared
23 PR	PR Port reset. Host mode: • 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. Device mode:

Table continues on the next page...

Field	Function
	• 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(PP) is zero. 0b - Port is not in reset. 1b - Port is in reset.
24 SUSP	SUSP Suspend. Host mode: • The port enabled bit (PE) and suspend (SUSP) bit define the port states as follows: 0x Disable 10 Enable 11 Suspend • When in suspend state, downstream propagation of data is blocked on this port, except for port reset. The blocking occurs at the end of the current transaction if a transaction was in progress when this bit was written to 1. In the suspend state, the port is sensitive to resume detection. Note that the bit status does not change until the port is suspended and that there may be a delay in suspending a port if there is a transaction currently in progress on the USB. • The module unconditionally sets this bit to zero when software clears the FPR bit. A write of zero to this bit is ignored by the host controller. If host software sets this bit to a one when the port is not enabled (that is, port enabled bit is a zero) the results are undefined. • This field is zero if Port Power (PP) is zero in host mode. In device mode this bit is a read-only status bit. 0b - Device mode: Port not in suspend state. Default. 1b - Port in suspend state.
25 FPR	FPR Force port resume. This bit is not-EHCI compatible. • Software sets this bit to one to drive resume signaling. The 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 a J-to-K transition is detected, USBSTS[PCI] (port change detect) is also set. 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 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 affect because the port controller will time the resume operation and clear this bit when the port control state switches to HS or FS idle. • This field is zero if Port Power (PP) is zero in host mode. Device mode: • After the device has been in Suspend State for 5 ms or more, software can set this bit to one to drive resume signaling before clearing. The USB DR 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 is cleared when the device returns to normal operation. Also, when this bit transitions to a one because a J-to-K transition detected, USBSTS[PCI] is also set. 0b - No resume (K-state) detected/driven on port. 1b - Resume detected/driven on port.

Table continues on the next page...

Field	Function
26 OCC	OCC Over-current change. This bit gets set when there is a change to over-current active. Software clears this bit by writing a one to this bit position. Host mode: • The user can provide over-current detection to the USB *n*_PWRFAULT signal for this condition. Device mode: • This bit must always be 0. 0b - No over current. 1b - Over current detect.
27 OCA	OCA Over-current active. This bit will automatically transition from one to zero when the over-current condition is removed. Host mode: • The user can provide over-current detection to the USB *n* _PWRFAULT signal for this condition. Device mode: • This bit must always be 0. **NOTE:** Whenever over-current condition occurs, the correct operation of controller is not guaranteed even if over-current condition goes away. Software should reset the controller by writing in USBCMD[RST] bit and repeat the host controller initialization steps again before expecting the controller to behave properly. 0b - Port not in over-current condition. 1b - Port currently in over-current condition.
28 PEC	PEC Port enable/disable change. For the root hub, this bit gets set 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. In device mode: • The device port is always enabled. (This bit is zero.) This field is zero if Port Power(PP) is zero. 0b - No change. 1b - Port disabled.
29 PE	PE Port enabled/disabled. Host mode: • Ports can only be enabled by the 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 and bus events. • When the port is disabled, (0) downstream propagation of data is blocked except for reset. • This field is zero if Port Power(PP) is zero in host mode.

Table continues on the next page...

Field	Function
	Device mode: • The device port is always enabled. (This bit is one.)
30 CSC	CSC Connect change status. Host mode: • This bit indicates a change has occurred in the port's Current Connect Status. the 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 is '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(PP) is zero. Device mode: • This bit is undefined. 0b - No change 1b - Connect Status has changed.
31 CCS	CCS Current Connect Status. Host Mode: This value reflects the current state of the port, and may not correspond directly to the event that caused the Connect Status Change bit to be set. This field is zero if Port Power(PP) is zero in host mode. Device Mode: A one indicates that the device successfully attached and is operating in either high speed or full speed as indicated by the PSPD bits in this register. A zero indicates that the device did not attach successfully or was forcibly disconnected by the software writing a zero to USBCMD[RS] (run bit). It does not state the device being disconnected or suspended. A value of zero also occurs if vbus de-asserts either when the host disables port power or the device has been disconnected from the bus. 0b - Host Mode: No device is present. Device Mode: Not attached. 1b - Host Mode: Device is present on port. Device Mode: Attached.

32.3.26 USB Device Mode (USBMODE)

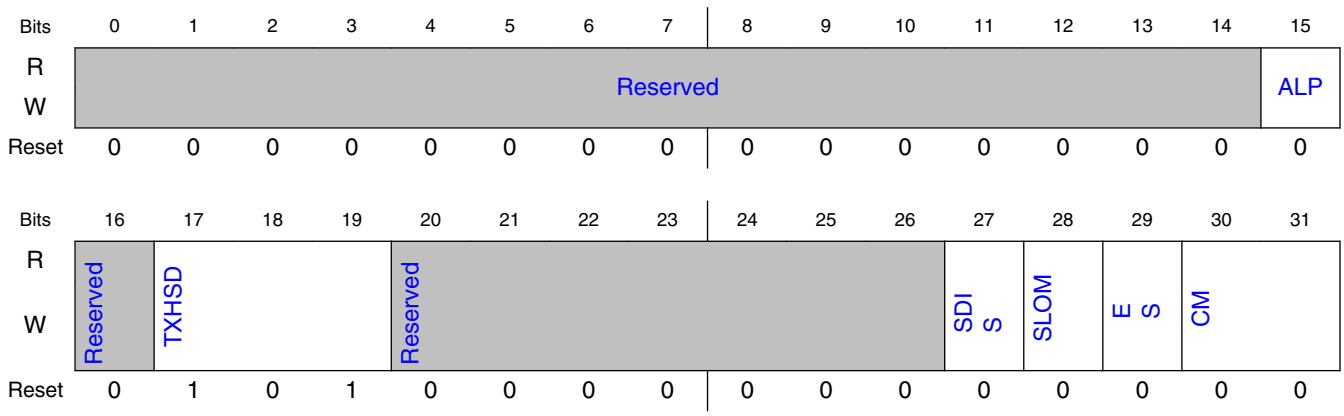
32.3.26.1 Offset

Register	Offset
USBMODE	1A8h

32.3.26.2 Function

The USB mode register is not defined in the EHCI specification. This register controls the operating mode of the module.

32.3.26.3 Diagram



32.3.26.4 Fields

Field	Function
0-14 —	- Reserved
15 ALP	ALP Auto Low Power. When the software enables this bit, the PHY is put in low power when in suspend (PORTSC[SUSP] is enabled) even if PORTSC[PHCD] is not set.
16 —	- Reserved
17-19 TXHSD	TXHSD Tx to Tx HS Delay. This bit controls the value of TX to TX HS interpacket delay by changing the internal delay count. The value of the global TX to TX interpacket delay depends on this internal counter and on the intrinsic PHY TX end delay and TX start delay values. The TX to TX interpacket gap must be within the interval [88,192] bit times. 88 = Controller internal delay – Tx End Delay + Tx Start Delay = 192 (HS bit times) The values of the internal controller counter in terms of PHY clock cycles are shown below. These values depend on the width of the PHY interface (8 or 16 bit). Value---Delay(8-bit, 60MHz)---Delay(16-bit, 30MHz)

Table continues on the next page...

Field	Function
	000---10---5 001---11---5 010---12---6 011---13---6 100---14---7 101---15---7 110---16---8 111---17---8
20-26 —	- Reserved
27 SDIS	SDIS Stream disable In host mode, setting this bit 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 or a complete packet is stored in FIFO before the packet is launched onto the USB. Note that time duration to pre-fill the FIFO becomes significant when stream disable is active. See Transmit FIFO Tuning Controls (TXFILLTUNING) to characterize the adjustments needed for the scheduler when using this feature. Also note that in systems with high system bus utilization, setting this bit will ensure no overruns or underruns during operation, at the expense of link utilization. For those who desire optimal link performance, SDIS can be left clear, and the rules used under the description of the TXFILLTUNING register to limit underruns/overruns. In device mode, setting this bit 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 that in high-speed mode, all packets received are responded to with a NYET handshake when stream disable is active. 0b - Inactive. 1b - Active.
28 SLOM	SLOM Setup lockout mode. In device mode, this bit controls behavior of the setup lock mechanism. 0b - Setup lockouts on 1b - Setup lockouts off. DCD requires use of setup data buffer tripwire in USBCMD (SUTW).
29 ES	ES Endian Select. This bit can change the byte ordering of the transfer buffers to match the host microprocessor bus architecture. The bit fields in the microprocessor interface and the DMA data structures (including the setup buffer within the device QH) are unaffected by the value of this bit, because they are based upon 32-bit words. The big endian implementation is BE-8.This means that access is the same for big endian and little endian only for one byte(8 bits).For accesses larger than 1 byte,the byte order is reversed from little endian to big endian. 0b - Little endian

Table continues on the next page...

Field	Function
	1b - Big endian
30-31 CM	CM Controller mode This register can only be written once after reset. If it is necessary to switch modes, software must reset the controller by writing to USBCMD[RST] before reprogramming this register. Defaults to the idle state and needs to be initialized to the desired operating mode after reset. 00b - Idle (default). 01b - Reserved, should be cleared. 10b - Device controller . 11b - Host controller .

32.3.27 Endpoint Setup Status (ENDPTSETUPSTAT)

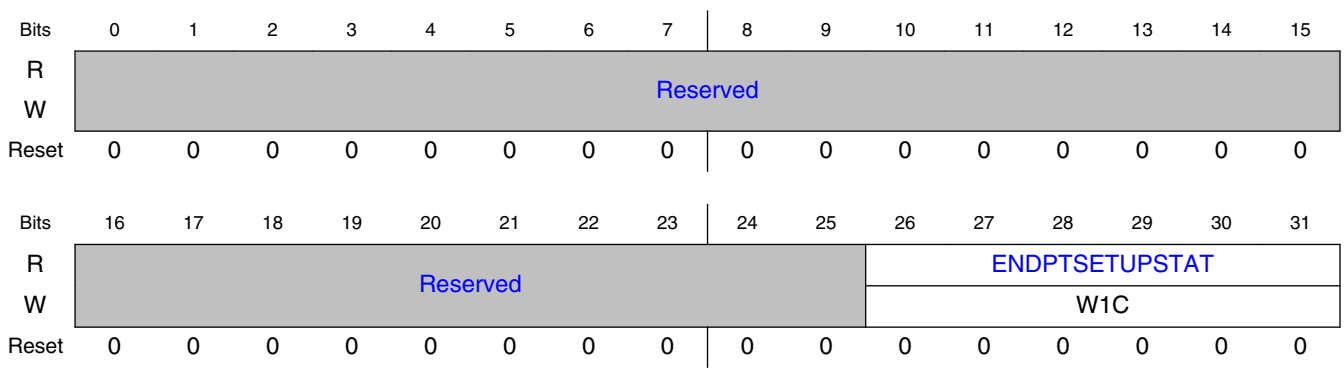
32.3.27.1 Offset

Register	Offset
ENDPTSETUPSTAT	1ACh

32.3.27.2 Function

The Endpoint setup status register is not defined in the EHCI specification. This register contains the Endpoint setup status. It is only used in device mode.

32.3.27.3 Diagram



32.3.27.4 Fields

Field	Function
0-25 —	- Reserved, should be cleared.
26-31 ENDPTSETUPSTAT	ENDPTSETUPSTAT Setup Endpoint status. For every setup transaction that is received, a corresponding bit in this register is set. 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 lockout mechanism is engaged. This register is only used in device mode.

32.3.28 Endpoint Initialization (ENDPOINTPRIME)

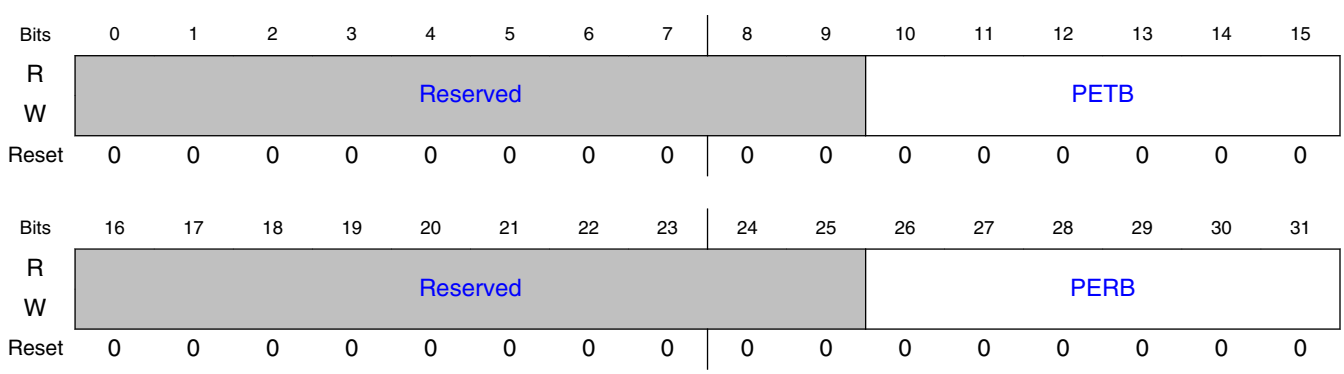
32.3.28.1 Offset

Register	Offset
ENDPOINTPRIME	1B0h

32.3.28.2 Function

The Endpoint initialization register is not defined in the EHCI specification. This register is used to initialize Endpoints. It is only used in device mode.

32.3.28.3 Diagram



32.3.28.4 Fields

Field	Function
0-9 —	- Reserved, should be cleared.
10-15 PETB	PETB Prime Endpoint transmit buffer. For each Endpoint a corresponding bit is used to request that a buffer prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction. Software should write a one to the corresponding bit when posting a new transfer descriptor to an Endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when the associated Endpoint(s) is (are) successfully primed. PETB[5] (bit 21 of the register) corresponds to Endpoint 5. Note that these bits are momentarily set by hardware during hardware re-priming operations when a dTD is retired, and the dQH is updated.
16-25 —	- Reserved, should be cleared.
26-31 PERB	PERB Prime Endpoint receive buffer. For each Endpoint, a corresponding bit is used to request a buffer prepare for a receive operation in order to respond to a USB OUT transaction. Software should write a one to the corresponding bit whenever posting a new transfer descriptor to an Endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when the associated Endpoint(s) is (are) successfully primed. PERB[5] corresponds to Endpoint 5. Note that these bits are momentarily set by hardware during hardware re-priming operations when a dTD is retired, and the dQH is updated.

32.3.29 Endpoint Flush (ENDPTFLUSH)

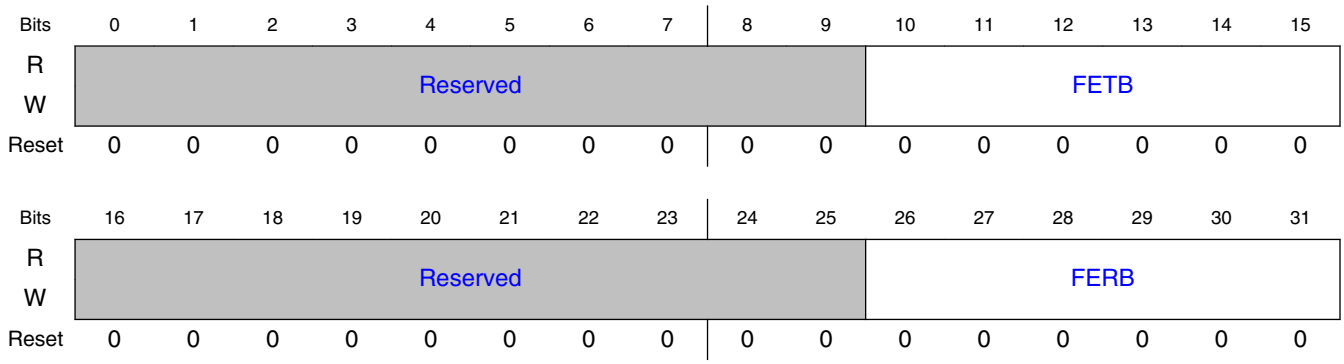
32.3.29.1 Offset

Register	Offset
ENDPTFLUSH	1B4h

32.3.29.2 Function

The Endpoint flush register is not defined in the EHCI specification. This register is only used in device mode.

32.3.29.3 Diagram



32.3.29.4 Fields

Field	Function
0-9 —	- Reserved, should be cleared.
10-15 FETB	FETB Flush Endpoint transmit buffer. Writing a one to a bit(s) in this register will cause the associated Endpoint(s) to clear any primed buffers. If a packet is in progress for one of the associated Endpoints, then that transfer will continue until completion. Hardware will clear this register after the Endpoint flush operation is successful. FETB[5] (bit 21 of the register) corresponds to Endpoint 5.
16-25 —	- Reserved, should be cleared.
26-31 FERB	FERB Flush Endpoint receive buffer. Writing a one to a bit(s) will cause the associated Endpoint(s) to clear any primed buffers. If a packet is in progress for one of the associated Endpoints, then that transfer will continue until completion. Hardware will clear this register after the Endpoint flush operation is successful. FERB[5] corresponds to Endpoint 5.

32.3.30 Endpoint Status (ENDPTSTATUS)

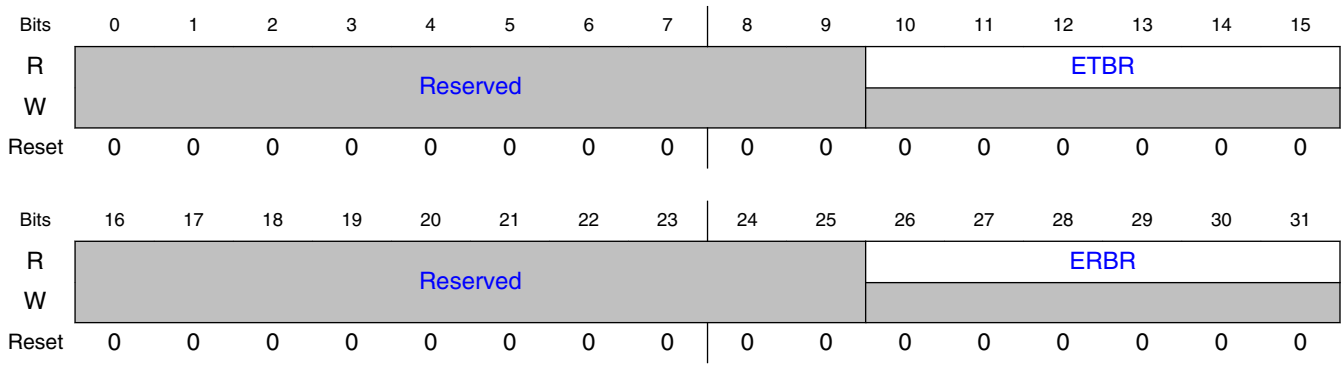
32.3.30.1 Offset

Register	Offset
ENDPTSTATUS	1B8h

32.3.30.2 Function

The Endpoint status register is not defined in the EHCI specification. This register is only used in device mode.

32.3.30.3 Diagram



32.3.30.4 Fields

Field	Function
0-9 —	- Reserved, should be cleared
10-15 ETBR	ETBR Endpoint transmit buffer ready. One bit for each Endpoint indicates status of the respective Endpoint buffer. This bit is set by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and Endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. ETBR[5] (bit 21 of the register) corresponds to Endpoint 5. Note that these bits are momentarily cleared by hardware during hardware Endpoint re-priming operations when a dTD is retired, and the dQH is updated.
16-25 —	- Reserved, should be cleared
26-31 ERBR	ERBR Endpoint receive buffer ready. One bit for each Endpoint indicates status of the respective Endpoint buffer. This bit is set by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and Endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. ERBR[5] corresponds to Endpoint 5.

Field	Function
	Note that these bits are momentarily cleared by hardware during hardware Endpoint re-priming operations when a dTD is retired, and the dQH is updated.

32.3.31 Endpoint Complete (ENDPTCOMPLETE)

32.3.31.1 Offset

Register	Offset
ENDPTCOMPLETE	1BCh

32.3.31.2 Function

The Endpoint complete register is not defined in the EHCI specification. This register is only used in device mode.

32.3.31.3 Diagram

Bits	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved										ETCE					
W											W1C					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved										ERCE					
W											W1C					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

32.3.31.4 Fields

Field	Function
0-9 —	- Reserved, should be cleared

Table continues on the next page...

Field	Function
10-15 ETCE	ETCE Endpoint transmit complete event. Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding Endpoint queue to determine the Endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit is set simultaneously with the USBINT. Writing a one will clear the corresponding bit in this register. ETCE[5] (bit 21 of the register) corresponds to Endpoint 5.
16-25 —	- Reserved, should be cleared
26-31 ERCE	ERCE Endpoint receive complete event. Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding Endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit is set simultaneously with the USBINT. Writing a one will clear the corresponding bit in this register. ERCE[5] corresponds to Endpoint 5.

32.3.32 Endpoint Control 0 (ENDPTCTRL0)

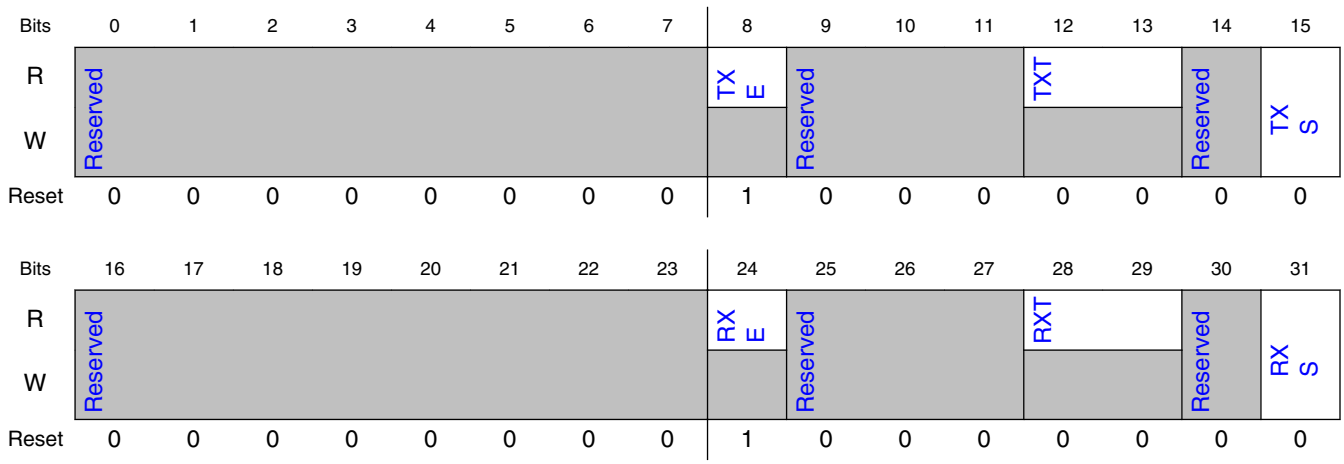
32.3.32.1 Offset

Register	Offset
ENDPTCTRL0	1C0h

32.3.32.2 Function

Endpoint control register 0 is not defined in the EHCI specification. Every device will implement Endpoint 0 as a control Endpoint.

32.3.32.3 Diagram



32.3.32.4 Fields

Field	Function
0-7 —	- Reserved, should be cleared.
8 TXE	TXE TX Endpoint enable. Endpoint zero is always enabled. 0b - Disable 1b - Enable
9-11 —	- Reserved, should be cleared.
12-13 TXT	TXT TX Endpoint type. Endpoint zero is always a control Endpoint (00).
14 —	- Reserved, should be cleared.
15 TXS	TXS TX Endpoint stall. Software can write a one to this bit to force the Endpoint to return a STALL handshake to the Host. It will continue returning STALL until the bit is cleared by software or it will automatically be cleared upon receipt of a new SETUP request. 0b - Endpoint OK 1b - Endpoint stalled
16-23 —	- Reserved, should be cleared.
24 RXE	RXE RX Endpoint enable. Endpoint zero is always enabled. 0b - Disabled

Table continues on the next page...

Field	Function
	1b - Enabled
25-27 —	- Reserved, should be cleared.
28-29 RXT	RXT RX Endpoint type. Endpoint zero is always a control Endpoint (00).
30 —	- Reserved, should be cleared.
31 RXS	RXS RX Endpoint stall Software can write a one to this bit to force the Endpoint to return a STALL handshake to the host. It will continue returning STALL until the bit is cleared by software or it will automatically be cleared upon receipt of a new SETUP request. 0b - Endpoint OK 1b - Endpoint stalled

32.3.33 Endpoint Control n (ENDPTCTRL1 - ENDPTCTRL5)

32.3.33.1 Offset

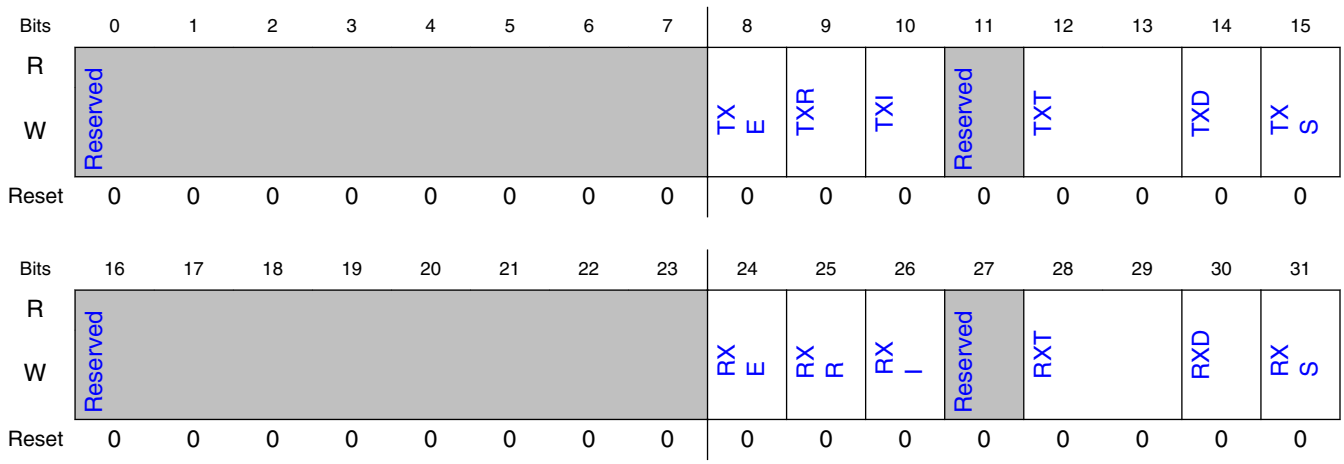
For a = 1 to 5:

Register	Offset
ENDPTCTRLa	1C0h + (a × 4h)

32.3.33.2 Function

The Endpoint control *n* registers are not defined in the EHCI specification. There is an ENDPTCTRL *n* register of each Endpoint in a device.

32.3.33.3 Diagram



32.3.33.4 Fields

Field	Function
0-7 —	- Reserved, should be cleared
8 TXE	TXE TX Endpoint enable 0b - Disabled 1b - Enabled
9 TXR	TXR TX data toggle reset. Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the Host and device.
10 TXI	TXI TX data toggle inhibit. Used only for test and should always be written as zero. Writing a one to this bit will cause this Endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet. 0b - PID sequencing enabled 1b - PID sequencing disabled
11 —	- Reserved, should be cleared
12-13 TXT	TXT TX Endpoint type **NOTE:** When only one Endpoint (RX or TX, but not both) of an Endpoint pair is used, the unused Endpoint should be configured as a bulk type Endpoint. 00b - Control 01b - Isochronous 10b - Bulk 11b - Interrupt

Table continues on the next page...

Field	Function
14 TXD	TXD TX Endpoint data source. This bit should always be written as 0, which selects the dual port memory/DMA engine as the source.
15 TXS	TXS TX Endpoint stall. 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 will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0b - Endpoint OK 1b - Endpoint stalled
16-23 —	- Reserved, should be cleared
24 RXE	RXE RX Endpoint enable 0b - Disabled 1b - Enabled
25 RXR	RXR RX data toggle reset. Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the Host and device.
26 RXI	RXI RX data toggle inhibit. This bit is only used for test and should always be written as zero. Writing a one to this bit will cause this Endpoint to ignore the data toggle sequence and always accept data packets regardless of their data PID. 0b - PID sequencing disabled 1b - PID sequencing enabled
27 —	- Reserved, should be cleared
28-29 RXT	RXT RX Endpoint type. **NOTE:** When only one Endpoint (RX or TX, but not both) of an Endpoint pair is used, the unused Endpoint should be configured as a bulk type Endpoint. 00b - Control 01b - Isochronous 10b - Bulk 11b - Interrupt
30 RXD	RXD RX Endpoint data sink. This bit should always be written as 0, which selects the dual port memory/DMA engine as the sink.
31 RXS	RXS RX Endpoint stall. 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 will continue to returning STALL until this bit is either cleared by software or automatically cleared as above,

Field	Function
	0b - Endpoint OK 1b - Endpoint stalled

32.3.34 Snoop n (SNOOP1 - SNOOP2)

32.3.34.1 Offset

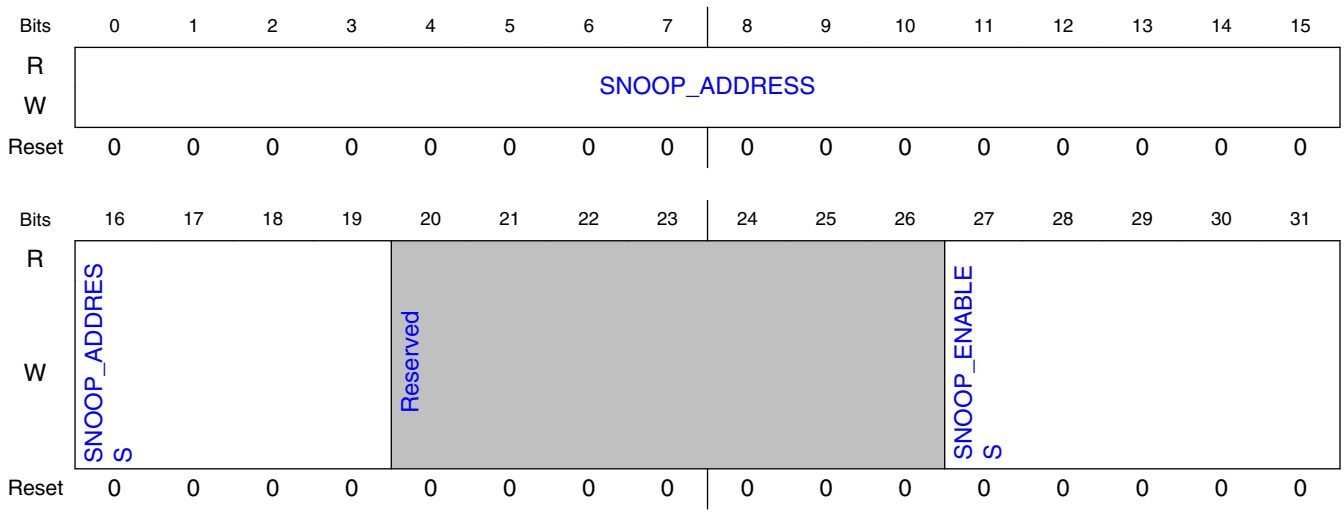
For a = 1 to 2:

Register	Offset
SNOOPa	3FCh + (a × 4h)

32.3.34.2 Function

These registers use big-endian byte ordering and are not defined in the EHCI specification. The SNOOP1 and SNOOP2 registers provide snooping control and address range selection function. Transactions that hit a snooping window will generate cache coherent transactions on the internal system bus. When the five lower bits (SNOOP *n* [27-31]) are equal to 00000, snooping is always disabled on the system bus for all DMA transfers. When SNOOP *n* [27-31] is 01011 through 11110, the twenty upper bits (SNOOP *n* [0-19]) provide the starting base address for which transactions are snooped. These twenty bits are compared to the twenty upper bits of the address provided by the DMA block of the USB controller. When a match occurs, the five lower bits are decoded as shown below. This provides a snooping region of 4 KB to 2 GB within each starting base address that is programmed by the core. The SNOOP *n* [20-26] are not used.

32.3.34.3 Diagram



32.3.34.4 Fields

Field	Function
0-19 SNOOP_ADDRESS	SNOOP_ADDRESS The starting base address for which transactions are snooped.
20-26 —	- Reserved, should be cleared
27-31 SNOOP_ENABLES	SNOOP_ENABLES Snoop_Enables 00000b - Snooping disabled 01011b - 4 KB snoop range starting at the value defined by SNOOP *n* [0-19] 01100b - 8 KB snoop range starting at the value defined by SNOOP *n* [0-18] 01101b - 16 KB snoop range starting at the value defined by SNOOP *n* [0-17] 01110b - 32 KB snoop range starting at the value defined by SNOOP *n* [0-16] 01111b - 64 KB snoop range starting at the value defined by SNOOP *n* [0-15] 10000b - 128 KB snoop range starting at the value defined by SNOOP *n* [0-14] 10001b - 256 KB snoop range starting at the value defined by SNOOP *n* [0-13] 10010b - 512 KB snoop range starting at the value defined by SNOOP *n* [0-12] 10011b - 1 MB snoop range starting at the value defined by SNOOP *n* [0-11] 10100b - 2 MB snoop range starting at the value defined by SNOOP *n* [0-10] 10101b - 4 MB snoop range starting at the value defined by SNOOP *n* [0-9] 10110b - 8 MB snoop range starting at the value defined by SNOOP *n* [0-8] 10111b - 16 MB snoop range starting at the value defined by SNOOP *n* [0-7] 11000b - 32 MB snoop range starting at the value defined by SNOOP *n* [0-6] 11001b - 64 MB snoop range starting at the value defined by SNOOP *n* [0-5] 11010b - 128 MB snoop range starting at the value defined by SNOOP *n* [0-4] 11011b - 256 MB snoop range starting at the value defined by SNOOP *n* [0-3] 11100b - 512 MB snoop range starting at the value defined by SNOOP *n* [0-2] 11101b - 1 GB snoop range starting at the value defined by SNOOP *n* [0-1] 11110b - 2 GB snoop range starting at the value defined by SNOOP *n* [0]

32.3.35 Age Count Threshold (AGE_CNT_THRESH)

32.3.35.1 Offset

Register	Offset
AGE_CNT_THRESH	408h

32.3.35.2 Function

Note that this register uses big-endian byte ordering and is not defined in the EHCI specification. The age count threshold (AGE_CNT_THRESH) register provides the aging counter threshold value used to determine the priority state of the USB controller's internal system interface. The threshold value is in units of platform clock cycles. This register should be written during system initialization or during normal system operation when the system bus interface is idle. It can be read at any time.

If the aging counter is less than the AGE_CNT_THRESH value, default (low) priority is chosen. If the aging counter is greater than or equal to the AGE_CNT_THRESH value >and USB_PRI_CTRL[PRI_LVL] = 1, an elevated priority is chosen.

The aging counter begins to count from zero when a bus access is requested. It increments every bus cycle until the bus transaction completes. At the completion of a bus transaction, the counter is synchronously reset to zero. If there are any outstanding bus requests, the aging counter will then begin counting immediately.

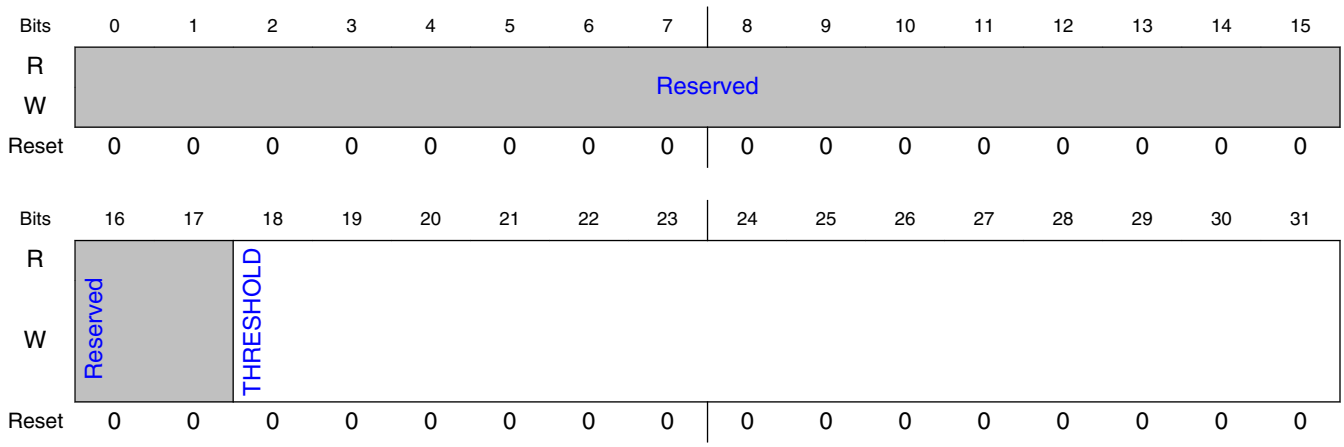
The AGE_CNT_THRESH is compared against the value of the aging counter during each clock cycle of the current transaction. If AGE_CNT_THRESH is equal to zero, an elevated priority is always chosen. If the aging counter is less than the AGE_CNT_THRESH value, default (low) priority is selected. If the aging counter is greater than or equal to the AGE_CNT_THRESH value and USB_PRI_CTRL[PRI_LVL] = 1, an elevated priority is chosen.

The setting of AGE_CNT_THRESH is highly dependent on both the mix of other controllers operating on the system bus as well as the kind of traffic moving through the USB controller. A recommended approach is first to try leaving the aging mechanism disabled and see if the USB meets performance requirements. If USB performance does not meet application requirements, try the following setting :

- Set AGE_CNT_THRESH to 80.

Raising AGE_CNT_THRESH benefits the other controllers on the system bus by reducing the frequency that this USB controller raises its priority to the arbiter.

32.3.35.3 Diagram



32.3.35.4 Fields

Field	Function
0-17 —	- Reserved, should be cleared
18-31 THRESHOLD	THRESHOLD Aging counter threshold value.

32.3.36 Priority Control (PRI_CTRL)

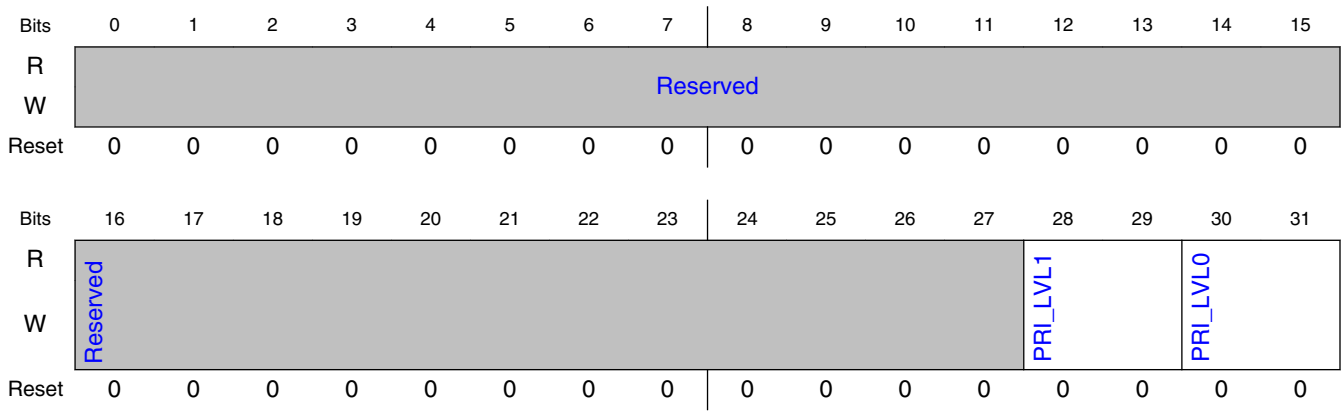
32.3.36.1 Offset

Register	Offset
PRI_CTRL	40Ch

32.3.36.2 Function

This register uses big-endian byte ordering and is not defined in the EHCI specification. The priority control (PRI_CTRL) register sets the priority level for each of two priority states. The priority state is determined by the value programmed in the AGE_CNT_THRESH register and the number of platform cycles that a particular transaction takes to complete .

32.3.36.3 Diagram



32.3.36.4 Fields

Field	Function
0-27 —	- Reserved, should be cleared
28-29 PRI_LVL1	PRI_LVL1 Priority level for priority state 1. The highest priority is 2'h3 and the lowest priority is 2'b0.
30-31 PRI_LVL0	PRI_LVL0 Priority level for priority state 0. The highest priority is 2'h3 and the lowest priority is 2'b0.

32.3.37 System Interface Control (SI_CTRL)

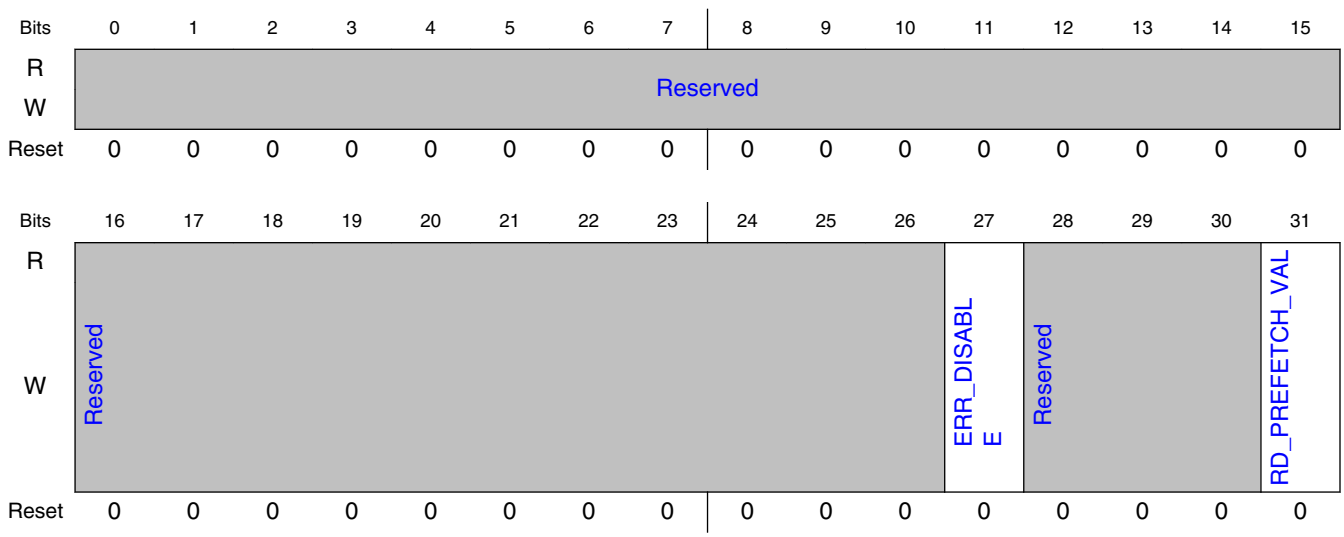
32.3.37.1 Offset

Register	Offset
SI_CTRL	410h

32.3.37.2 Function

This register uses big-endian byte ordering and is not defined in the EHCI specification. The system interface control register (SI_CTRL) controls various functions pertaining to the internal system interface.

32.3.37.3 Diagram



32.3.37.4 Fields

Field	Function
0-26 —	- Reserved, should be cleared
27 ERR_DISABLE	ERR_DISABLE When this bit is set, it causes the controller to ignore system bus errors. If cleared the controller responds according to the values set in USBSTS[SEI] and USBINT[SEE]. 0b - enable

Table continues on the next page...

Field	Function
	1b - disable
28-30 —	- Reserved, should be cleared
31 RD_PREFETCH_VAL	RD_PREFETCH_VAL Selects whether 32 bytes or 64 bytes are fetched during burst read transactions at the system interface. When this input is ZERO, 64 bytes are fetched, and when it is ONE, 32 bytes are fetched. **NOTE:** 32 byte fetch mode is not supported, this value must be set to 0. 0b - 64-byte fetch 1b - 32-byte fetch

32.3.38 Control (CONTROL)

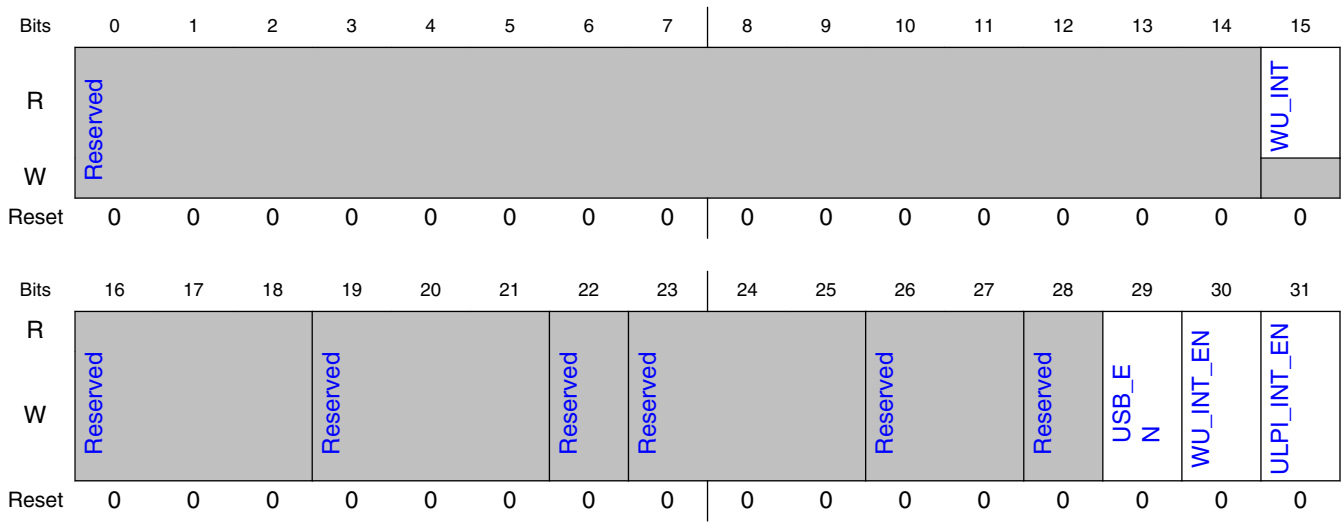
32.3.38.1 Offset

Register	Offset
CONTROL	500h

32.3.38.2 Function

This register uses big-endian byte ordering and is not defined in the EHCI specification. The USB general purpose (CONTROL) register contains the general-purpose IP control register outputs.

32.3.38.3 Diagram



32.3.38.4 Fields

Field	Function
0-14 —	- Reserved
15 WU_INT	WU_INT Reflects the state of the wake up interrupt. The wake up interrupt signal is asserted when a wake-up event occurs while in a low-power suspend state. If WU_INT_EN is set, this WU_INT signal generates an interrupt to the system to indicate wake up servicing is required. WU_INT will remain set until the USB controller is exited from the low power by clearing the PORTSC[PHCD] bit. 0b - Normal operation or Low Power mode waiting for wakeup event 1b - Low power wakeup event has occurred
16-18 —	- Reserved
19-21 —	- Reserved
22 —	- Reserved
23-25 —	-
26-27 —	- Reserved
28 —	- Reserved

Table continues on the next page...

Field	Function
29 USB_EN	USB_EN ULPI mode: In safe mode, all USB interface signals are put into input mode or driven inactive, except for SUSPEND_STP, which is driven high. Also, the input signal DIR is forced to appear high to the controller. This prevents any start-up problems that otherwise could occur if the PHY and the controller take significantly different times to complete power-on reset. 0b - Safe mode 1b - Normal operation
30 WU_INT_EN	WU_INT_EN This bit is used to mask/unmask the system wakeup interrupt signal **NOTE:** PORTSC[PHCD] bit must be set for the system wakeup interrupt generation. 0b - System wakeup interrupt disabled 1b - System wakeup interrupt enabled
31 ULPI_INT_EN	ULPI_INT_EN Used to enable the ULPI low power wakeup interrupt from the PHY when the PHY is in low power mode only. **NOTE:** PORTSC[PHCD] bit must be set. 0b - ULPI low power wakeup interrupt disabled 1b - ULPI low power wakeup interrupt enabled

32.4 Functional description

The USB DR module can be broken down into functional sub-blocks, which are described below.

32.4.1 System interface

The system interface block contains all the control and status registers that allow a processor to interface to the USB DR module.

These registers allow the processor to control the configuration of the module, ascertain the capabilities of the module, and control the module's operation. It also has registers to control snoopability and priority of the DMA interface.

32.4.2 DMA engine

The module contains a local DMA engine.

The DMA engine interfaces internally to the system memory bus. It is responsible for moving all of the data to be transferred over the USB between the module and buffers in system memory.

Like the system interface block, the DMA engine block uses a simple synchronous bus signaling protocol that eases connections to a number of different standard buses.

The DMA controller must access both control information and packet data from system memory. The control information is contained in link list-based queue structures. The DMA controller has state machines that are able to parse data structures defined in the EHCI specification. In host mode, the data structures are EHCI compliant and represent queues of transfers to be performed by the host controller, including the split-transaction requests that allow an EHCI controller to direct packets to FS and LS devices. In device mode, the data structures are designed to be similar to those in the EHCI specification and are used to allow device responses to be queued for each of the active pipes in the device.

32.4.3 FIFO RAM controller

The FIFO RAM controller is used for context information and to control FIFOs between the protocol engine and the DMA controller.

These FIFOs decouple the system processor/memory bus requests from the extremely tight timing required by USB.

The use of the FIFO buffers differs between host and device mode operation. In host mode, a single data channel is maintained in each direction through the buffer memory. In device mode, multiple FIFO channels are maintained for each of the active Endpoints in the system.

In host mode, the USB DR module uses a 512-byte Tx buffer and a 512-byte Rx buffer. Device operation uses a single 512-byte Rx buffer and a 512-byte Tx buffer for each Endpoint. The 512-byte buffers allow the module to buffer a complete HS bulk packet.

32.4.4 PHY interface

The ULPI interface connects to an external PHY. The USB DR module interfaces to any ULPI-compatible PHY.

The primary function of the port controller block is to isolate the rest of the module from the transceiver, and to move all of the transceiver signaling into the primary clock domain of the module.

This allows the module to run synchronously with the system processor and its associated resources.

Due to pincount limitations the module only supports certain combinations of PHY interfaces and USB functionality. Refer to the table below for more information.

Table 32-3. Supported PHY interfaces

PHY	Function
ULPI	Host/Device

32.5 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 section 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) support for the host controller interface. The asynchronous list is the root for all the bulk and control transfer type support. 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 with 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

Software must ensure that no interface data structure reachable by the EHCI host controller spans a 4K-page boundary.

The data structures defined in this section are (from the host controller's perspective) a mix of read-only and read/writable fields. The host controller must preserve the read-only fields on all data structure writes.

32.5.1 Periodic frame list

The figure below shows the organization of the periodic schedule. This schedule is for all periodic transfers (isochronous and interrupt).

The periodic schedule is referenced from the operational registers space using the PERIODICLISTBASE address register and the FRINDEX register. The periodic schedule is based on an array of pointers called the periodic frame list. The PERIODICLISTBASE address register is combined with the FRINDEX register to produce a memory pointer into the frame list. The periodic frame list implements a sliding window of work over time.

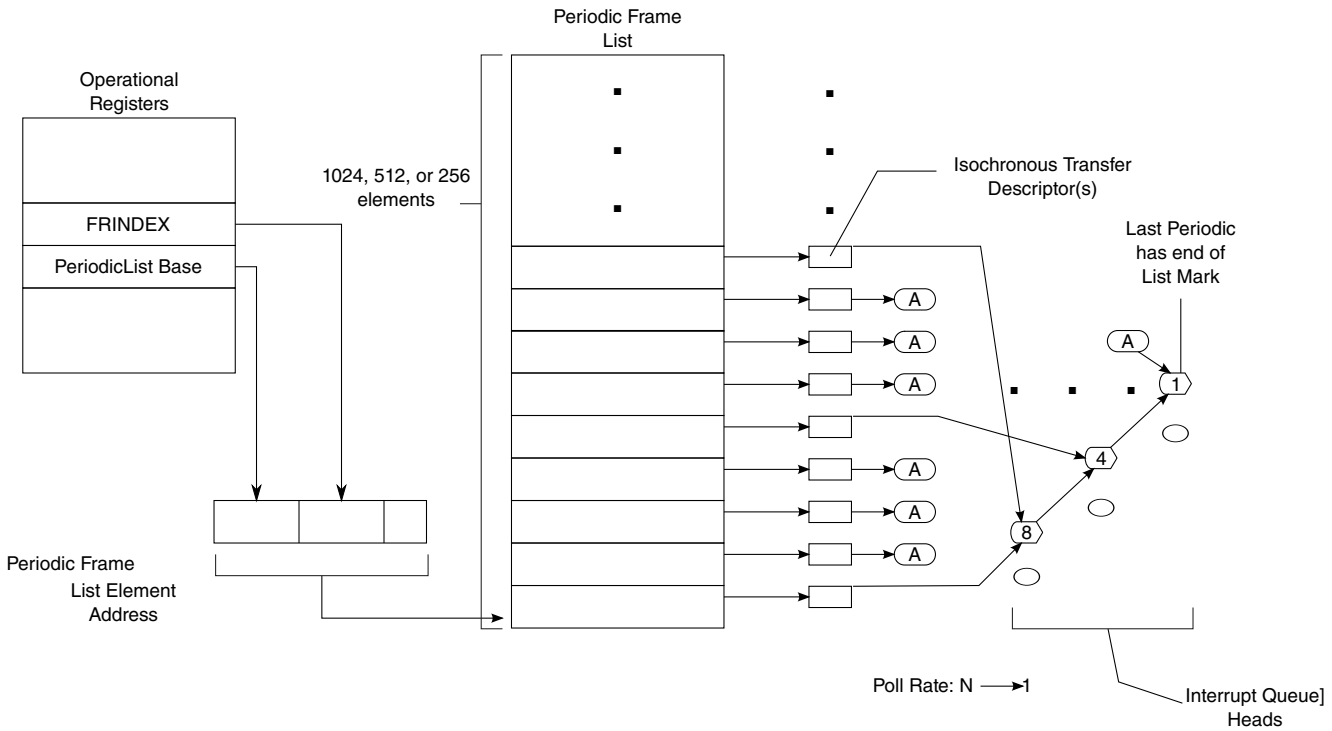


Figure 32-2. Periodic schedule organization

The periodic frame list is a 4K-page aligned array of Frame List Link pointers. The length of the frame list is programmable. The programmability of the periodic frame list is exported to system software through the HCCPARAMS register. The length can be selected by system software as one of 8, 16, 32, 64, 128, 256, 512 or 1024 elements. 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 USBCMD register.

Frame list link pointers direct the host controller to the first work item in the frame's periodic schedule for the current microframe. The link pointers are aligned on DWord boundaries within the frame list. The table below shows the format for the frame list link pointer.

Table 32-4. Frame list link pointer format

31																										5	4	3	2	1	0
Frame List Link Pointer																											00		Typ		T

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 in 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, the host controller never uses the value of the frame list pointer as a physical memory pointer. The Typ field indicates the exact type of data structure being referenced by this pointer. The value encodings for the Typ field are given in the table below.

Table 32-5. Typ field encodings

Typ	Description
00	Isochronous transfer descriptor
01	Queue head
10	Split transaction isochronous transfer descriptor
11	Frame span traversal node

32.5.2 Asynchronous list queue head pointer

The asynchronous transfer list (based at the ASYNCLISTADDR register) is where all 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. The figure below shows the asynchronous schedule organization.

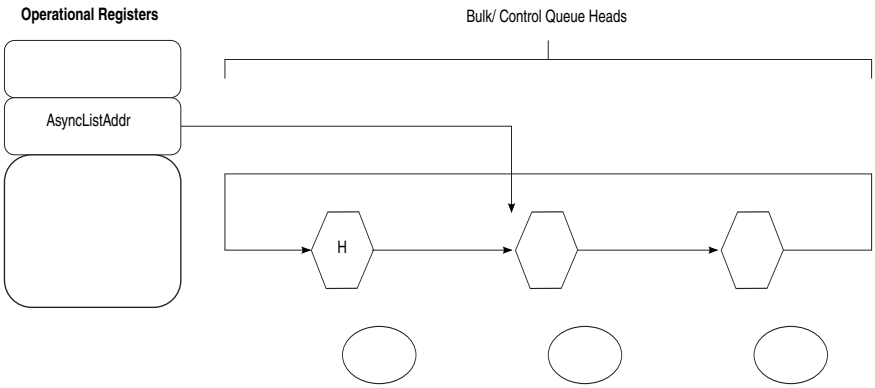


Figure 32-3. Asynchronous schedule organization

The asynchronous list is a simple circular list of queue heads. The 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.

32.5.3 Isochronous (high-speed) transfer descriptor (iTD)

The table below shows the format of an isochronous transfer descriptor.

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 32-6. 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	offset
Next link pointer																											00		Typ		T	0x00
Status[1]				Transaction 0 Length[1]												ioc	PG[2]			Transaction 0 Offset[2]													0x04
Status[1]				Transaction 1 length[1]												ioc	PG[2]			Transaction 1 offset[2]													0x08
Status[1]				Transaction 2 length[1]												ioc	PG[2]			Transaction 2 offset[2]													0x0C
Status[1]				Transaction 3 length[1]												ioc	PG[2]			Transaction 3 offset[2]													0x10
Status[1]				Transaction 4 length[1]												ioc	PG[2]			Transaction 4 offset[2]													0x14
Status[1]				Transaction 5 length[1]												ioc	PG[2]			Transaction 5 offset[2]													0x18
Status[1]				Transaction 6 length[1]												ioc	PG[2]			Transaction 6 offset[2]													0x1C
Status[1]				Transaction 7 length[1]												ioc	PG[2]			Transaction 7 offset[2]													0x20
Buffer pointer (page 0)																				EndPt				R	Device address							0x24
Buffer pointer (page 1)																				I/O	Maximum packet size											0x28
Buffer pointer (Page 2)																				Reserved											Mult	0x2C
Buffer pointer (page 3)																				Reserved												0x30
Buffer pointer (page 4)																				Reserved												0x34

Table continues on the next page...

Table 32-6. Isochronous transaction descriptor (iTD) (continued)

Buffer pointer (page 5)	Reserved	0x38
Buffer pointer (page 6)	Reserved	0x3C

1. Host controller read/write; all others read-only.
2. These fields may be modified by the host controller if the I/O field indicates an OUT.

32.5.3.1 Next link pointer-iTD

The first DWord of an iTD is a pointer to the next schedule data structure, as shown in the table below.

Table 32-7. Next schedule element pointer

Bits	Name	Description
31-5	Link Pointer	Correspond to memory address signals [31:5], respectively. This field points to another isochronous transaction descriptor (iTD/siTD) or queue head (QH).
4-3	-	Reserved, should be cleared. These bits are reserved and their value has no effect on operation. Software should initialize this field to zero.
2-1	Typ	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: 00 iTD (isochronous transfer descriptor) 01 QH (queue head) 10 siTD (split transaction isochronous transfer descriptor) 11 FSTN (frame span traversal node)
0	T	Terminate 1 Link Pointer field is not valid. 0 Link Pointer field is valid.

32.5.3.2 iTD transaction status and control list

DWords 1-8 constitute eight slots of transaction control and status.

Each transaction description includes the following:

- Status results field
- Transaction length (bytes to send for OUT transactions and bytes received for IN transactions).
- Buffer offset. The PG and Transaction *n* 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 table below shows the iTD transaction status and control fields.

Table 32-8. iTD transaction status and control

Bits	Name	Description
31-28	Status	Records the status of the transaction executed by the host controller for this slot. This field is a bit vector with the following encoding: 31 Active. Set 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 clears this bit indicating that a transaction for this element should not be executed when it is next encountered in the schedule. 30 Data buffer error. Set 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 (underrun). If an overrun condition occurs, no action is necessary. 29 Babble detected. Set by the host controller during status update when" babble" is detected during the transaction generated by this descriptor. 28 Transaction error (XactErr). Set by the host controller during status update in the case where the host did not receive a valid response from the device (Time-out, CRC, Bad PID, and so on). This bit may only be set for isochronous IN transactions.
27-16	Transaction *n* Length	For an OUT, this field is the number of data bytes the host controller will send 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 (for example, 0 zero length data, 1 one byte, 2 two bytes, and so on). The maximum value this field may contain is 0xC00 (3072).
15	ioc	Interrupt on complete. If this bit is set, it specifies that when this transaction completes, the host controller should issue an interrupt at the next interrupt threshold.
14-12	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 *n* 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.

32.5.3.3 iTD buffer page pointer list (plus)

DWords 9-15 of an isochronous transaction descriptor are nominally page pointers (4K 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) x 1024 (maximum packet size) x 8 (transaction records) = 24 576 bytes to be moved with this data structure, regardless of the alignment offset of the first page.

Since 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 following tables describe buffer pointer page *n*.

Table 32-9. Buffer pointer page 0 (Plus)

Bits	Name	Description
31-12	Buffer pointer (Page 0)	A 4K-aligned pointer to physical memory. Corresponds to memory address bits 31-12.
11-8	EndPt	Selects the particular Endpoint number on the device serving as the data source or sink.
7	-	Reserved, should be cleared. 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 32-10. iTD buffer pointer page 1 (plus)

Bits	Name	Description
31-12	Buffer Pointer (Page 1)	This is a 4K aligned pointer to physical memory. Corresponds to memory address bits 31-12.
11	I/O	Direction (I/O). This field encodes whether the high-speed transaction should use an IN or OUT PID. 0 OUT 1 IN
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 (for example, per microframe). 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 (0x400). Any value larger yields undefined results.

Table 32-11. Buffer pointer page 2 (plus)

Bits	Name	Description
31-12	Buffer pointer (page 2)	This is a 4K-aligned pointer to physical memory. Corresponds to memory address bits 31-12.
11-2	-	Reserved, should be cleared. This bit reserved for future use and should be cleared.
1-0	Mult	Indicates to the host controller the number of transactions that should be executed per transaction description (for example, per microframe). 00 Reserved, should be cleared. A zero in this field yields undefined results. 01 One transaction to be issued for this Endpoint per microframe 10 Two transactions to be issued for this Endpoint per microframe 11 Three transactions to be issued for this Endpoint per microframe

Table 32-12. Buffer pointer page 3-6

Bits	Name	Description
31-12	Buffer Pointer	This is a 4K aligned pointer to physical memory. Corresponds to memory address bits 31-12.
11-0	-	Reserved, should be cleared. These bits reserved for future use and should be cleared.

32.5.4 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 table below shows the split-transaction isochronous transfer descriptor (siTD).

Table 32-13. Split-transaction isochronous transaction descriptor (siTD)

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	offset
Next Link Pointer																											00		Typ		T	0x00
I/O	Port Number					0	Hub Address									0000			EndPt			0	Device Address									0x04
0000_0000_0000_00000																μFrame C-mask						μFrame S-mask										0x08
ioc	P[1]	0000					Total Bytes to Transfer[1]									μFrame C-prog-mask[1]						Status[1]										0x0C
Buffer Pointer (Page 0)																			Current Offset[1]													0x10
Buffer Pointer (Page 1)																			000_0000								TP[1]		T-count[1]			0x14
Back Pointer																											0000				T	0x18

1. Host controller read/write; all others read-only.

32.5.4.1 Next link pointer-siTD

DWord0 of a siTD is a pointer to the next schedule data structure.

The table below describes the next link pointer fields.

Table 32-14. Next link pointer

Bits	Name	Description
31-5	Next Link Pointer	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, should be cleared. These bits must be written as zeros.

Table continues on the next page...

Table 32-14. Next link pointer (continued)

Bits	Name	Description
2-1	Typ	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: 00 iTD (isochronous transfer descriptor) 01 QH (queue head) 10 siTD (split transaction isochronous transfer descriptor 11 FSTN (frame span traversal node)
0	T	Terminate. 0 Link pointer is valid. 1 Link pointer field is not valid.

32.5.4.2 siTD Endpoint capabilities/characteristics

DWords 1 and 2 specify static information about the full-speed Endpoint, the addressing of the parent Companion Controller, and microframe scheduling control.

The table below describes the Endpoint and transaction translator characteristics.

Table 32-15. Endpoint and transaction translator characteristics

Bits	Name	Description
31	I/O	Direction (I/O). This field encodes whether the full-speed transaction should be an IN or OUT. 0 OUT 1 IN
30-24	Port Number	This field is the port number of the recipient transaction translator.
23	-	Reserved, should be cleared. Bit reserved and should be cleared.
22-16	Hub Address	This field holds the device address of the companion controllers' hub.
15-12	-	Reserved, should be cleared. Field reserved and should be cleared.
11-8	EndPt	Endpoint Number. Selects the particular Endpoint number on the device serving as the data source or sink.
7	-	Reserved, should be cleared. Bit is reserved for future use. It should be cleared.
6-0	Device Address	Selects the specific device serving as the data source or sink.

The table below describes the microframe schedule control.

Table 32-16. Microframe schedule control

Bits	Name	Description
31-16	-	Reserved, should be cleared. This field reserved for future use. It should be cleared.

Table continues on the next page...

Table 32-16. Microframe schedule control (continued)

Bits	Name	Description
15-8	µFrame C-mask	Split completion mask. This field (along with the Active and SplitX- state fields in the status byte) is used to determine during which microframes 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 µFrame C-Mask field is a one, this siTD is a candidate for transaction execution. There may be more than one bit in this mask set.
7-0	µFrame S-mask	Split start mask. This field (along with the Active and SplitX-state fields in the Status byte) is used to determine during which microframes 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 µFrame 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.

32.5.4.3 siTD transfer state

DWords 3-6 manage the state of the transfer, as described in the table below.

Table 32-17. siTD transfer status and control

<table>
<tr><th>Bits</th><th>Name</th><th colspan="2">Description</th></tr>
<tr><td>31</td><td>ioc</td><td colspan="2">Interrupt on complete
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 will assert a hardware interrupt at the next interrupt threshold.</td></tr>
<tr><td>30</td><td>P</td><td colspan="2">Page select. Indicates which data page pointer should be concatenated with the CurrentOffset field to construct a data buffer pointer
0 Selects Page 0 pointer
1 Selects Page 1 pointer
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).</td></tr>
<tr><td>29-26</td><td>-</td><td colspan="2">Reserved, should be cleared. This field reserved for future use and should be cleared.</td></tr>
<tr><td>25-16</td><td>Total Bytes to Transfer</td><td colspan="2">This field is initialized by software to the total number of bytes expected in this transfer. Maximum value is 1023 (3FFh)</td></tr>
<tr><td>15-8</td><td>µFrame C-prog-mask</td><td colspan="2">Split complete progress mask. This field is used by the host controller to record which split-completes have been executed.</td></tr>
<tr><td>7-0</td><td>Status</td><td colspan="2">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:</td></tr>
<tr><td></td><td></td><td>Status Bits</td><td>Definition</td></tr>
<tr><td></td><td></td><td>7</td><td>Active. Set by software to enable the execution of an isochronous split transaction by the host controller.</td></tr>
<tr><td></td><td></td><td>6</td><td>ERR. Set by the host controller when an ERR response is received from the companion controller.</td></tr>
</table>

Table continues on the next page...

Table 32-17. siTD transfer status and control (continued)

Bits	Name	Description	
		5	Data buffer error. Set 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 will transmit an incorrect CRC (thus invalidating the data at the Endpoint). If an overrun condition occurs, no action is necessary.
		4	Babble detected. Set by the host controller during status update when" babble" is detected during the transaction generated by this descriptor.
		3	Transaction error (XactErr). Set by the host controller during status update in the case where the host did not receive a valid response from the device (Time-out, CRC, Bad PID, and so on). This bit will only be set for IN transactions.
		2	Missed microframe. 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: 0 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. 1 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, should be cleared. Bit reserved for future use and should be cleared.

32.5.4.4 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 4K (page) aligned buffer pointers. The least-significant 12 bits of each DWord are used as additional transfer state.

The table below describes the siTD buffer pointer page 0.

Table 32-18. siTD buffer pointer page 0 (plus)

Bits	Name	Description
31-12	Buffer Pointer (Page 0)	Bits 31-12 are 4K page-aligned, physical memory addresses. These bits correspond to physical address bits 31-12 respectively. The field P specifies the current active pointer
11-0	Current Offset	The 12 least-significant bits of the Page 0 pointer is 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).

The table below describes the siTD buffer pointer page 1.

Table 32-19. siTD buffer pointer page 1 (plus)

Bits	Name	Description
31-12	Buffer Pointer (Page 1)	Bits 31-12 are 4K page-aligned, physical memory addresses. These bits correspond to physical address bits 31-12 respectively. The field P specifies the current active pointer
11-5	-	Reserved, should be cleared.
4-3	TP	Transaction position. 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: 00 All. The entire full-speed transaction data payload is in this transaction (that is, less than or equal to 188 bytes). 01 Begin. This is the first data payload for a full-speed transaction that is greater than 188 bytes. 10 Mid. This is the middle payload for a full-speed OUT transaction that is larger than 188 bytes. 11 End. This is the last payload for a full-speed OUT transaction that was larger than 188 bytes.
2-0	T-Count	Transaction count. Software initializes this field with the number of OUT start-splits this transfer requires. Any value larger than 6 is undefined.

32.5.4.5 siTD back link pointer

DWord 6 of a siTD is simply another schedule link pointer.

This pointer is always zero, or references a siTD. This pointer cannot reference any other schedule data structure.

The table below describes the siTD back link pointer.

Table 32-20. siTD back link pointer

Bits	Name	Description
31-5	Back Pointer	A physical memory pointer to an siTD
4-1	-	Reserved, should be cleared. This field is reserved for future use. It should be cleared.
0	T	Terminate 0 siTD Back Pointer field is valid 1 siTD Back Pointer field is not valid

32.5.5 Queue element transfer descriptor (qTD)

This data structure is only used with a queue head.

This data structure is used for one or more USB transactions. This data structure is used to transfer up to 20,480 (5 x 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. This structure is 32 bytes (or one 32-byte cache line). This data structure must be physically contiguous.

The buffer associated with this transfer must be virtually contiguous. The buffer may start on any 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 table below shows the queue element transfer descriptors.

Table 32-21. Queue element transfer descriptor (qTD)

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	offset
Next qTD Pointer																											0000				T	0x00
Alternate Next qTD Pointer																											0000				T	0x04
dt[1]	Total Bytes to Transfer[1]															ioc	C_Page[1]			Cerr[1]		PID Code	Status[1]									0x08
Buffer Pointer (Page 0)																				Current Offset[1]												0x0C
Buffer Pointer (Page 1)																				0000_0000_0000												0x10
Buffer Pointer (Page 2)																				0000_0000_0000												0x14
Buffer Pointer (Page 3)																				0000_0000_0000												0x18
Buffer Pointer (Page 4)																				0000_0000_0000												0x1C

1. Host controller read/write; all others read-only.

Queue element transfer descriptors must be aligned on 32-byte boundaries.

32.5.5.1 Next qTD pointer

The first DWord of an element transfer descriptor is a pointer to another transfer element descriptor.

The table below describes the qTD next element transfer pointer.

Table 32-22. qTD next element transfer pointer (DWord 0)

Bits	Name	Description
31-5	Next qTD Pointer	This field contains the physical memory address of the next qTD to be processed and corresponds to memory address signals [31:5], respectively.
4-1	-	Reserved, should be cleared. These bits are reserved and their value has no effect on operation.

Table continues on the next page...

Table 32-22. qTD next element transfer pointer (DWord 0) (continued)

Bits	Name	Description
0	T	Terminate. Indicates to the host controller that there are no more valid entries in the queue. 0 Pointer is valid (points to a valid transfer element descriptor) 1 Pointer is invalid

32.5.5.2 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 client buffer 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 table below describes the alternate qTD next element transfer pointer.

Table 32-23. qTD alternate next element transfer pointer (DWord 1)

Bits	Name	Description
31-5	Alternate next qTD 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, should be cleared. These bits are reserved and their value has no effect on operation.
0	T	Terminate. Indicates to the host controller that there are no more valid entries in the queue. 0 Pointer is valid (points to a valid transfer element descriptor) 1 Pointer is invalid

32.5.5.3 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 that some of the field descriptions in the table below reference fields are defined in the queue head. See Queue head, for more information on these fields.

Table 32-24. qTD token (DWord 2)

Bits	Name	Description
31	dt	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.

Table continues on the next page...

Table 32-24. qTD token (DWord 2) (continued)

Bits	Name	Description	
30-16	Total bytes to transfer	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 x 4K (0x5000). 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 QH[Maximum Packet Length]. If software builds such a transfer descriptor for an OUT transfer, the last transaction will always be less than QH[Maximum Packet Length]. Although it is possible to create a transfer up to 20K this assumes the 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 (0x4000).	
15	ioc	Interrupt on complete. If this bit is set, the host controller should issue an interrupt at the next interrupt threshold when this qTD is completed.	
14-12	C_Page	Current range. This field is used as an index into the qTD buffer pointer list. Valid values are in the range 0x0 to 0x4. The host controller is not required to write this field back when the qTD is retired.	
11-10	Cerr	Error counter. 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 setup, 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 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 USBINTR[UEE] is set. If the host controller driver (HCD) software programs this field to zero during setup, the host controller will 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.	
		Error	**Decrement Counter**
		Transaction Error	Yes
		Data Buffer Error	No. Data buffer errors are host problems. They don't count against the device's retries. Note that 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.
		Stalled	No. Detection of babble or stall automatically halts the queue head. Thus, count is not decremented
		Babble Detected	No. Detection of babble or stall automatically halts the queue head. Thus, count is not decremented
		No Error	No. 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 (0b11) 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 0b00.
9-8	PID Code	This field is an encoding of the token, which should be used for transactions associated with this transfer descriptor. Encodings are: 00 OUT Token generates token (E1H) 01 IN Token generates token (69H)	

Table continues on the next page...

Table 32-24. qTD token (DWord 2) (continued)

Bits	Name	Description	
		10 SETUP Token generates token (2DH) (undefined if Endpoint is an Interrupt transfer type, for example. µFrame S-mask field in the queue head is non-zero.) 11 Reserved, should be cleared	
7-0	Status	This field is used by the host controller to communicate individual command execution states back to the host controller driver (HCD) software. This field contains the status of the last transaction performed on this qTD. The bit encodings are:	
		Bits	**Status Field Description**
		7	Active. Set by software to enable the execution of transactions by the host controller.
		6	Halted. Set 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 of the STALL handshake from the device during a transaction. Any time that a transaction results in the Halted bit being set, the Active bit is also cleared.
		5	Data buffer error. Set 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 will force a time-out 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 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. Since 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 by the host controller during status update in the case where the host did not receive a valid response from the device (time-out, CRC, bad PID). If the host controller sets this bit to a one, then it remains a one for the duration of the transfer.
		2	Missed microframe. 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: 0 Do start split. This value directs the host controller to issue a start split transaction to the Endpoint. 1 Do complete split. This value directs the host controller to issue a Complete split transaction to the Endpoint.

Table continues on the next page...

Table 32-24. qTD token (DWord 2) (continued)

Bits	Name	Description	
		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: 0 Do OUT. This value directs the host controller to issue an OUT PID to the Endpoint. 1 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 by the host controller whenever a periodic split-transaction receives an ERR handshake.

32.5.5.4 qTD buffer page pointer list

The last five DWords of a queue element transfer descriptor make up an array of physical memory address pointers.

These pointers reference the individual pages of a data buffer.

System software initializes the Current Offset field to the starting offset into the current page, where current page is selected with the value in the C_Page field.

The table below describes the qTD buffer pointer.

Table 32-25. qTD buffer pointer

Bits	Name	Description
31-12	Buffer pointer (page *n*)	Each element in the list is a 4K 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 4K 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 via the new buffer pointer.
11-0	Current offset (Page 0)/ - (pages 1-4)	Reserved in all pointers except the first one (that is, 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 zeros.

32.5.6 Queue head

The table below shows the queue head structure.

Table 32-26. Queue head 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	offset
Queue Head Horizontal Link Pointer																											00		Typ		T	0x00[1]
RL			C	Maximum Packet Length							H	dtc	EPS	EndPt							I	Device Address										0x04[1]
Mult		Port Number					Hub Addr				µFrame C-mask										µFrame S-mask											0x08
Current qTD Pointer[2]																											00000					0x0C
Next qTD Pointer[2]																											0000				T[2]	0x10[3,4]
Alternate Next qTD Pointer[2]																											NakCnt[2]				T[2]	0x14[3,4]
dt[2]	Total Bytes to Transfer[2]															ioc[2]	C_Page[2]					Cerr[2]		PID Code[2]			Status[2]						0x18[3,4]
Buffer Pointer (Page 0)[2]																						Current Offset[2]										0x1C[3,4]
Buffer Pointer (Page 1)[2]																						0000		C-prog-mask[2]								0x20[3,4]
Buffer Pointer (Page 2)[2]																						S-bytes[2]								FrameTag[2]		0x24[3,4]
Buffer Pointer (Page 3)[2]																						0000_0000_0000										0x28[3]
Buffer Pointer (Page 4)[2]																						0000_0000_0000										0x2C[3]

1. Offsets 0x00 through 0x07 contain the static Endpoint state.
2. Host controller read/write; all others read-only.
3. Offsets 0x10 through 0x2F contain the transfer overlay.
4. Offsets 0x10 through 0x27 contain the transfer results.

32.5.6.1 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 table below describes the queue head.

Table 32-27. Queue head DWord 0

Bits	Name	Description
31-5	QHLP	Queue head horizontal link pointer. 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, should be cleared. These bits must be written as zeros.
2-1	Typ	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.

Table continues on the next page...

Table 32-27. Queue head DWord 0 (continued)

Bits	Name	Description
		00 iTD (isochronous transfer descriptor) 01 QH (queue head) 10 siTD (split transaction isochronous transfer descriptor) 11 FSTN (frame span traversal node)
0	T	Terminate. 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.

32.5.6.2 Endpoint capabilities/characteristics

The second and third DWords of a queue head specify 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, which include addressing, maximum packet size, and Endpoint speed.
- Endpoint capabilities. These are adjustable parameters of the Endpoint. They affect how the Endpoint data stream is managed by the host controller.
- Split transaction characteristics. This data structure manages full- and low-speed data streams for bulk, control, and interrupt with split transactions to USB 2.0 Hub transaction translator. Additional fields exist for addressing the hub and scheduling the protocol transactions (for periodic).

The host controller must not modify the bits in this region.

The following tables describe the Endpoint characteristics.

Table 32-28. Endpoint characteristics: Queue head DWord 1

Bits	Name	Description
31-28	RL	Nak count reload. This field contains a value, which is used by the host controller to reload the Nak Counter field.
27	C	Control Endpoint flag. 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 a zero.

Table continues on the next page...

Table 32-28. Endpoint characteristics: Queue head DWord 1 (continued)

Bits	Name	Description
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	H	Head of reclamation list flag. This bit is set by system software to mark a queue head as being the head of the reclamation list.
14	dtc	Data toggle control (DTC). Specifies where the host controller should get the initial data toggle on an overlay transition. 0 Ignore DT bit from incoming qTD. Host controller preserves DT bit in the queue head. 1 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	EPS	Endpoint speed. This is the speed of the associated Endpoint. 00 Full-speed (12 Mbps) 01 Low-speed (1.5 Mbps) 10 High-speed (480 Mbps) 11 Reserved, should be cleared This field must not be modified by the host controller.
11-8	EndPt	Endpoint number. Selects the particular Endpoint number on the device serving as the data source or sink.
7	I	Inactivate on next transaction. This bit is used by system software to request that the host controller set the Active bit to zero. 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 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	Selects the specific device serving as the data source or sink.

Table 32-29. Endpoint capabilities: Queue head DWord 2

Bits	Name	Description
31-30	Mult	High-bandwidth pipe multiplier. 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). 00 Reserved, should be cleared. A zero in this field yields undefined results. 01 One transaction to be issued for this Endpoint per microframe 10 Two transactions to be issued for this Endpoint per microframe 11 Three transactions to be issued for this Endpoint per microframe
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	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 microframes the host controller should

Table continues on the next page...

Table 32-29. Endpoint capabilities: Queue head DWord 2 (continued)

Bits	Name	Description
		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 µFrame 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	µFrame S-mask	Interrupt schedule 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 µFrame 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.

32.5.6.3 Transfer overlay

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 never follows 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 table below describes the current qTD link pointer.

Table 32-30. Current qTD link pointer

Bits	Name	Description
31-5	Current qTD pointer	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, should be cleared. 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 an execution cache for the transfer.

The table below describes the host-controller rules for bits in overlay.

Table 32-31. Host-controller rules for bits in overlay (DWords 5, 6, 8, and 9)

DWord	QH Offset	Bits	Name	Description
5	0x14	4-1	NakCnt	Nak counter-RW. 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	0x18	31	dt	Data toggle. The Data toggle control controls whether the host controller preserves this bit when an overlay operation is performed.
6	0x18	15	ioc	Interrupt on complete. The ioc control bit is always inherited from the source qTD when the overlay operation is performed.
6	0x18	11-10	Cerr	Error counter. Copied from the qTD during the overlay and written back during queue advancement.
6	0x18	0	Status[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	0x20	7-0	C-prog-mask	Split-transaction complete-split progress. Initialized to zero during any overlay. This field is used to track the progress of an interrupt split-transaction.
9	0x24	11-5	S-bytes	Software must ensure that the S-bytes field in a qTD is zero before activating the qTD. Keeps track of the number of bytes sent or received during an IN or OUT split transaction.
9	0x24	4-0	FrameTag	Split-transaction frame tag. Initialized to zero during any overlay. This field is used to track the progress of an interrupt split-transaction.

32.5.7 Periodic frame span traversal node (FSTN)

The periodic frame span traversal node (FSTN) data structure, shown in the table below, is to be used only for managing full- and low-speed transactions that span a host-frame boundary.

Software must not use an FSTN in the asynchronous schedule. An FSTN in the asynchronous schedule results in undefined behavior.

Table 32-32. Frame span traversal node 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	offset
Normal path link pointer																											-		Typ		T	0x00
Back path link pointer																											-		Typ		T	0x04

NOTE

The host controller performs only read operations to the FSTN data structure.

32.5.7.1 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 table below describes the FSTN normal path pointer.

Table 32-33. FSTN normal path pointer

Bits	Name	Description
31-5	NPLP	Normal path link pointer. 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, should be cleared. These bits must be written as 0s.
2-1	Typ	Indicates to the host controller whether the item referenced is a iTD/siTD, QH, or FSTN. This allows the host controller to perform the proper type of processing on the item after it is fetched. 00 iTD (isochronous transfer descriptor) 01 QH (queue head) 10 siTD (split transaction isochronous transfer descriptor) 11 FSTN (frame span traversal node)
0	T	Terminate. 0 Link pointer is valid. 1 Link pointer field is not valid.

32.5.7.2 FSTN back path link pointer

The second DWord of an FSTN node contains a link pointer to a queue head.

If the T-bit in this pointer is a 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, then this FSTN is the Restore indicator. When the T-bit is a one, the host controller ignores the Typ field.

The table below describes the FSTN back path link pointer.

Table 32-34. FSTN back path link pointer

Bits	Name	Description
31-5	BPLP	Back path link pointer. Contains the address of a queue head. This field corresponds to memory address signals [31:5], respectively.
4-3	-	Reserved, should be cleared. These bits must be written as 0s.
2-1	Typ	Software must ensure this field is set to indicate the target data structure is a Queue Head (01). Any other value in this field yields undefined results.
0	T	Terminate. 0 Link pointer is valid (that is, the host controller may use bits 31-5 as a valid memory address). This value also indicates that this FSTN is a Save-Place indicator. 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.

32.6 Host operations

The general operational model for the USB DR module in host mode is defined by the Enhanced Host Controller Interface (EHCI) Specification. The EHCI specification describes the register-level interface for a host controller for the USB Revision 2.0. It includes a description of the hardware/software interface between system software and host controller hardware. Information concerning the initialization of the USB module is included in the following section; however, the full details of the EHCI specification are beyond the scope of this document.

32.6.1 Host controller initialization

After initial power-on or host controller reset (hardware or through USBCMD[RST]), all of the operational registers are at their default values.

To configure the external ULPI PHY, the following initialization sequence is required:

1. The UTMI PHY should remain disabled if the ULPI is being used.
3. Wait for 10 ms.

The user can proceed to the host controller initialization phase.

In order to initialize the USB DR module, software should perform the following steps:

1. Set the controller mode to host mode. Optionally set USBMODE[SDIS] (streaming disable)

NOTE

Transitioning from device mode to host mode requires a host controller reset before modifying USBMODE.

2. Program the PTS field of the PORTSC register if using a non-ULPI PHY.
3. Set CONTROL[USB_EN].
4. Write the appropriate value to the USBINTR register to enable the appropriate interrupts.
5. Write the base address of the periodic frame list to the PERIODICLIST BASE register. If there are no work items in the periodic schedule, all elements of the periodic frame list should have their T-Bits set.
6. Write the USBCMD register to set the desired interrupt threshold, frame list size (if applicable) and turn on the controller by setting the RS bit.

At this point, the USB DR module is up and running and the port registers begin reporting device connects. 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 high-speed ports, but the schedules have not yet been enabled. The EHCI host controller will not transmit SOFs to enabled Full- or Low-speed ports.

In order to communicate with devices via the asynchronous schedule, system software must write the ASYNDLISTADDR register with the address of a control or bulk queue head. Software must then enable the asynchronous schedule by writing a one to USBCMD[ASE]. In order to communicate with devices via the periodic schedule, system software must enable the periodic schedule by writing a one to USBCMD[PSE]. Note that the schedules can be turned on before the first port is reset (and enabled).

Any time the USBCMD register is written, system software must ensure the appropriate bits are preserved, depending on the intended operation.

32.6.2 Power port

The HCSPARAMS[PPC] bit indicates whether the USB 2.0 host controller has port power control.

When the PPC bit is set, the host controller supports port power switches. Each available switch has an output enable. PPE is controlled based on the state of the combination bits-PPC bit, EHCI Configured (CF)-bit and individual Port Power (PP) bit. The Configured

Flag and Port Power Control bits are always 1'b1 in Host Mode. The PPE always follows the state of Port Power (PP) bit that is, if PP is 0, PPE will be 0 and if PP is 1, PPE will be 1.

32.6.3 Reporting over-current

Host ports by definition are power providers on USB.

Whether the ports are considered high- or low-powered is a platform implementation issue. The EHCI PORTSC register has an over-current status and over-current change bit. The functionality of these bits is specified in the USB Specification Revision 2.0.

The over current detection and limiting logic resides outside the DR logic. The over-current condition effects the following bits in the PORTSC register on the EHCI port:

- Over-current active bit (OCA) is set. When the over-current condition goes away, the OCA will transition from a one to a zero.
- Over-current change bit (OCC) is set. On every transition of OCA, the controller will set OCC to a one. Software sets OCC to a zero by writing a one to this bit.
- Port enabled/disabled bit (PE) is cleared. When this change bit gets set, USBSTS[PCI] (the port change detect bit) is set.
- Port power (PP) bit may optionally be cleared. 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 OCC transitions from a zero to a one, the controller also sets USBSTS[PCI] to a one. In addition, if the Port Change Interrupt Enable bit, USBINTR[PCE], is a one, the controller issues an interrupt to the system. Refer to Table 32-35 for summary of behavior for over-current detection when the controller is halted (suspended from a device component point of view).

32.6.4 Suspend/resume

The 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 wakeup events are:

- Remote-wakeup enabled device asserts resume signaling. In similar kind to USB 2.0 hubs, when in host mode the host controller responds 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 port control bits in the PORTSC register.

Selective suspend is a feature supported by the PORTSC 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 bus, it should suspend the enabled port, then shut off the controller by setting the USBCMD[RS] to a zero.

When a wake event occurs the system will resume operation and system software must set the RS bit to a one and resume the suspended port.

32.6.4.1 Port suspend/resume

System software places the USB into suspend mode by writing a one into the appropriate PORTSC Suspend bit.

Software must only set the Suspend bit when the port is in the enabled state (Port Enabled bit is a one).

The host controller may evaluate the Suspend bit immediately or wait until a microframe 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 microframes 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 the suspended port by writing a one to PORTSC[FPR]. Software should not attempt to resume a port unless the port reports that it is in the suspended state. If system software sets PORTSC[FPR] 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 milliseconds after a port indicates that it is suspended (Suspend bit is a one) before initiating a port resume through PORTSC[FPR]. When PORTSC[FPR] is set, the host controller sends resume signaling down the port. System software times the duration of the resume (nominally 20 milliseconds) then clears PORTSC[FPR]. When the host controller receives the write to transition PORTSC[FPR] to zero, it completes the resume sequence as defined in the USB specification, and clears both PORTSC[FPR] and PORTSC[SUSP]. Software-initiated port resumes do not affect the port change detect bit (USBSTS[PCI]) nor do they cause an interrupt if USBINTR[PCE] (port change interrupt enable) is a one. 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 PORTSC[FPR] bit is set and USBSTS[PCI] is set. If USBINTR[PCE] is a one, the host controller issues a hardware interrupt.

System software observes the resume event on the port, delays a port resume time (nominally 20 milliseconds), then terminates the resume sequence by clearing PORTSC[FPR] in the port. The host controller receives the write of zero to PORTSC[FPR], terminates the resume sequence and clears PORTSC[FPR] and PORTSC[SUSP]. Software can determine that the port is enabled (not suspended) by sampling the PORTSC register and observing that the SUSP and FPR bits are zero. Software must ensure that the host controller is running (that is, USBSTS[HCH] is a zero), before terminating a resume by clearing the port's PORTSC[FPR] bit. If HCH is a one when PORTSC[FPR] is cleared, then SOFs will not occur down the enabled port and the device will return to suspend mode in a maximum of 10 milliseconds.

The table below summarizes the wake-up events. Whenever a resume event is detected, USBSTS[PCI] is set. If USBINTR[PCE] (port change interrupt enable) is a one, the host controller also generates an interrupt on the resume event. Software acknowledges the resume event interrupt by clearing the USBSTS[PCI].

Table 32-35. 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. PORTSC[FPR] is set. USBSTS[PCI] is set.	[1], [2]	[2]
Port is enabled, disabled or suspended, and the port's PORTSC[WKDS], is set. A disconnect is detected.	Depending on the initial port state, the PORTSC Connect (CCS) and Enable (PE) status bits are cleared, and the Connect Change status bit (CSC) is set. USBSTS[PCI] is set.	[1], [2]	[2]
Port is enabled, disabled or suspended, and the port's PORTSC[WKDS], is cleared. A disconnect is detected.	Depending on the initial port state, the PORTSC Connect (CCS) and Enable (PE) status bits are cleared, and the Connect Change status bit (CSC) is set. USBSTS[PCI] is set.	[1], [3]	[3]
Port is not connected and the port's PORTSC[WKCN] bit is a one. A connect is detected.	PORTSC Connect Status (CCS) and Connect Status Change (CSC) bits are set. USBSTS[PCI] is set.	[1], [2]	[2]
Port is not connected and the port's PORTSC[WKCN] bit is a zero. A connect is detected.	PORTSC Connect Status (CCS) and Connect Status Change (CSC) bits are set. USBSTS[PCI] is set.	[1], [3]	[3]
Port is connected and the port's PORTSC[WKOC] bit is a one. An over-current condition occurs.	PORTSC Over-current Active (OCA), Over-current Change (OCC) bits are set. If Port Enable/Disable bit (PE) is a one, it is cleared. USBSTS[PCI] is set	[1], [2]	[2]
Port is connected and the port's PORTSC[WKOC] bit is a zero. An over-current condition occurs.	PORTSC Over-current Active (OCA), Over-current Change (OCC) bits are set. If Port Enable/Disable bit (PE) is a one, it is cleared. USBSTS[PCI] is set.	[1], [3]	[3]

[1] Hardware interrupt issued if USBINTR[PCE] (port change interrupt enable) is set.

Table 32-35. Behavior during wake-up events

Port status and signaling type	Signaled port response	Device state	
		D0	Not D0
[2] PME# asserted if enabled (Note: PME Status must always be set).			
[3] PME# not asserted.			

32.6.5 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. The 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 microframe, if the periodic schedule is enabled (see Periodic schedule) then the host controller must execute from the periodic schedule before executing from the asynchronous schedule. It will only execute 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 PERIODICLISTBASE and the FRINDEX registers (see the figure below). 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. When the host controller encounters a T-Bit set 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. Once this transition is made, the host controller executes from the asynchronous schedule until the end of the microframe.

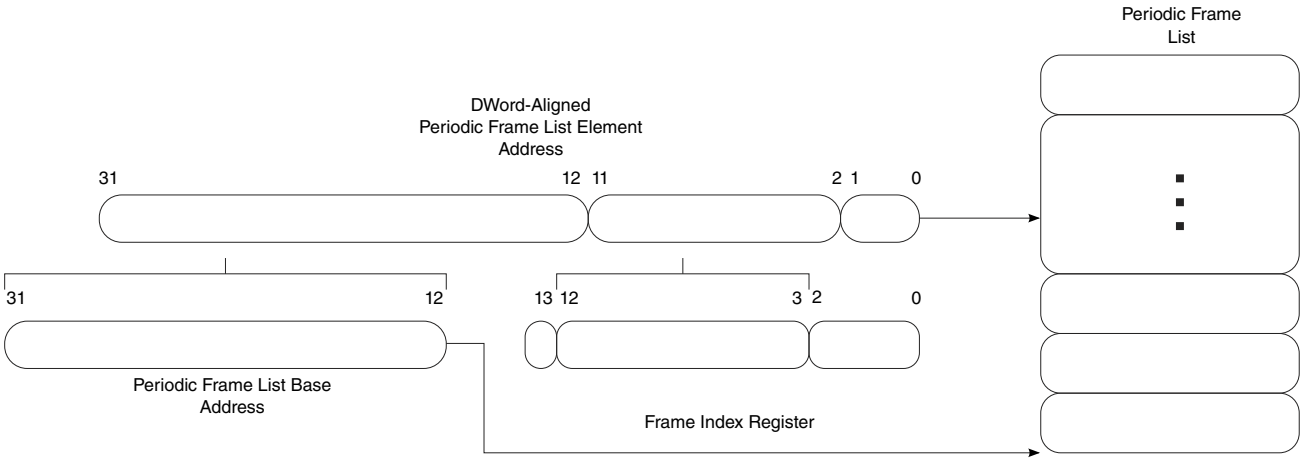


Figure 32-4. Derivation of pointer into frame list array

When the host controller determines that it is time to execute from the asynchronous list, it uses the operational register ASYNCLISTADDR to access the asynchronous schedule, as shown in the figure below.

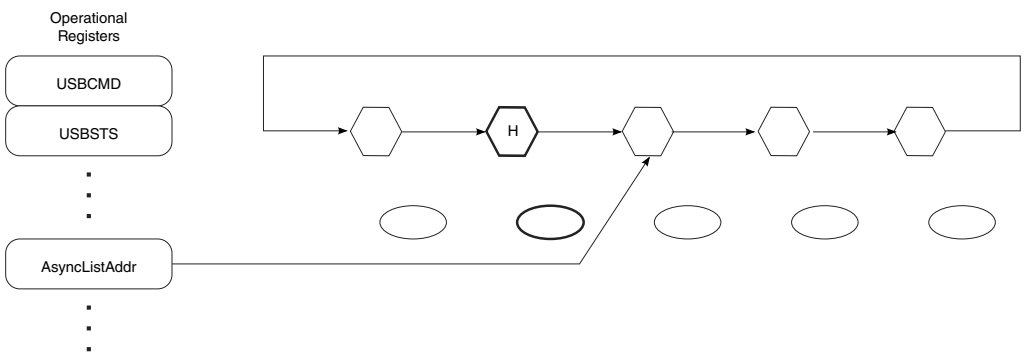


Figure 32-5. General format of asynchronous schedule list

The 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 ASYNCLISTADDR register. Software must set queue head horizontal pointer T-bits to a zero for queue heads in the asynchronous schedule.

32.6.6 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(es) 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 via a microframe pipeline (see start- (SS) and complete- (CS) splits illustrated in the figure below). 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.

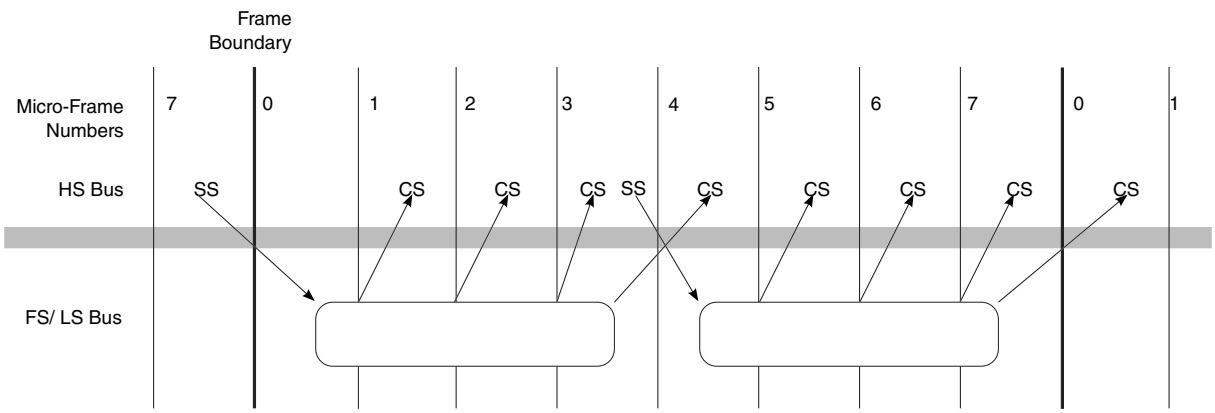


Figure 32-6. Frame boundary relationship between HS bus and FS/LS bus

The simple projection, as the figure above 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 a one microframe 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 via the Frame List Index Register (FRINDEX). Bits FRINDEX[2-0], represent the microframe number. The SOF value is coupled to the value of FRINDEX[13-3]. Both FRINDEX[13-3] and the SOF value are incremented based on FRINDEX[2-0]. It is required that the SOF value be delayed from the FRINDEX value by one microframe. The one microframe delay yields a host controller periodic schedule and bus frame boundary relationship as illustrated in the figure below. 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 figure below 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-millisecond boundaries is called H-Frames. The high-speed bus's view of the 1-millisecond boundaries is called B-Frames.

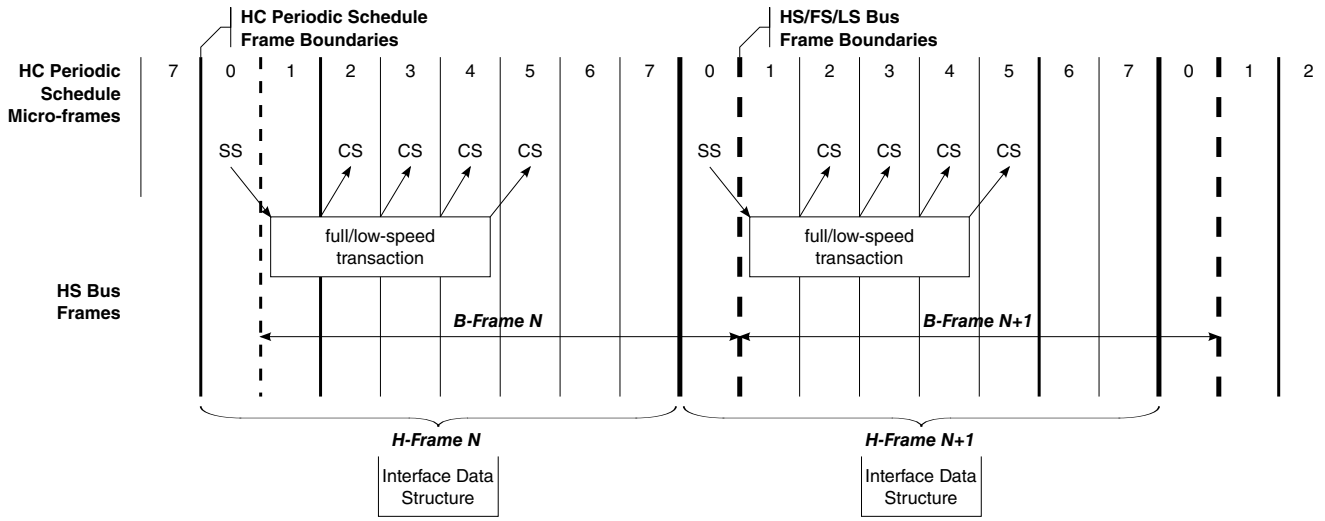


Figure 32-7. Relationship of periodic schedule frame boundaries to bus frame boundaries

H-Frame boundaries for the host controller correspond to increments of FRINDEX[13-3]. Microframe numbers for the H-Frame are tracked by FRINDEX[2-0]. B-Frame boundaries are visible on the high-speed bus via changes in the SOF token's frame number. Microframe numbers on the high-speed bus are only derived from the SOF token's frame number (that is, the high-speed bus will see 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 microframe 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_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 microframe count. The table below illustrates the required relationship between the value of FRINDEX and the value of SOFV. 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_FRINDEX provides the requirements that software should adhere when writing a new value in FRINDEX.

Table 32-36. Operation of FRINDEX and SOFV (SOF value register)

Current			Next		
FRINDEX[13-3]	SOFV	FRINDEX[2-0]	FRINDEX[13-3]	SOFV	FRINDEX[2-0]
N	N	111	N+1	N	000
N+1	N	000	N+1	N+1	001
N+1	N+1	001	N+1	N+1	010
N+1	N+1	010	N+1	N+1	011
N+1	N+1	011	N+1	N+1	100
N+1	N+1	100	N+1	N+1	101
N+1	N+1	101	N+1	N+1	110
N+1	N+1	110	N+1	N+1	111

32.6.7 Periodic schedule

The periodic schedule traversal is enabled or disabled through USBCMD[PSE] (periodic schedule enable).

If USBCMD[PSE] is cleared, then the host controller simply does not try to access the periodic frame list via the PERIODICLISTBASE register. Likewise, when USBCMD[PSE] is a one, then the host controller does use the PERIODICLISTBASE register to traverse the periodic schedule. The host controller will not react to modifications to USBCMD[PSE] immediately. In order to eliminate conflicts with split transactions, the host controller evaluates USBCMD[PSE] 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 0b000 microframe. These work items must be removed from the schedule before USBCMD[PSE] is cleared. USBSTS[PS] (periodic schedule status) indicates status of the periodic schedule. System software enables (or disables) the periodic schedule by setting (or clearing) USBCMD[PSE]. Software then can poll USBSTS[PS] to determine when the periodic schedule has made the desired transition. Software must not modify USBCMD[PSE] unless the value of USBCMD[PSE] equals that of USBSTS[PS].

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 USB. The figure below 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.

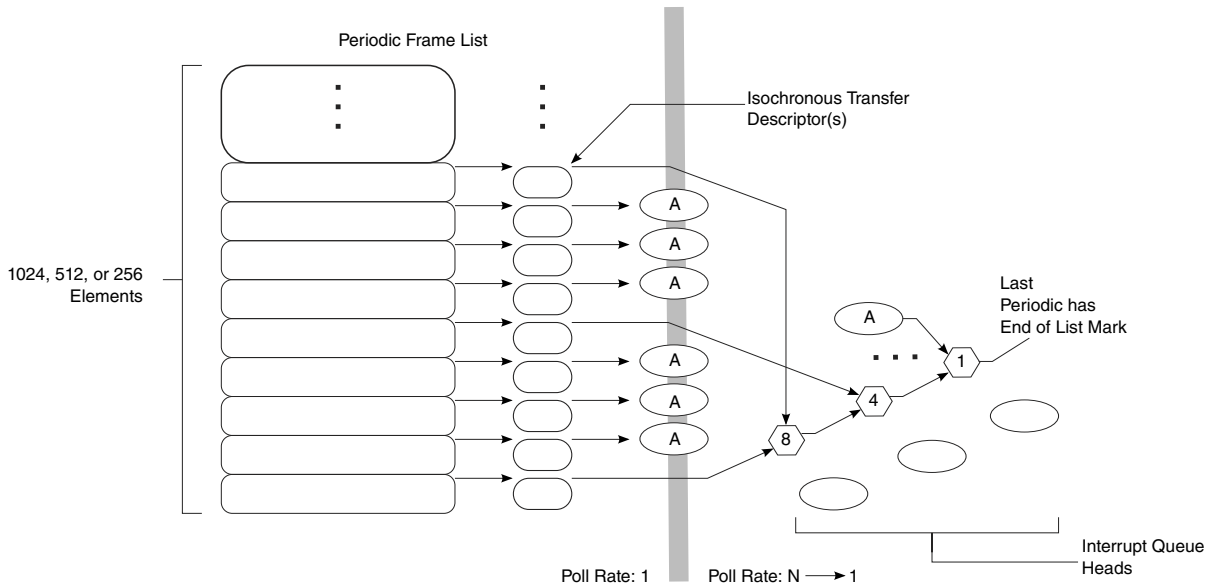


Figure 32-8. Example periodic schedule

32.6.8 Managing isochronous transfers using iTDs

The structure of an iTD is presented in isochronous (high-speed) transfer descriptor (iTD).

There are four distinct sections to an iTD:

- Next link pointer
  - This is the first field.
  - This field is for schedule linkage purposes only.
- Transaction description array
  - This is an eight-element array.
  - Each element represents control and status information for one microframe's worth of transactions for a single high-speed isochronous Endpoint.
- Buffer page pointer array
  - This is a 7-element array of physical memory pointers to data buffers.
  - These are 4K aligned pointers to physical memory.
- Endpoint capabilities

- This area utilizes the unused low-order 12 bits of the buffer page pointer array.
- Its fields are used across all transactions executed for this iTD, including Endpoint addressing, transfer direction, maximum packet size and high-bandwidth multiplier.

32.6.8.1 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. Each iTD can span 8 microframes worth of transactions. When the host controller fetches an iTD, it uses FRINDEX register bits 2-0 to index into the transaction description array. When the first iTD in the periodic list is traversed after periodic schedule is enabled, the value of FRINDEX[2:0] may be other than 0, so the first transaction issued by the controller may be any of the eight available active transactions. If the active bit in the Status field of the indexed transaction description is cleared, the host controller ignores the iTD and follows the Next pointer to the next schedule data structure.

When the indexed active bit is a 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 a 0, then the host controller will store 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 (example: page 0 pointer) selected by the active transaction descriptions' PG (example value: 0b00) field with the transaction offset field. As the transaction moves data, the host controller must detect when an increment of the current buffer pointer will cross a page boundary. When

this occurs the host controller simply replaces the current buffer pointer's page portion with the next page pointer (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 via 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 microframe. In other words, the Mult field represents a transaction count for the Endpoint in the current microframe. 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 *n* Length field represents the total bytes to be sent during the microframe. The Mult field must be set by software to be consistent with Transaction *n* Length and Maximum Packet Size. The host controller will send the bytes in Maximum Packet Sized portions. After each transaction, the host controller decrements it's local copy of Transaction *n* Length by Maximum Packet Size. The number of bytes the host controller sends is always Maximum Packet Size or Transaction *n* 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 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 *n* Length field.

After all transactions for the Endpoint have completed for the microframe, Transaction *n* Length contains the total bytes received. The following actions can occur:

- If the final value of Transaction *n* Length is less than the value of Maximum Packet Size, less data was received than was allowed for from the associated Endpoint. This short packet condition does not set USBSTS[UI] (USB interrupt). The host controller does not detect this condition.
- If the device sends more than Transaction *n* Length or Maximum Packet Size bytes (whichever is less), the host controller sets the Babble Detected bit and clears the Active bit. Note, that the host controller does not update the iTD field Transaction *n* Length in this error scenario.
- If the Mult field is greater than one, the host controller automatically executes the value of Mult transactions. The host controller does not execute all Mult transactions in the following cases:

- The Endpoint is an OUT and Transaction *n* Length goes to zero before all the Mult transactions have executed (ran out of data).
- 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 microframe 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 is written back to the iTD; and the host controller proceeds to processing the next microframe.

32.6.8.2 Software operational model for iTDs

A client buffer request to an isochronous Endpoint may span 1 to N microframes.

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 figure below 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).

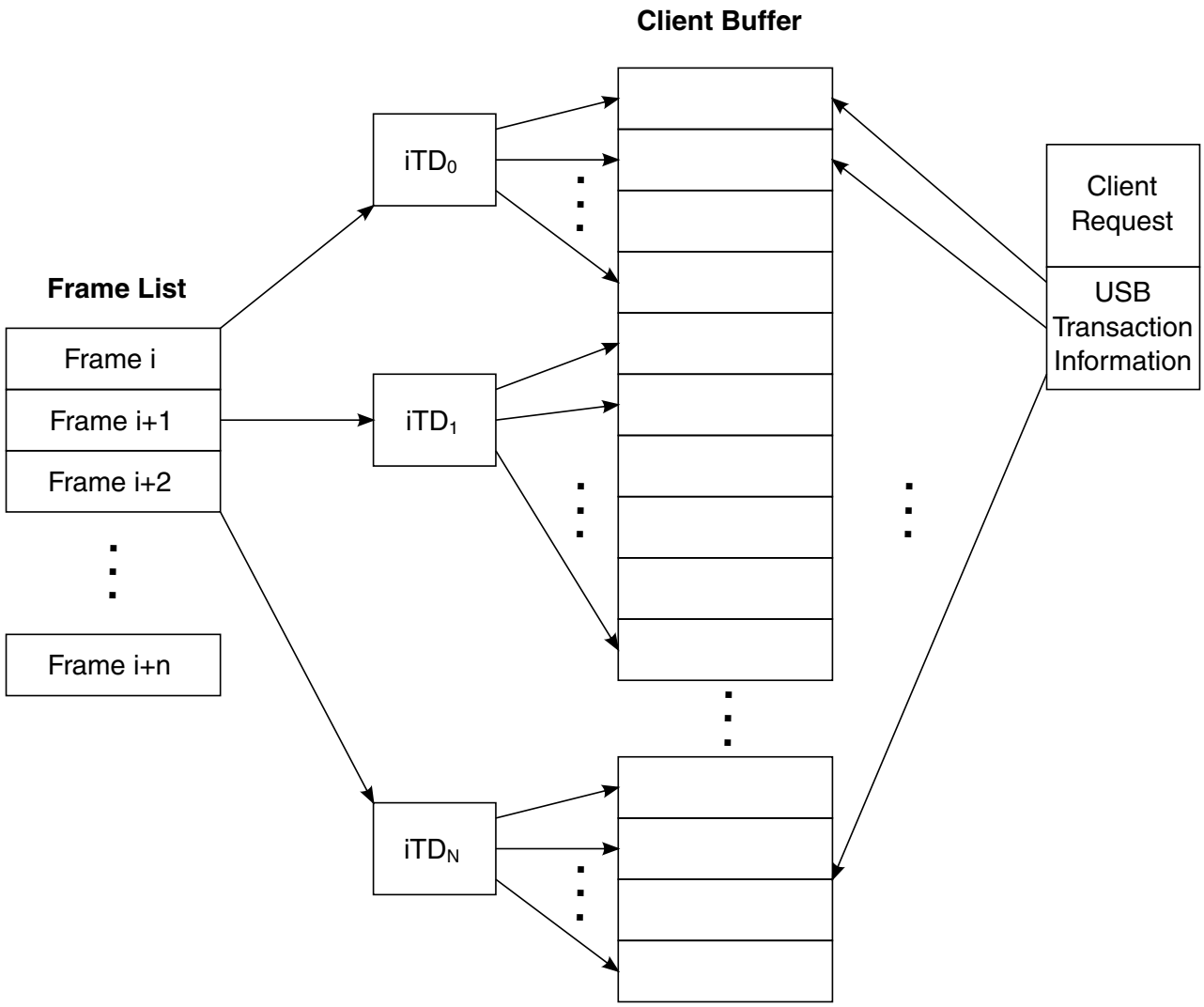


Figure 32-9. Example Association of iTDs to Client Request Buffer

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 microframe's worth of transactions. The EHCI controller does not provide per transaction results within a microframe. It treats the per microframe 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, 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.

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 will wrap 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 a one.

32.6.8.2.1 Periodic scheduling threshold

The isochronous scheduling threshold field in the 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 microframes worth of transactions. The host controller is allowed to cache one (or more) of these data structures in order to reduce memory traffic. There are three basic caching models that account for the fact the isochronous data structures span 8 microframes. The three caching models are: no caching, microframe caching and frame caching.

When software is adding new isochronous transactions to the schedule, it always performs a read of the FRINDEX register to determine the current frame and microframe the host controller is currently executing. Of course, there is no information about where in the microframe the host controller is, so a constant uncertainty factor of one microframe 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 microframe) but will always dump any accumulated schedule state at the end of the microframe. At the appropriate time relative to the beginning of every microframe, the host controller always begins schedule traversal from the frame list. Software can use the value of the 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 microframes 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 microframes). Software uses the value of the FRINDEX register (plus the constant 1 uncertainty) to determine the current microframe/frame (assume modulo 8 arithmetic in adding the constant 1 to the microframe number). For any current frame N, if the current microframe is 0 to 6, then software can safely add isochronous transactions to Frame N + 1. If the current microframe is 7, then software can add isochronous transactions to Frame N + 2.

Microframe 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 microframes indicated in theisochronous scheduling threshold field. For example, if the count value were 2, then the host controller keeps a window of two microframes worth of state (current microframe, plus the next) on chip. On each microframe boundary, the host controller releases the current microframe state and begins accumulating the next microframe state.

32.6.9 Asynchronous schedule

The asynchronous schedule traversal is enabled or disabled through USBCMD[ASE] (asynchronous schedule enable).

If USBCMD[ASE] is cleared, then the host controller simply does not try to access the asynchronous schedule via the ASYNCLISTADDR register. Likewise, if USBCMD[ASE] is set, the host controller does use the ASYNCLISTADDR register to traverse the asynchronous schedule. Modifications to USBCMD[ASE] are not necessarily immediate. Rather the new value of the bit will only be taken into consideration the next time the host controller needs to use the value of the ASYNCLISTADDR register to get the next queue head.

USBSTS[AS] indicates status of the asynchronous schedule. System software enables (or disables) the asynchronous schedule by writing a one (or zero) to USBCMD[ASE]. Software then can poll USBSTS[AS] to determine when the asynchronous schedule has made the desired transition. Software must not modify USBCMD[ASE] unless the value of USBCMD[ASE] equals that of the USBSTS[AS] (asynchronous schedule status).

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 ASYNCLISTADDR register. The default value of the ASYNCLISTADDR register after reset is undefined and the schedule is disabled when USBCMD[ASE] is cleared.

Software may only write this register with defined results when the schedule is disabled, for example, USBCMD[ASE] and the USBSTS[AS] are cleared. 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 USBCMD[ASE]. The asynchronous schedule is actually enabled when USBSTS[AS] is set.

When the host controller begins servicing the asynchronous schedule, it begins by using the value of the 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 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 occurs:

- The end of a microframe occurs.
- The host controller detects an empty list condition
- The schedule has been disabled through USBCMD[ASE].

The queue heads in the asynchronous list are linked into a simple circular list as shown in Figure 32-5. 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.

32.6.9.1 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 ASYNCLISTADDR register, then enables the list by setting USBCMD[ASE] to a 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 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
```

32.6.9.2 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 USBCMD[ASE] to a zero. Software can determine when the list is idle when USBSTS[AS] is cleared. 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, and then remove the queue head from the asynchronous list. Software removes a queue head from the asynchronous list using the following algorithm. 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, it must select another queue head still linked into the schedule and set it's H-bit. This should be completed before removing the queue head. The requirement is that software keep one queue head in the asynchronous schedule, with it's H-bit set. 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). 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 (USBCMD[IAA]-interrupt on async advance doorbell) that allows software to inform the host controller that something has been removed from it's asynchronous schedule. The second bit is a status bit (USBSTS[AAI]-interrupt on async advance) 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, it also clears the command bit. The third bit is an interrupt enable (USBINTR[AAE]-interrupt on async advance enable) that is matched with the status bit. If the status bit is set and the interrupt enable bit is set, then the host controller asserts a hardware interrupt.

The figure below illustrates a general example where 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 will remain in the asynchronous schedule.

When the host controller observes that doorbell bit being set, it makes a note of the local reachable schedule information. In this example, the local reachable schedule information includes both queue heads (A & 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).

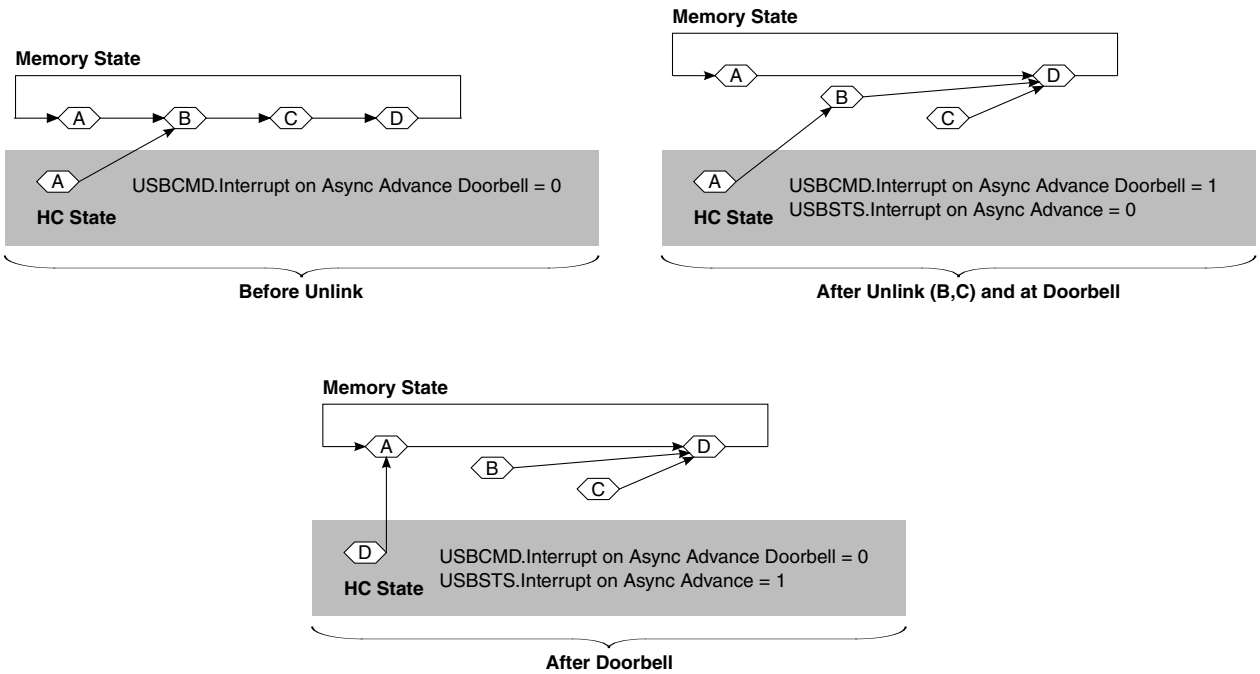


Figure 32-10. Generic queue head unlink scenario

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 USBSTS[AAI].

Software may re-use the memory associated with the removed queue heads after it observes USBSTS[AAI] is set, following assertion of the doorbell. Software should acknowledge the interrupt on async advance status as indicated in the USBSTS register, before using the doorbell handshake again

32.6.9.3 Empty asynchronous schedule detection

EHCI uses two bits to detect when the asynchronous schedule is empty.

The queue head data structure (see Table 32-26) defines an H-bit in the queue head, which allows software to mark a queue head as being the head of the reclaim list. Host controller also keeps a 1-bit flag in the USBSTS register (Reclamation) that is cleared when the host controller observes a queue head with the H-bit set. The reclamation flag in the status register is set when any USB transaction from the asynchronous schedule is executed (or whenever the asynchronous schedule starts, see Asynchronous schedule traversal: Start event.

If the controller ever encounters an H-bit of one and a Reclamation bit of zero, the controller simply stops traversal of the asynchronous schedule.

The figure below shows an example illustrating the H-bit in a schedule.

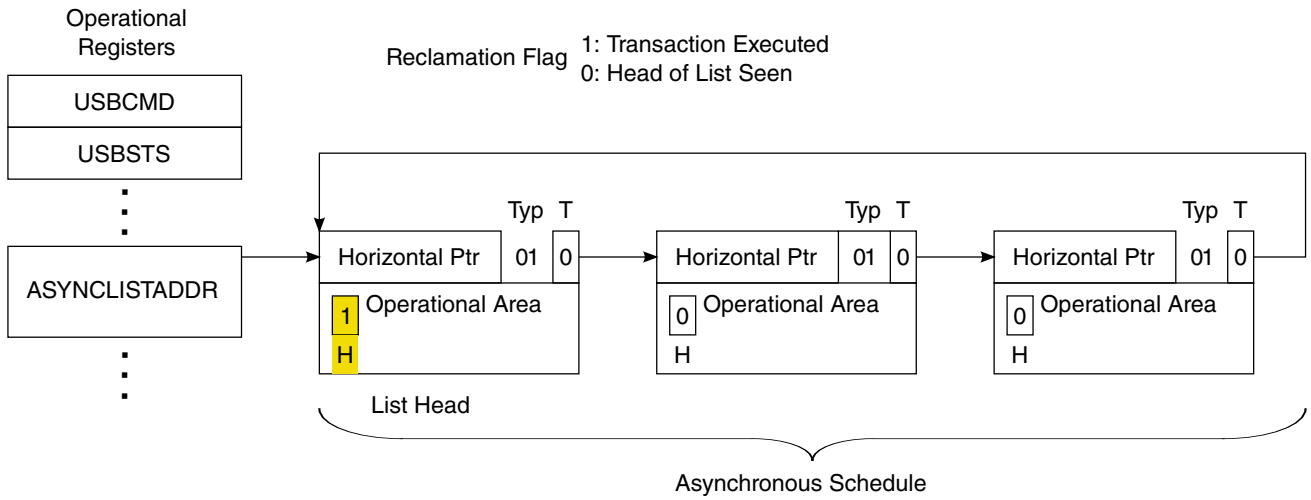


Figure 32-11. Asynchronous schedule list with annotation to mark head of list

32.6.9.4 Asynchronous schedule traversal: Start event

Once the host controller has idled itself using the empty schedule detection, it naturally activates and begins processing from the Periodic Schedule at the beginning of each microframe.

In addition, it may have idled itself early in a microframe. When this occurs (idles early in the microframe) the host controller must occasionally reactivate during the microframe and traverse the asynchronous schedule to determine whether any progress can be made. Asynchronous schedule Start Events 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 microframe is equivalent to the transition from the periodic schedule, or
- The asynchronous schedule traversal restarts from a sleeping state.

32.6.9.5 Reclamation status bit (USBSTS Register)

The operation of the empty asynchronous schedule detection feature depends on the proper management of the Reclamation bit (RCL) in the USBSTS register.

The host controller tests for an empty schedule just after it fetches a new queue head while traversing the asynchronous schedule. The host controller sets USBSTS[RCL] whenever an asynchronous schedule traversal Start Event occurs. USBSTS[RCL] is also set whenever the host controller executes a transaction while traversing the asynchronous schedule.The host controller clears USBSTS[RCL] whenever it finds a queue head with its H-bit set. 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, the resulting behavior is undefined. The host controller may clear USBSTS[RCL] when executing from the periodic schedule.

32.6.10 Managing control/bulk/interrupt transfers via 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 defined in Queue element transfer descriptor (qTD).

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. Each qTD represents one or more bus transactions, which is defined in the context of the EHCI 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 will set one 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 (that is, 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 will occur for the Endpoint and the host controller will not advance the queue.

32.6.10.1 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.

The EHCI specification requires that the buffer associated with the transfer be virtually contiguous. This means that if the buffer spans more than one physical page, it must obey the following rules:

- 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 4K chunk beyond the first page, each buffer portion matches to a full 4K 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 figure below illustrates these requirements.

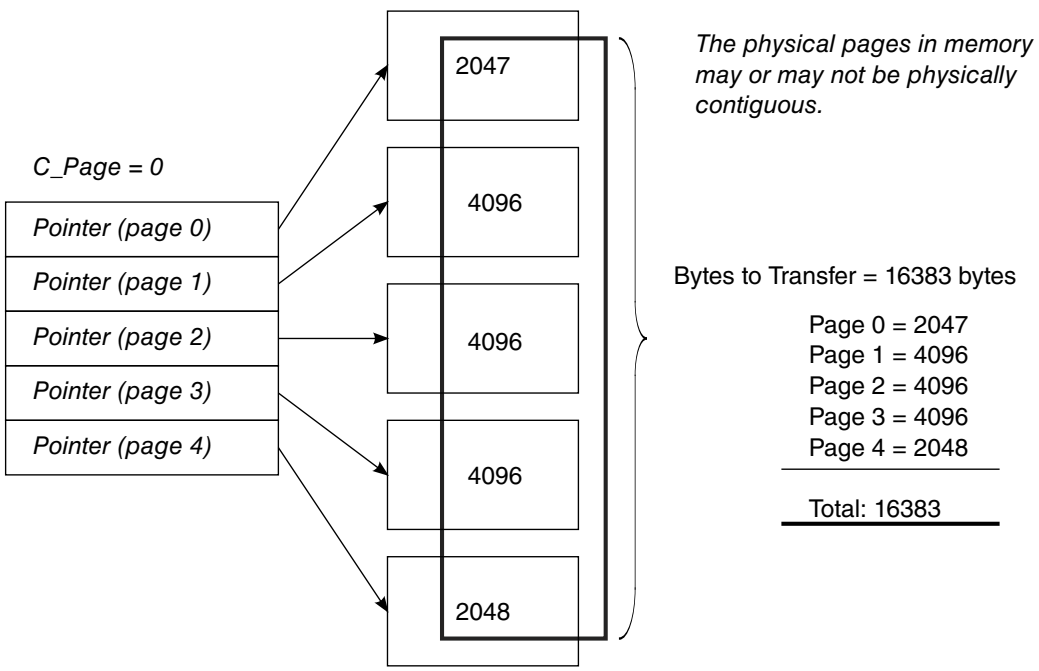


Figure 32-12. Example mapping of qTD buffer pointers to buffer pages

The buffer pointer list in the qTD is long enough to support a maximum transfer size of 20K bytes. This case occurs when all five buffer pointers are used and the first offset is zero. A qTD handles a 16Kbyte buffer with any starting buffer alignment.

The host controller uses the 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 figure above 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 cleared. 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 4K page.

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 cleared) 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 will increment 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. There are 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 that the only valid adjustment the host controller may make to C_Page is to increment by one.

32.6.10.2 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 microframe within a 1 millisecond 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 a 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 table below.

Table 32-37. Example periodic reference patterns for interrupt transfers

Frame # reference sequence	Description
0, 2, 4, 6, 8, ... S-Mask = 0x01	A queue head for the bInterval of 2 milliseconds (16 microframes) 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 0x01, indicating that the transaction for the Endpoint should be executed on the bus during microframe 0 of the frame.
0, 2, 4, 6, 8, ... S-Mask = 0x02	Another example of a queue head with a bInterval of 2 milliseconds is linked into the periodic frame list at exactly the same interval as the previous example. However, the S-Mask is set to 0x02 indicating that the transaction for the Endpoint should be executed on the bus during microframe 1 of the frame.

32.6.10.3 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, 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.

32.6.11 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 will use in the next transaction to the Endpoint (see Table 32-24). The Ping State bit is only managed by the host controller for queue heads that meet all of the following criteria:

- The queue head is not an interrupt
- The EPS field equals High-Speed
- The PIDCode field equals OUT

The table below 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 32-38. 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[3]
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 cleared and Halt set). 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 is described in Table 32-24. The defined ping protocol allows the host to be imprecise on the initialization of the ping protocol (that is, start in Do OUT when there is uncertainty about the space in the device). 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.

NOTE

For high-speed bulk and control Endpoints, a host controller queries the high-speed device Endpoint with a special ping token to determine whether the device has sufficient space for the next OUT transaction. The mechanism avoids using bus time to send data until the host controller knows that the Endpoint has space for the data. If a timeout occurs after the data phase of an OUT transaction, the host controller fails to enter the ping state and instead retries the OUT token again.

The Ping flow control for the high-speed devices does not work under this condition. Therefore, some USB bandwidth could be wasted. However, NAK response to PING token or timeout is expected to be an unusual occurrence. A high-speed bulk/ control Endpoint must specify its maximum NAK rate in its Endpoint descriptor. The Endpoint is allowed to NAK at most one time each micro-frame period.

32.6.12 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 a USB 2.0 hub, utilizing the split transaction protocol. Refer to the 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.

EHCI uses dedicated data structures for managing full-speed isochronous data streams. Control, Bulk and Interrupt are managed using the queuing data structures. 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 processes and manages the split transaction protocol.

32.6.12.1 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 via 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 a one. If this is not a control transfer type Endpoint, the C bit must be initialized by software to be a 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 a zero, the split transaction token's ET field is set to indicate a bulk Endpoint. When the C bit is a 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.

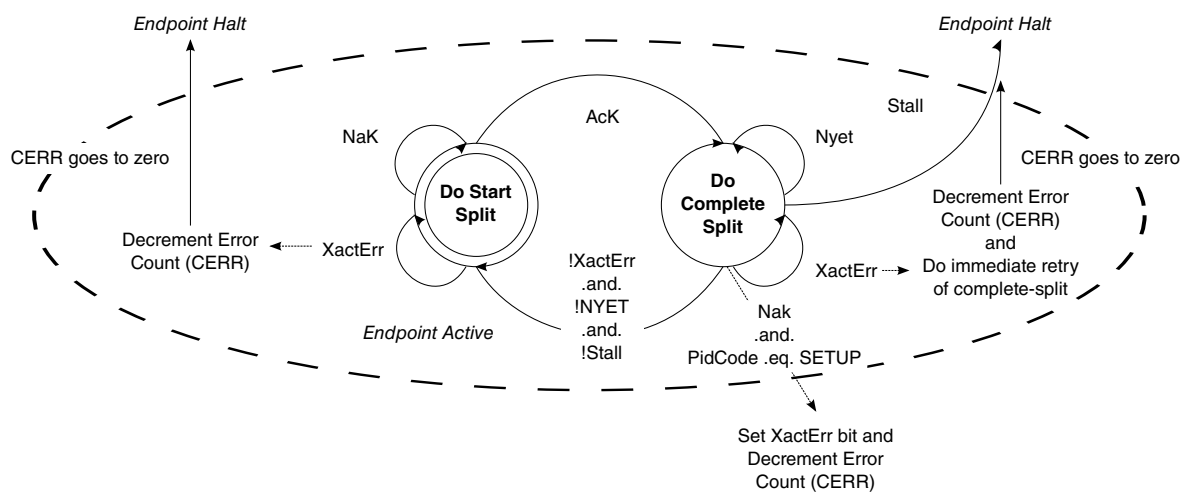


Figure 32-13. Host controller asynchronous schedule split-transaction state machine

32.6.12.1.1 Asynchronous-do-start-split

Do-start-split 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 transaction translator. If the bus transaction completes without an error and PID Code indicates an IN or OUT transaction, then the host controller reloads the error counter (Cerr). If it is a successful bus transaction and the PID Code indicates a SETUP, the host controller will 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.

32.6.12.1.2 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 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 PID Code 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 PID Code indicates a SETUP, the host controller must not adjust the value of Cerr.

Independent of PID Code, the following responses have the indicated effects:

- Transaction Error (XactErr). Timeout/data CRC failure. 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 microframe to execute the retry, the host controller ensures 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. When the host controller returns to the asynchronous schedule in the next microframe, the first transaction from the schedule will be the retry for this Endpoint. If Cerr went to zero, the host controller halts 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 PID Code is a SETUP, then the Nak response is a protocol error. The XactErr status bit is set 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 PID Code indicates an IN, then any of following response is 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, moves the data pointer by the number of bytes received, decrements the BytesToTransfer field by the number of bytes received, and toggles the dt bit). The host controller then exits 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 PID Code indicates an OUT/SETUP, then any of following response is 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 bytes to transfer field is decremented by the same amount and the data toggle bit (dt) is toggled. The host controller then exits this state.

  Advancing the transfer state may cause other processing events such as retirement of the qTD and advancement of the queue.

32.6.12.2 Split transaction interrupt

Split-transaction interrupt-IN/OUT Endpoints are managed using 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 will visit 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.

32.6.12.2.1 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 microframes the start-splits and complete-splits for each Endpoint will occur. The characteristics of the transaction translator are such that the high-speed transaction protocol must execute during explicit microframes, or the data or response information in the pipeline is lost. The figure below illustrates the general scheduling boundary conditions that are supported by the EHCI periodic schedule and queue head data structure. The S and C_n labels indicate microframes where software can schedule start-splits and complete splits (respectively).

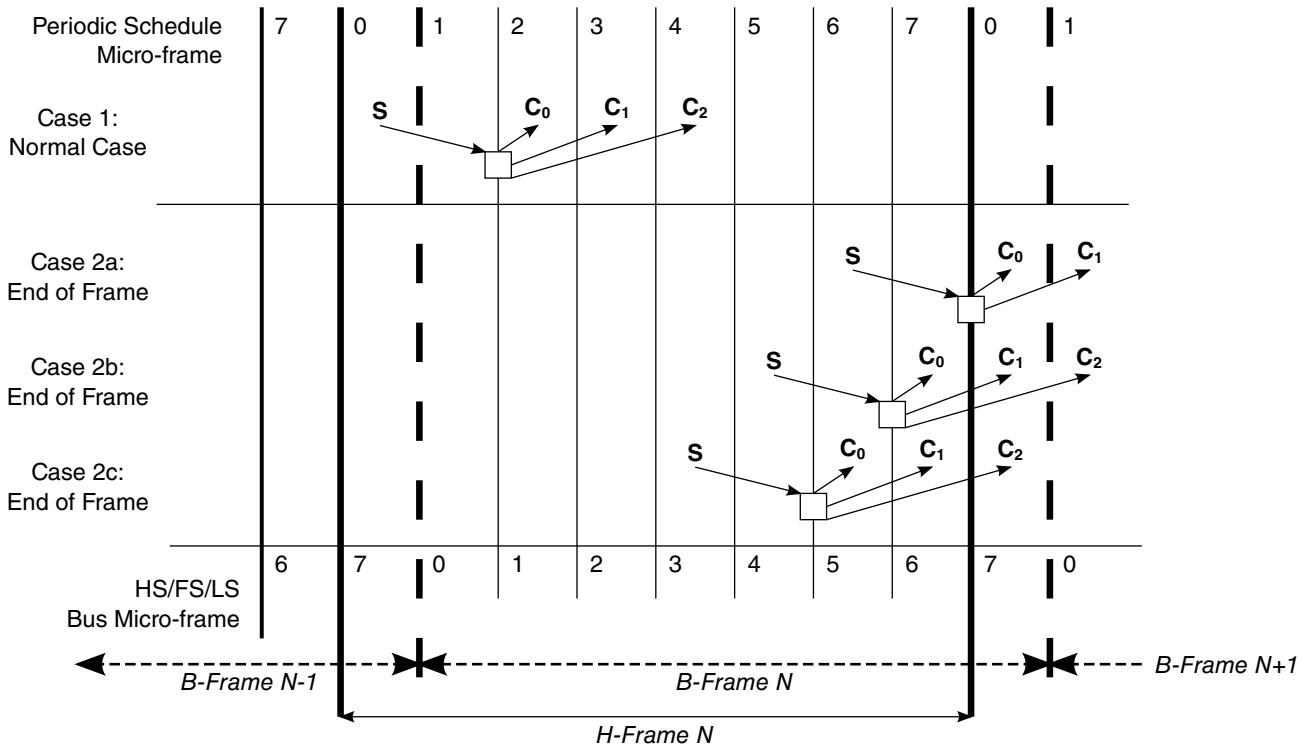


Figure 32-14. Split transaction, interrupt scheduling boundary conditions

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 microframe 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.

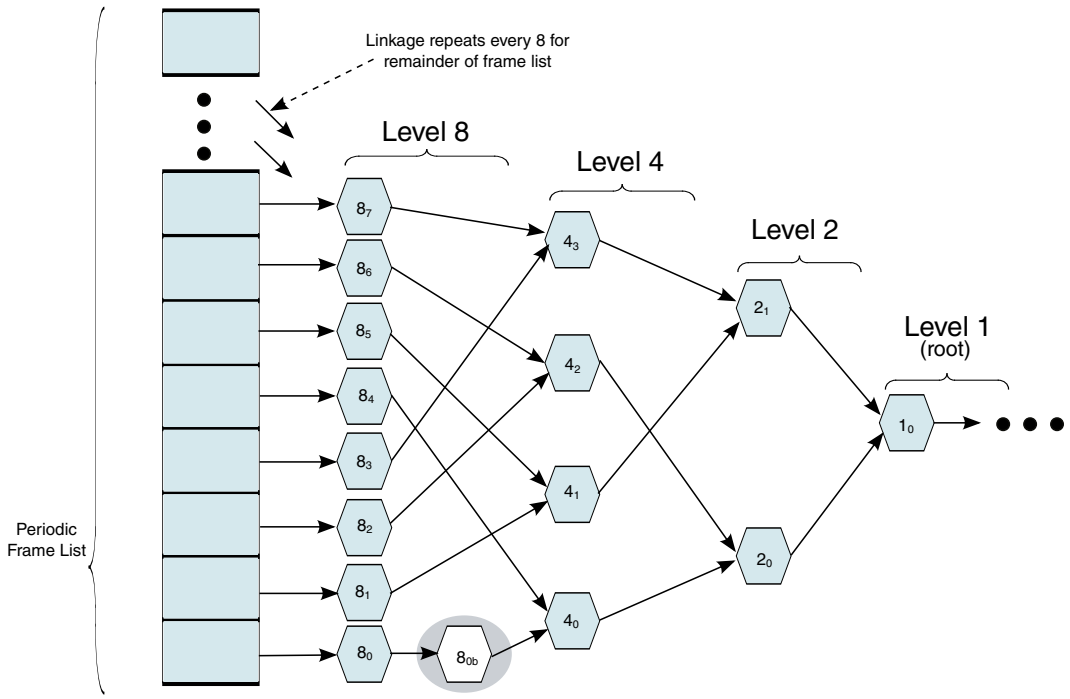


Figure 32-15. General structure of EHCI periodic schedule utilizing interrupt spreading

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} were 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. Periodic frame span traversal node (FSTN), 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 a single bit residing in the Status field of a queue head (Table 32-24). 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 microframe (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 32-14, case one, the S-mask would have a value of 0b0000_0001 indicating that if the queue head is traversed by the host controller, and the SplitXState indicates Do_Start, and the current microframe 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 microframes (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 32-14, case one, the C-mask would have a value of 0b0001_1100 indicating that if the queue head is traversed by the host controller, and the SplitXState indicates Do_Complete, and the current microframe 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.

32.6.12.2.2 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.

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.

There are four components to the use of FSTNs:

- FSTN data structure, defined in Periodic frame span traversal node (FSTN).
- A Save Place indicator; this is always an FSTN with its Back Path Link Pointer[T] bit cleared.
- A Restore indicator; this is always an FSTN with its Back Path Link Pointer[T] bit set.
- Host controller FSTN traversal rules.

When the host controller encounters an FSTN during microframes 2 through 7 it simply follows the node's Normal Path Link Pointer to access the next schedule data structure. Note that the FSTN's Normal Path Link Pointer[T] bit may set, which the host controller must interpret as the end of periodic list mark.

When the host controller encounters a Save-Place FSTN in microframes 0 or 1, it saves the value of the Normal Path Link Pointer and sets 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 are 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 microframe.

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 only considers it for execution if its SplitXState is DoComplete (note: this applies whether the PID Code indicates an IN or an OUT). Refer to the *EHCI Specification* for a complete list of additional conditions that must be met in general for the host controller to issue a bus transaction. Note that the host controller must not execute a Start-split transaction while executing in Recovery Path mode. Refer to the *EHCI Specification* 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 microframe 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 microframe, the host controller starts traversal at the frame list.

An example traversal of a periodic schedule that includes FSTNs is illustrated in the figure below.

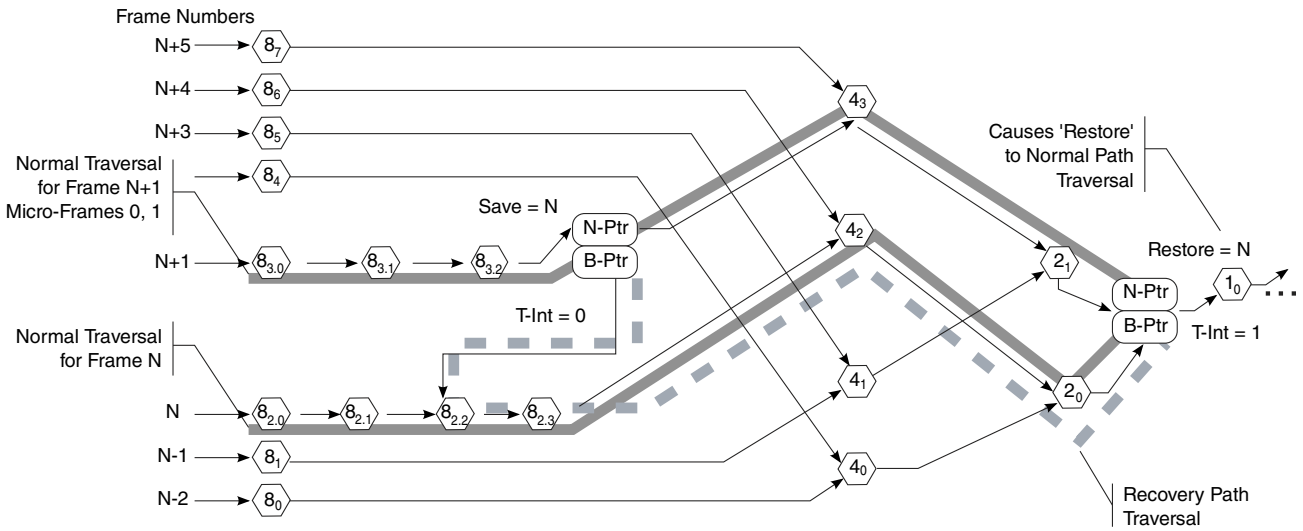


Figure 32-16. Example host controller traversal of recovery path via FSTNs

In frame N (microframes 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 is not executing in Recovery Path mode. When it encounters the Restore FSTN, (Restore-N), during microframes 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 (microframes 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 (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 microframes 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, 10 ...}.

In frame N+1 (microframes 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 microframes include: {$8_{3.0}$, $8_{3.1}$, $8_{3.2}$, Save-A, 4_3, 2_1, Restore-N, 1_0 ...}.

32.6.12.2.3 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. Note that 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.

  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, 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 are 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 rebalance 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.

32.6.12.2.4 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 is 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 microframe 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).

32.6.12.2.5 Split transaction execution state machine for interrupt

In the following section, all references to microframe are in the context of a microframe 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 microframe number. It is an eight-bit value calculated by the host controller at the beginning of every microframe. 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 microframe number. For example, if the current microframe is 0, then cMicroFrameBit will equal 0b0000_0001.

  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-split transaction for the current microframe.

The figure below illustrates how a complete interrupt split transaction is managed. 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 is 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 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 stops retrying the bus transaction and halts the endpoint, thus requiring system software to detect the condition and perform system-dependent recovery).

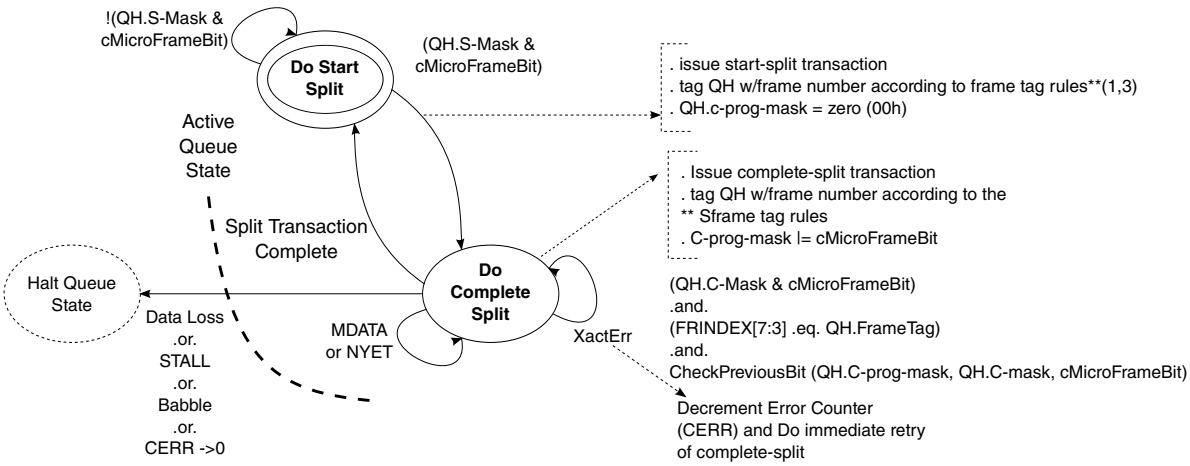


Figure 32-17. Split transaction state machine for interrupt

32.6.12.2.6 Periodic interrupt-do-start-split

This is the state when 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, and so on).
- 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 Periodic interrupt-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), bit-wise ANDs QH[S-mask] with cMicroFrameBit to determine whether to execute a start-split. If the result is non-zero, then the host controller issues a start-split transaction. If the PID Code 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, sets C-prog-mask to zero (0x00), 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.

32.6.12.2.7 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 microframe.
- 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 microframe. 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 microframe. Note this is the same test performed in the Do Start Split state. 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 microframe. Test D and Test A evaluating to TRUE at the same time is a system software error. Behavior is undefined.

If (Test 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. 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 complete-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 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, and so on

  The Cerr field is decremented and the XactErr bit in the Status field is set. The complete split transaction is immediately retried (if Cerr is non-zero). If there is not enough time in the microframe 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 microframe 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 on 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.

- 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 exits 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.

  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, 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 cleared and the Halted bit is set 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 32-39. 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 microframe. Host controller should continue walking the schedule.

Table continues on the next page...

Table 32-39. Interrupt IN/OUT do complete split state execution criteria (continued)

Condition	Action	Description
A not(C)	If PIDCode = IN Halt QHDIf 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 PID Code is an IN, then the queue head must be halted. If PID Code 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 Cerr. In either case, set the Missed Microframe 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 PID Code is an IN, then the queue head must be halted. If PID Code 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 Cerr. In either case, set the Missed Microframe 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 QHDIf 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 PID Code is an IN, then the Queue head must be halted. If PID Code 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 Cerr. In either case, set the Missed Microframe bit in the status field to a one. Note that 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 normal processing mode. Regardless, the host controller must not execute a start-split in the context of a executing in a Recovery Path mode.

32.6.12.2.8 Managing the 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 32-14).
- 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 in Figure 32-14.
- 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 in Figure 32-14.

32.6.12.2.9 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 host controller provides a simple assist to system software. System software sets the Inactivate-on-next-Transaction (I) bit to signal the host controller that it intends to update the S-mask and C-mask on this queue head. System software then waits for the host controller to observe the I-bit is set and transitions the Active bit to a zero. The rules for how and when the host controller clears the Active bit are:

- If the Active bit is cleared, no action is taken. The host controller does not attempt to advance the queue when the I-bit is set.
- If the Active bit is set and the SplitXState is DoStart (regardless of the value of S-mask), the host controller simply clears the Active bit. 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 microframe, the host controller must not issue the start-split bus transaction; it must clear the Active bit.

System software must save transfer state before setting the I-bit. This is required so that it can correctly determine what transfer progress (if any) occurred after the I-bit was set and the host controller executed it's final bus-transaction and cleared the Active bit.

After system software has updated the S-mask and C-mask, it must then reactivate the queue head. Since 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 using the I-bit.

1. Set the Halted bit, then
2. Clear the I-bit, then
3. Set the Active bit and clear the Halted bit in the same write.

Setting the Halted bit inhibits the host controller from attempting to advance the queue between the time the I-bit is cleared and the Active bit is set.

32.6.12.3 Split transaction isochronous

Full-speed isochronous transfers are managed using the split-transaction protocol through a USB 2.0 transaction translator in a USB 2.0 hub.

The host controller utilizes siTD data structure to support the special requirements of isochronous split-transactions. This data structure uses the scheduling model of isochronous TDs (see 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.

32.6.12.3.1 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 microframes the start-splits and complete-splits for each full-speed isochronous endpoint occur. The requirements described in Split transaction scheduling mechanisms for interrupt , apply.

Figure 32-18 illustrates the general scheduling boundary conditions that are supported by the EHCI periodic schedule. The S_n and C_n labels indicate microframes 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 figure below illustrates the relationship of an siTD to the H-Frame.

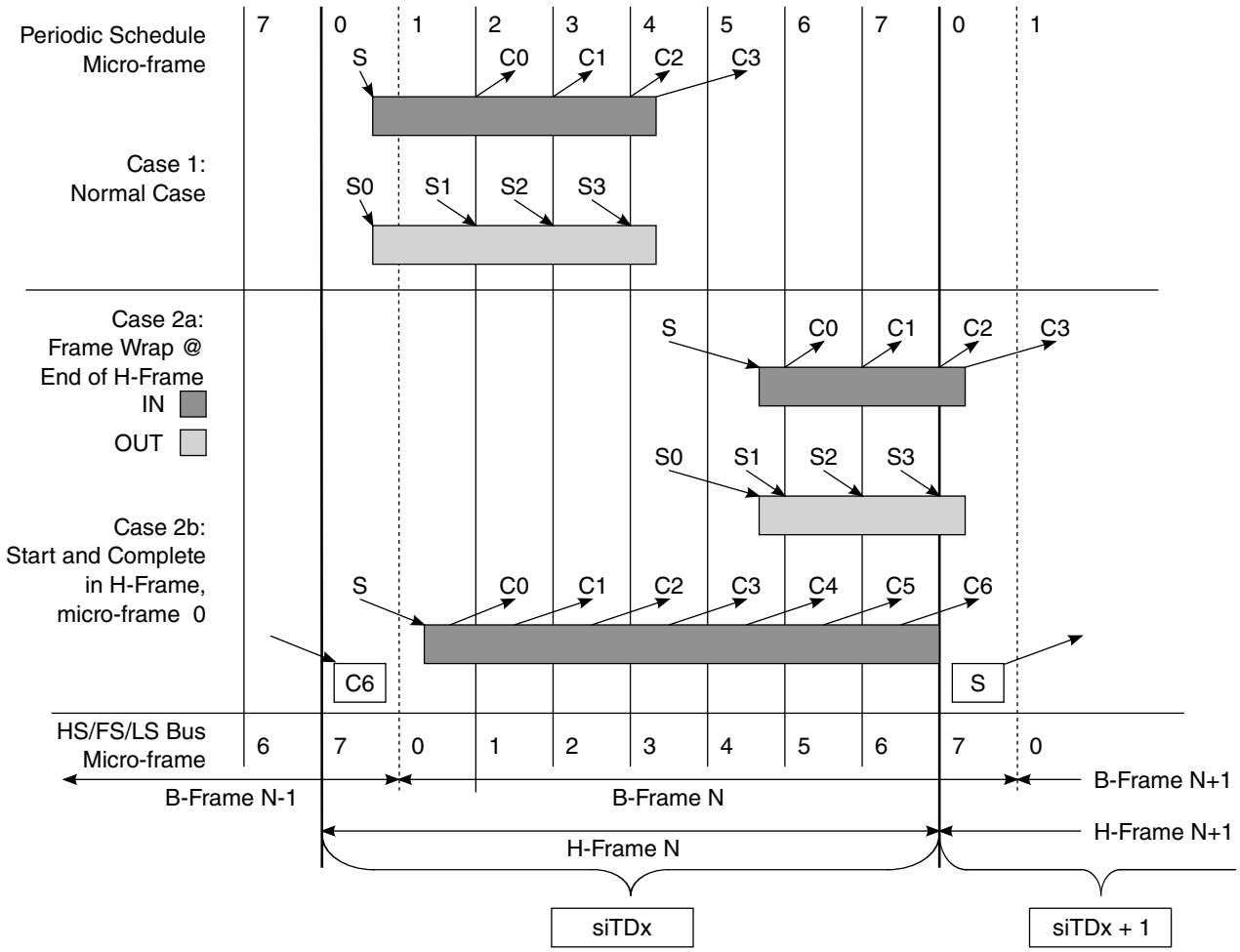


Figure 32-18. Split transaction, isochronous scheduling boundary conditions

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, microframes 6 or 7 (H-Frame microframe 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.

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 microframe. 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 Table 32-17). This bit is used to track the current state of the split transaction. The rules for managing this bit are described in Split transaction execution state machine for isochronous .

- Frame S-mask

  This is a bit-field wherein system software sets a bit corresponding to the microframe (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 32-18, case 1, the S-mask would have a value of 0b0000_0001 indicating that if the siTD is traversed by the host controller, and the SplitXState indicates Do Start Split, and the current microframe 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 microframes (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 32-18, case 1, the C-mask would have a value of 0b 0011_1100 indicating that if the siTD is traversed by the host controller, and the SplitXState indicates Do Complete Split, and the current microframe as indicated by 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.

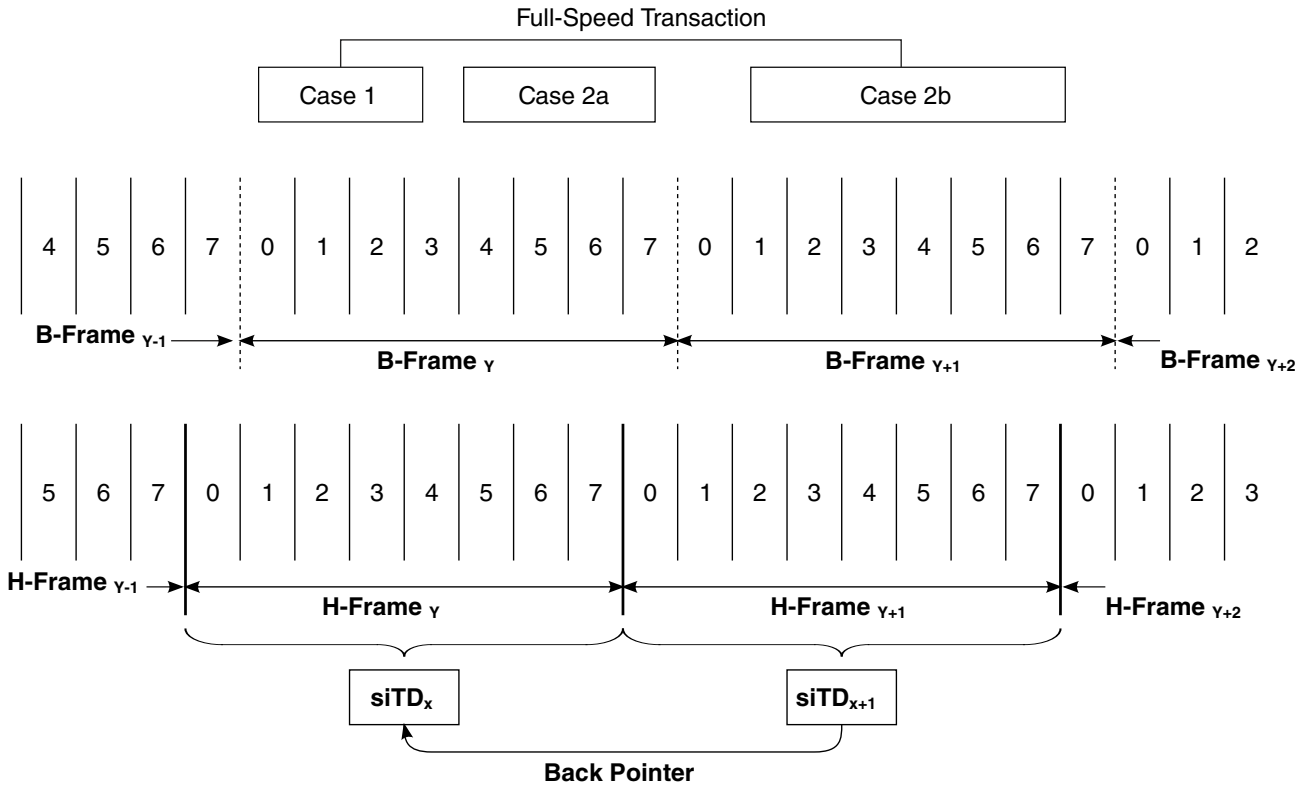


Figure 32-19. siTD scheduling boundary examples

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. siTDX 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 microframe 7 of H-Frame$_{Y+1}$, or microframe 0 of H-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-Frame$_{Y+2}$ is to use siTD$_{X+2}$'s back pointer.

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, microframe 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 microframe 1 of H-Frame N and the last complete-split would need to occur in microframe 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.

32.6.12.3.2 Tracking split transaction progress for isochronous transfers

Isochronous endpoints do not employ the concept of a halt on error, however the host controller does identify and report per-packet errors observed in the data stream.

This includes schedule traversal problems (skipped microframes), timeouts and corrupted data received.

In similar kind to interrupt split-transactions, the portions of the split transaction protocol must execute in the microframes 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 using the schedule in the exact microframe 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 re-initialized (activated) before the host controller gets back to the siTD (in a future microframe).

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 using 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 Split transaction scheduling mechanisms for isochronous , 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 microframe (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. 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[Total Bytes to Transfer] field 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 clearing the Active bit. 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. In summary, the periodic pipeline rules require that on a microframe boundary, the transaction translator holds the final two bytes received (if it has not seen an End Of Packet (EOP)) in the full-speed bus pipe stage and gives the remaining bytes to the high-speed pipeline stage. At the microframe boundary, the transaction translator could have received the entire packet (including both CRC bytes)

but not received the packet EOP. In the next microframe, the transaction translator responds with an MDATA and sends 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 it's 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 is not consistent and the transaction translator detects and reacts 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 (for example, state not advanced) and report the appropriate error to the client driver.

32.6.12.3.3 Split transaction execution state machine for isochronous

In this section, all references to microframe are in the context of a microframe within an H-Frame.

If the active bit in the status byte is a zero, the host controller ignores the siTD and continues traversing the periodic schedule. Otherwise the host controller processes 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 Tracking split transaction progress for isochronous transfers , plus the variable cMicroFrameBit defined in 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.

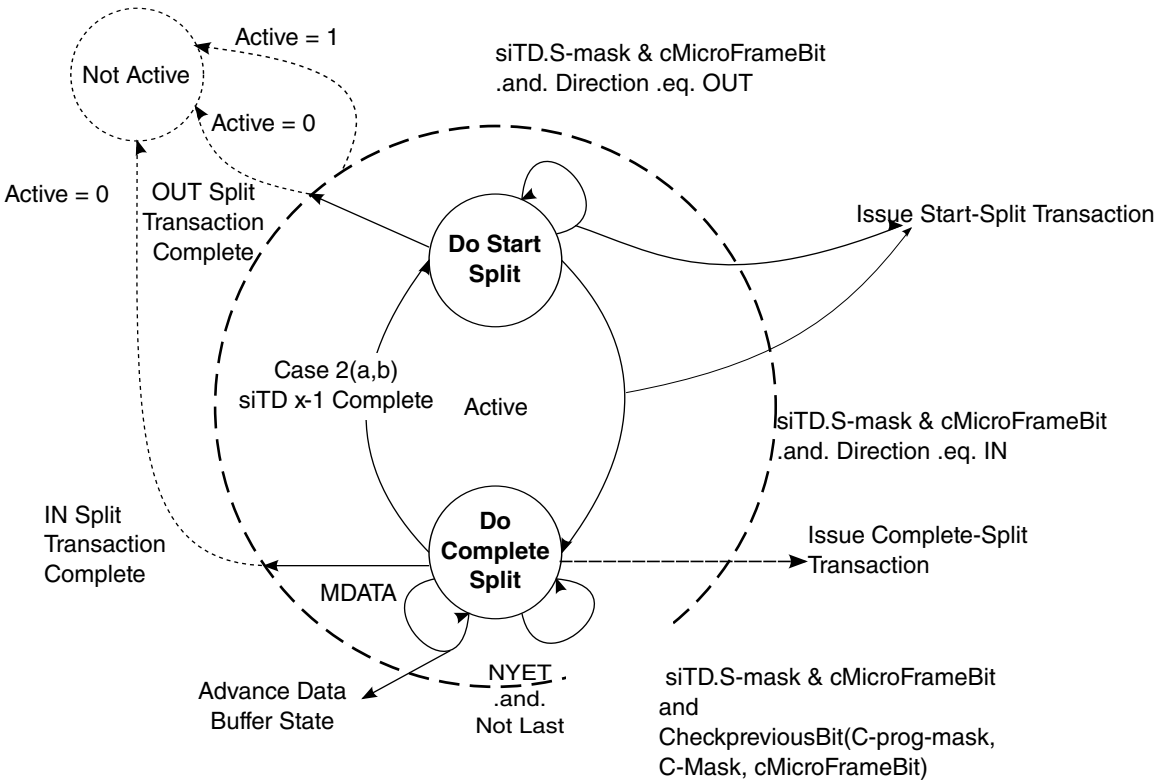


Figure 32-20. Split transaction state machine for isochronous

32.6.12.3.4 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 executes 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 select 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 32-19) is used to determine the initial value of TP. The initial cases are summarized in the table below.

Table 32-40. Initial conditions for OUT siTD 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 32-41. 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 (for example, 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 select) bit is updated appropriately.
- The siTD[TP] and siTD[T-count] fields are updated appropriately as defined in Table 32-41.

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 clear the Active bit when it detects that all of the schedule data has been sent to the bus. Setting the Active bit to zero depends on siTD.TP being 00 or 11, and siTD.Total Bytes decrements to 0. 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 yields 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 does not progress appropriately. The transaction translator observes protocol violations in the arrival of the start-splits for the OUT endpoint (that is, the transaction position annotation is incorrect as received by the transaction translator).

Example scenarios are described in Split transaction for isochronous-processing example.

The host controller 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 microframes. It can then clear the siTD's Active bit and stop execution of this siTD. This saves on both memory and high-speed bus bandwidth.

32.6.12.3.5 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 sequence in which they are applied depends on which microframe 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. The individual tests are as follows.

- Test A

cMicroFrameBit is bit-wise ANDed with the siTD[C-mask] field. A non-zero result indicates that software scheduled a complete-split for this endpoint, during this microframe. 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 given below (this is slightly different than the algorithm used in Periodic interrupt-do-complete-split). The sequence in which this test is applied depends on the current value of FRINDEX[2-0]. If 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 FRINDEX[2-0] is zero or one, this is a case 2a or 2b scheduling boundary (see Figure 32-18). See Complete-split for scheduling boundary cases 2a, 2b, for details in handling this condition.

If Test A and Test B evaluate to true, the host controller executes 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 perform the following actions:

1. Decrement the number of bytes received from siTD[Total Bytes To Transfer]

2. Adjust siTD[Current Offset] by the number of bytes received
3. Adjust the siTD[P] (page select) field if the transfer caused the host controller to use the next page pointer
4. 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 cleared and the siTD[Status-Babble Detected] bit must be set. The fields siTD[Total Bytes To Transfer], siTD[Current Offset], and siTD[P] are not required to be updated as a result of this transaction attempt.

The host controller accepts (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. The host controller may optionally clear siTD[Status-Active] and set siTD[Status-Babble Detected] when it receives 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 clears the Active bit.

- Transaction Error (XactErr)

  The complete-split transaction encounters a Timeout, CRC16 failure, and so on. The siTD[Status] field XactErr field is set 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 microframe occurs, the Active bit is cleared.

- 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 cleared. 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 USB interrupt status bit (USBSTS[UI]) 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 Periodic interrupt-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 cleared. 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 complete-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-masks incorrectly. The host controller must set the XactErr bit and clear the Active bit.

- 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 microframe X to X+1 and during microframe X, the transaction translator responds with an MDATA and the data accumulated up to the end of microframe 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 clears the Active bit.

32.6.12.3.6 Complete-split for scheduling boundary cases 2a, 2b

Boundary cases 2a and 2b (INs only) (see Figure 32-18) 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 32-42. 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 have the T bit in the siTD[Back Pointer] field cleared. Otherwise, software must set the T bit in siTD[Back Pointer]. 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 0x1 and the $siTD_X$[Back Pointer] T-bit is zero, or
- If cMicroFrameBit is a 0x2 and $siTD_X$[S-mask[0]] is 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 set 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 32-42) 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 set and SplitXStat is Do Start Split), then the host controller must clear the Active bit and set the Missed Micro-Frame status bit and the resultant status is written back to memory.

If $siTD_{X-1}$'s Active bit is cleared, (because it was cleared 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 cleared it), 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 1 and $siTD_X$[S-mask[0]] is 1). 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 cleared 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 and an S-mask with bit 0 set. This scheduling combination is not supported and the behavior of the host controller is undefined.

32.6.12.3.7 Split transaction for isochronous-processing example

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 using the Execute Transaction queue head traversal state machine.

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 few frames worth of scheduling required to schedule a case 2a full-speed isochronous data stream.

Table 32-43. Example case 2a-software scheduling siTDs for an IN endpoint

$siTD_X$		Micro-frames								InitialSplitXState
#	Masks	0	1	2	3	4	5	6	7	
X	S-mask					1				Do start split
	C-mask	1	1					1	1	

Table continues on the next page...

Table 32-43. Example case 2a-software scheduling siTDs for an IN endpoint (continued)

$siTD_X$		Micro-frames								InitialSplitXState
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. Since this is the case-2a frame-wrap case, S-masks of all siTDs for this endpoint have a value of 0x10 (a one bit in microframe 4) and C-mask value of 0xC3 (one-bits in microframes 0,1, 6 and 7). Additionally, software ensures that the Back Pointer field of each siTD references the appropriate siTD data structure (and the Back Pointer T-bits are cleared).

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 microframes 0 and 1 are ignored because the state is Do Start Split. During microframe 4, the host controller determines that it can run a start-split (and does) and changes SplitXState to Do Complete Split. During microframes 6 and 7, the host controller executes complete-splits. Notice the siTD for frame X+1 has it's 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 microframes 0 and 1 it will detect that it must execute complete-splits.

During H-Frame X+1, microframe 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 cleared and results written back to $siTD_X$. The host controller retains the fact that $siTD_X$ is retired and transitions the SplitXState in $siTD_{X+1}$ to Do Start Split. At this point, the host controller is prepared to execute the start-split for $siTD_{X+1}$ when it reaches microframe 4. If the split-transaction completes early (transaction-complete is defined in Periodic isochronous-do complete split), that is, before all the scheduled complete-splits have been executed, the host controller changes $siTD_X$[SplitXState] to Do Start Split early and naturally skips the remaining scheduled complete-split transactions. For this example, $siTD_{X+1}$ does not receive a DATA0 response until H-Frame X+2, microframe 1.

During H-Frame X+2, microframe 0, the host controller detects that $siTD_{X+2}$'s Back Pointer[T] bit is 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 it's next pointer without any state change updates to $siTD_{X+2}$.

During H-Frame X+2, microframe 1, the host controller detects $siTD_{X+2}$'s S-mask[0] bit is 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 clears the Active bit. 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 microframe 4.

32.6.13 Port test modes

EHCI host controllers 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 required, port test sequence, assuming the CF-bit in the CONFIGFLAG register is set, is as follows:

1. Disable the periodic and asynchronous schedules by clearing the USBCMD[ASE] and USBCMD[PSE].
2. Place all enabled root ports into the suspended state by setting the Suspend bit in the PORTSC register (PORTSC[SUSP]).
3. Clear USBCMD[RS] (run/stop) and wait for USBSTS[HCH] to transition to a one. In Device mode, the Test Mode starts only if Run/Stop bit is set to 1. In Host mode, the Test Mode starts regardless of Run/Stop bit.
4. 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 USBCMD[RS] must then be transitioned back to one, in order to enable transmission of SOFs out of the port under test.
5. When the test is complete, system software must ensure the host controller is halted (HCH bit is a one) then it terminates and exits test mode by setting USBCMD[RST].

32.6.14 Interrupts

The EHCI host controller hardware provides interrupt capability based on a number of sources.

The following list describes the 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, and so on)
- Host controller error events

All transaction-based sources are maskable through the host controller's Interrupt Enable register (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 generates 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 microframes. This means that the host controller will not generate interrupts any more frequently than once every eight microframes.

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 system 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, CPU 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. 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 only method software should use for acknowledging an interrupt is by transitioning the appropriate status bits in the USBSTS register from a one to a zero.

32.6.14.1 Transfer/transaction based interrupts

These interrupt sources are associated with transfer and transaction progress. They are all dependent on the next interrupt threshold.

32.6.14.1.1 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.

Table 32-44. Summary of transaction errors

Event/ Result	Queue Head/qTD/iTD/siTD Side Effects		USBSTS[USBERRINT]
	Cerr	Status Field	
CRC	-1	XactErr set	1[1]
Timeout	-1	XactErr set	1[1]
Bad PID[2]	-1	XactErr set	1[1]
Babble	N/A	See Serial bus babble	1
Buffer Error	N/A	See 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.

[2] The host controller received a response from the device, but it could not recognize the PID as a valid PID.

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 PID Code indicates a SETUP, the following responses are protocol errors and result in XactErr bit being set 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.

32.6.14.1.2 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. 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.

USBSTS[UEI] (USB error interrupt) is set and if the USBINTR[UEE] (USB error interrupt enable) is set, then a hardware interrupt is signaled to the system at the next interrupt threshold. The host controller must never start an OUT transaction that babbles across a microframe 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 re-sends 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.

32.6.14.1.3 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 forces 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*.

NOTE

When the USB controller is in device mode and the host sends two consecutive ISO OUT transactions (for example: OUT - DATA0 - OUT - DATA1) with a short inter-packet delay between DATA0 and the second OUT (less than 200 ns), the device sees the DATA1 packet as a short-packet even if it is correctly formed. In this case, the device terminates the transfer, generating an IOC interrupt (USBSTS[UI]). Note however, that DATA0 is correctly received.

32.6.14.1.4 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 USBSTS[UI] (USB interrupt) to be set. 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. If USBINTR[UE] (USB interrupt enable) is set, a hardware interrupt is signaled to the system at the next interrupt threshold. If the completion is because of errors, USBSTS[UEI] (USB error interrupt) is also set.

32.6.14.1.5 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, USBSTS[UI] (USB interrupt bit) is set. If the USB interrupt enable bit is set (USBINTR[UE]), a hardware interrupt is signaled to the system at the next interrupt threshold.

32.6.14.2 Host controller event interrupts

These interrupt sources are independent of the interrupt threshold (with the one exception being the Interrupt on Async Advance.

32.6.14.2.1 Port change events

Port registers contain status and status change bits.

When the status change bits are set, the host controller sets the USBSTS[PCI]. If the port change interrupt enable bit (PCE) in the USBINTR register is set, the host controller issues a hardware interrupt. The port status change bits in PORTSC include:

- Connect change status (CSC)
- Port enable/disable change (PEC)
- Over-current change (OCC)
- Force port resume (FPR)

32.6.14.2.2 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 occurs every 1024 milliseconds, if it is 512, then it occurs every 512 milliseconds, and so on. When a frame list rollover is detected, the host controller sets the frame list rollover bit, USBSTS[FRI]. If USBINTR[FRE] is set (frame list rollover enable), the host controller issues a hardware interrupt. This interrupt is not delayed to the next interrupt threshold.

32.6.14.2.3 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 USBCMD[IAA]. If it is set, it sets USBSTS[AAI]. If USBINTR[AAE] is set, the host controller issues a hardware interrupt at the next interrupt threshold. A detailed explanation of this feature is described in Removing queue heads from asynchronous schedule.

32.6.14.2.4 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 making it impossible for the host controller to continue in a coherent fashion. Behavior for these types of errors is to halt the host controller. Host-based error must result in the following actions:

- USBCMD[RS] is cleared.
- USBSTS[SEI] and USBSTS[HCH] register are set
- If the host system error enable bit, USBINTR[SEE] is set, the host controller issues a hardware interrupt. This interrupt is not delayed to the next interrupt threshold.

The table below summarizes the required actions taken on the various host errors.

Table 32-45. Summary behavior on host system errors

Cycle type	Master abort	Target abort	Data phase parity
Frame list pointer fetch (read)	Fatal	Fatal	Fatal
siTD fetch (read)	Fatal	Fatal	Fatal
siTD status write-back (write)	Fatal	Fatal	Fatal
iTD fetch (read)	Fatal	Fatal	Fatal
iTD status write-back (write)	Fatal	Fatal	Fatal
qTD fetch (read)	Fatal	Fatal	Fatal
qHD status write-back (write)	Fatal	Fatal	Fatal
Data write	Fatal	Fatal	Fatal
Data read	Fatal	Fatal	Fatal

NOTE

After a host system error, software must reset the host controller using USBCMD[RST] before re-initializing and restarting the host controller.

32.7 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 section 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 USB DR module includes DCD software called the USB 2.0 Device API. The device API provides an easy to use Application Program Interface for developing device (peripheral) applications. The device API incorporates and abstracts for the application developer all of the elements of the program interface.

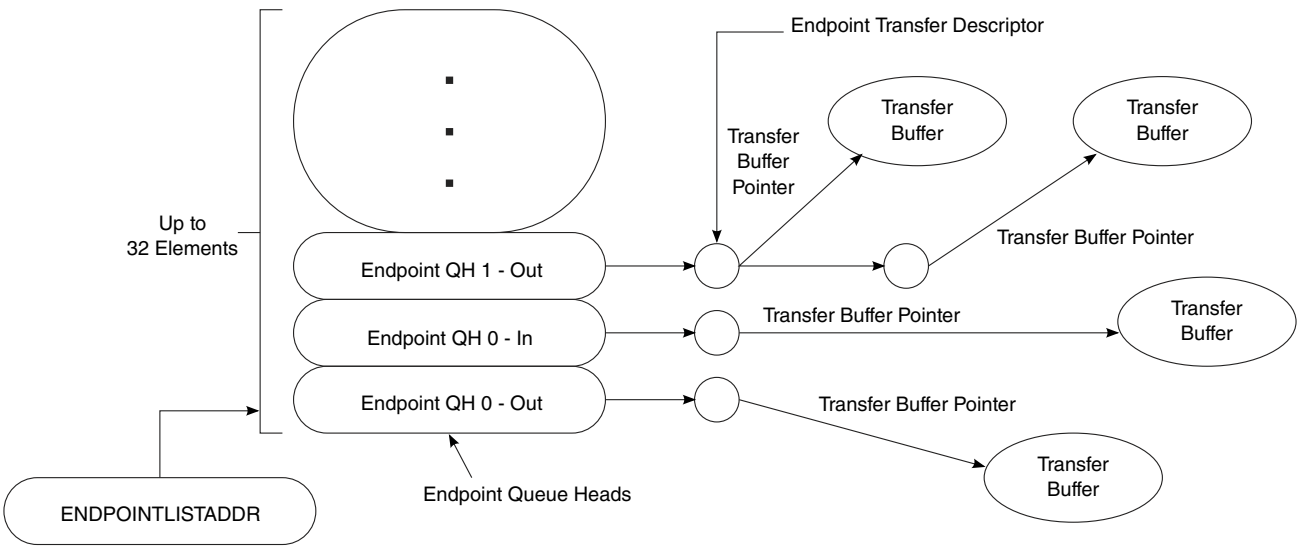


Figure 32-21. Endpoint queue head organization

32.7.1 Endpoint queue head

The device endpoint queue head (dQH) is where all transfers 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.

The figure below shows the endpoint queue head structure.

Table 32-46. Endpoint queue head 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	offset
Mult		zlt	00		Maximum Packet Length											ios	000_0000_0000_0000																	0x00
Current dTD Pointer[1]																											0_0000					0x04
Next dTD Pointer[1]																											0000				T[1]	0x08[2]
0	Total Bytes[1]															ioc[1]	000			MultO[1]		00		Status[1]								0x0C[2]
Buffer Pointer (Page 0)[1]																				Current Offset[1]												0x10[2]
Buffer Pointer (Page 1)[1]																				Reserved												0x14[2]
Buffer Pointer (Page 2)[1]																				Reserved												0x18[2]
Buffer Pointer (Page 3)[1]																				Reserved												0x1C[2]
Buffer Pointer (Page 4)[1]																				Reserved												0x20[2]
Reserved																																0x24
Setup Buffer Bytes 3-0[1]																																0x28
Setup Buffer Bytes 7-4[1]																																0x2C

1. Device controller read/write; all others read-only.
2. Offsets 0x08 through 0x20 contain the transfer overlay.

32.7.1.1 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.

The table below describes the Endpoint capabilities and characteristics fields.

Table 32-47. Endpoint capabilities/characteristics

Bits	Name	Description
31-30	Mult	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. **NOTE:** ISO Endpoints must set Mult = 01, 10, or 11 as needed.
29	zlt	Zero length termination select. This bit is used to indicate when a zero length packet is used to terminate transfers where the total transfer length is a multiple. This bit is not relevant for Isochronous transfers.

Table continues on the next page...

Table 32-47. Endpoint capabilities/characteristics (continued)

Bits	Name	Description
		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, should be cleared. These bit reserved for future use and should be cleared.
26-16	Maximum Packet Length	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	ios	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, should be cleared. Bits reserved for future use and should be cleared.

32.7.1.2 Transfer overlay

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 read, the dTD is 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.

32.7.1.3 Current dTD pointer

The current dTD pointer is used by the device controller to locate the transfer in progress.

This word is for USB_DR controller (hardware) use only and should not be modified by DCD software.

The table below describes the current dTD pointer fields.

Table 32-48. Current dTD pointer

Bits	Description
31-5	Current dtd. This field is a pointer to the dTD that is represented in the transfer overlay area. This field is modified by the Device Controller to next dTD pointer during endpoint priming or queue advance.
4-0	Reserved, should be cleared. Bit reserved for future use and should be cleared.

32.7.1.4 Setup buffer

The setup buffer is dedicated storage for the 8-byte data that follows a setup 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 table below describes the multiple mode control fields.

Table 32-49. Multiple mode control

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.

32.7.2 Endpoint transfer descriptor (dTD)

The dTD describes the location and quantity of data to be sent/received for given transfer to the device controller.

The DCD should not attempt to modify any field in an active dTD except the Next Link Pointer, which should only be modified as described in Managing transfers with transfer descriptors.

The figure below shows the endpoint transfer descriptor.

Table 32-50. 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	offset
Next Link Pointer																											0000				T	0x00
0	Total Bytes[1]															ioc	000			MultO		00		Status[1]								0x04

Table continues on the next page...

Table 32-50. Endpoint transfer descriptor (dTD) (continued)

Buffer Pointer (Page 0)	Current Offset[1]	0x08
Buffer Pointer (Page 1)	0 Frame Number[1]	0x0C
Buffer Pointer (Page 2)	0000_0000_0000	0x10
Buffer Pointer (Page 3)	0000_0000_0000	0x14
Buffer Pointer (Page 4)	0000_0000_0000	0x18

1. Device controller read/write; all others read-only.

The table below describes the next dTD pointer fields.

Table 32-51. Next dTD pointer

Bits	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, should be cleared. Bits reserved for future use and should be cleared.
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 table below describes the next dTD token fields.

Table 32-52. dTD token

Bits	Description
31	Reserved, should be cleared. Bit reserved for future use and should be cleared.
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 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 a 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, should be cleared. Bits reserved for future use and should be cleared.
11-10	Multiplier Override (MultO). This field can be used for transmit ISO's (that is, ISO-IN) to override the multiplier in the QH. This field must be zero for all packet types that are not transmit-ISO. Example: 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

Table continues on the next page...

Table 32-52. dTD token (continued)

Bits	Description
	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, should be cleared. Bits reserved for future use and should be cleared.
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, should be cleared

The following tables describe the buffer pointer page *n* fields.

Table 32-53. Buffer pointer page 0

Bits	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.
11-0	Current Offset. Offset into the 4kb buffer where the packet is to begin.

Table 32-54. Buffer pointer page 1

Bits	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.
11	Reserved
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.

Table 32-55. Buffer pointer pages 2-4

Bits	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.
11-0	Reserved

32.8 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.

32.8.1 Device controller initialization

After hardware reset, the USB DR module is disabled until the run/stop bit (USBCMD[RS]) 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.

To configure the external ULPI PHY the following initialization sequence is required.

In order to initialize a device, the software should perform the following steps:

1. Set the controller mode to device mode. Optionally set USBMODE[SDIS] (streaming disable).

NOTE

Transitioning from host mode to device mode requires a device controller reset before modifying USBMODE.

2. Program PORTSC[PTS] if using a non-ULPI PHY.
3. Set CONTROL[USB_EN]
4. Allocate and initialize device queue heads in system memory Minimum: Initialize device queue heads 0 Tx and 0 Rx.

NOTE

All device queue heads must be initialized for control endpoints before the endpoint is enabled. Device queue heads for non-control endpoints must be initialized before the endpoint can be used.

For information on device queue heads, refer to Device data structures.

5. Configure the ENDPOINTLISTADDR pointer.

   For additional information on ENDPOINTLISTADDR, refer to the register table.

6. Enable the microprocessor interrupt associated with the USB DR module and optionally change setting of USBCMD[ITC].

   Recommended: enable all device interrupts including: USBINT, USBERRINT, Port Change Detect, USB Reset Received, DCSuspend.

   For a list of available interrupts refer to the USBINTR and the USBSTS register tables.

7. Set USBCMD[RS] to run mode.

   After the run bit is set, a device reset will occur. The DCD must monitor the reset event and set the DEVICEADDR register, set the ENDPTCTRLx registers, and adjust the software state as described in Bus reset.

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 initially prime Endpoint 0 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 command set.

32.8.2 Port state and control

From a chip or system reset, the USB_DR controller enters the powered state.

A transition from the powered state to the attach state occurs when the run/stop bit (USBCMD[RS]) 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 protocol reset described in Appendix C.2 of the *USB Specification Rev. 2.0*. The figure below depicts the state of a USB 2.0 device.

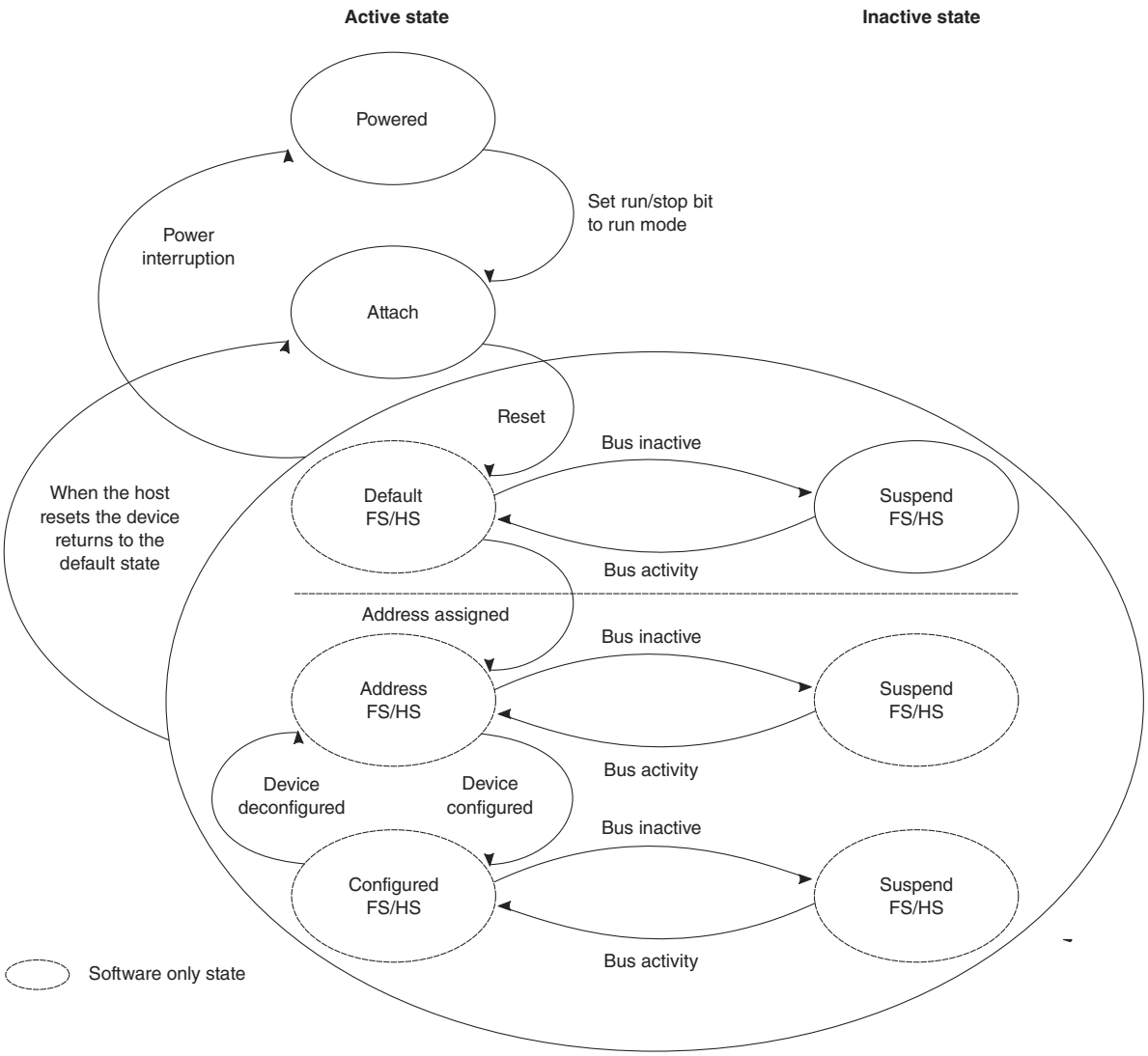


Figure 32-22. USB 2.0 device states

States Powered, Attach, DefaultFS/HS, SuspendFS/HS are implemented in the USB_DR controller and are communicated to the DCD using status bits, as shown in the table below:

Table 32-56. Device controller state information bits

Bits	Register
DCSuspend (SLI)	USBSTS
USB Reset Received (URI)	USBSTS
Port Change Detect (PCI)	USBSTS
High-Speed Port	PORTSC

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 ENDPTCTRL*n* registers and initializing the associated queue heads.

32.8.2.1 Bus reset

A bus reset is used by the host to initialize downstream devices.

When a bus reset is detected, the USB_DR controller will renegotiate its attachment speed, reset the device address to 0, and notify the DCD by interrupt (assuming the USB reset interrupt enable bit, USBINTR[URE], is set). After a reset is received, all endpoints (except endpoint 0) are disabled and any primed transactions are canceled by the device controller. The concept of priming is clarified below, but the DCD must perform the following tasks when a reset is received:

1. Clear all setup token semaphores by reading the ENDPTSETUPSTAT register and writing the same value back to the ENDPTSETUPSTAT register.
2. Clear all the endpoint complete status bits by reading the ENDPTCOMPLETE register and writing the same value back to the ENDPTCOMPLETE register.
3. Cancel all primed status by waiting until all bits in the ENDPTPRIME are 0 and then writing 0xFFFF_FFFF to ENDPTFLUSH.
4. Read the reset bit in the PORTSC register (PORTSC[PR]) and make sure that it is still active.
    - A USB reset occurs for a minimum of 3 ms, and the DCD must reach this point in the reset cleanup before end of the reset occurs.
    - If it does not, a hardware reset of the device controller is recommended. A hardware reset can be performed by writing a one to the USB_DR controller reset bit in (USBCMD[RST]). Note that a hardware reset will cause the device to detach from the bus by clearing USBCMD[RS] bit. Thus, the DCD must completely re-initialize the USB_DR controller after a hardware reset.
5. 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 PORTSC 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.

32.8.2.2 Suspend/resume

This section discusses the suspend and resume functions.

32.8.2.2.1 Suspend description

In order to conserve power, USB_DR controller automatically enters the suspended state when no bus traffic has been observed for a specified period.

When suspended, the USB_DR controller 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.

The USB_DR controller exits suspend mode when there is bus activity. It may also request the host to exit suspend mode or selective suspend by using electrical signaling to indicate remote wake-up. The ability of a device to signal remote wake-up is optional. The USB_DR controller is capable of remote wake-up signaling. When the USB_DR controller is reset, remote wake-up signaling must be disabled.

32.8.2.2.2 Suspend operational model

The USB_DR 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 USBSTS[SLI] (device controller suspend) is set, 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.

32.8.2.2.3 Resume

If the USB_DR controller is suspended, its operation is resumed when any non-idle signaling is received on its upstream facing port.

In addition, the USB_DR controller 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 PORTSC[FPR] (resume bit) 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 or 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.

32.8.3 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_DR controller supports up to six endpoint specified numbers. The DCD can enable, disable, and configure each endpoint.

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, for example, is always a control endpoint and uses the pair of directions.

Each endpoint direction requires a queue head allocated in memory. If the maximum of 6 endpoint numbers, one for each endpoint direction are being used by the device controller, then 12 queue heads are required. The operation of an endpoint and use of queue heads are described later in this document.

32.8.3.1 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 ENDPTCTRL*n* register. Each 32-bit ENDPTCTRL*n* is split into an upper and lower half. The lower half of ENDPTCTRL*n* 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 ENDPTCTRL*n* register otherwise the behavior is undefined. The table below shows how to construct a configuration word for endpoint initialization.

Table 32-57. 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

32.8.3.1.1 Stalling

There are two occasions where the USB_DR 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 (Chapter 9). 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 ENDPTCTRL*n* 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 ENDPTCTRL*n* register can ensure that both stall bits are set at the same instant.

NOTE

Any write to the ENDPTCTRL*n* register during operational mode must preserve the endpoint type field (that is, perform a read-modify-write).

The table below describes the device controller stall response matrix.

Table 32-58. 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

32.8.3.2 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 *Universal Serial Bus Revision 2.0 Specification*.

32.8.3.2.1 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 ENDPTCTRL*n* register.

This should only be necessary when configuring/initializing an endpoint or returning from a STALL condition.

32.8.3.2.2 Data toggle inhibit

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 USB_DR 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 USB_DR controller checks the DATA0/DATA1 bit against the data toggle state bit 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 USB_DR 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.

32.8.3.3 Device 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 *Universal Serial Bus Revision 2.0 Specification*.

A USB host will send requests to the USB_DR controller in an order that cannot 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 2 (transmit direction) is configured as a bulk pipe, then the host sends IN requests to that endpoint. This USB_DR controller prepares 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 is used throughout the following documentation to describe the

USB_DR 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.

32.8.3.3.1 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 is 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.

After a priming request is complete, an endpoint state of primed is indicated in the 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.

Since 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 has been sized to account for the maximum latency that can be incurred by the system memory bus.

32.8.3.3.2 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.

32.8.3.4 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 is 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 the following tables 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 32-59. 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 32-60. 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 is 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 is cleared and the next pointer will be followed when the Terminate bit is clear. When the Terminate bit is set, the USB_DR 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 is left pointing to the dTD that was in error. In order to recover from this error condition, the DCD must properly re-initialize 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.

32.8.3.4.1 Interrupt/bulk endpoint bus response matrix

The table below shows the interrupt/bulk endpoint bus response matrix.

Table 32-61. 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[1]	N/A
Out	STALL	NAK	Receive + NYET/ACK[2]	N/A	NAK
Ping	STALL	NAK	ACK	N/A	N/A
Invalid	Ignore	Ignore	Ignore	Ignore	Ignore

1. Force Bit Stuff Error.
2. 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.

32.8.3.5 Control endpoint operation model

This section discusses the control endpoint operation model.

32.8.3.5.1 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 USB_DR 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 tripwire type semaphore will ensure that the setup packet payload is extracted from the queue head without being corrupted by 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.

Setup Packet Handling

- Disable Setup Lockout by writing '1' to Setup Lockout Mode (SLOM) in 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 a potential compliance issue.

- After receiving an interrupt and inspecting ENDPTSETUPSTAT to determine that a setup packet was received on a particular pipe:
  - Write '1' to clear corresponding bit ENDPTSETUPSTAT.
  - Write '1' to Setup Tripwire (SUTW) in USBCMD register.
  - Duplicate contents of dQH.SetupBuffer into local software byte array.
  - Read Setup TripWire (SUTW) in USBCMD register. (if set-continue; if cleared-goto 2)
  - Write '0' to clear Setup Tripwire (SUTW) in USBCMD register.
  - 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 and de-allocated before linking a new status and/or handshake dTD for the most recent setup packet.

32.8.3.5.2 Data phase

If the control transfer requires a data stage 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 ENDPTSETUPSTAT register immediately verifying that the prime had completed. A prime will complete when the associated bit in the ENDPTPRIME register is zero and the associated bit in the ENDPTSTATUS register is a one. If a prime fails, that is, The ENDPTPRIME bit goes to zero and the ENDPTSTATUS 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 (ENDPTSTATUS) 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.

NOTE

Error handling of data phase packets is the same as bulk packets described previously.

32.8.3.5.3 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 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.

NOTE

Error handling of data phase packets is the same as bulk packets described previously.

32.8.3.5.4 Control endpoint bus response matrix

The table below shows the device controller response to packets on a control endpoint, according to the device controller state.

Table 32-62. Control endpoint bus response matrix

Token type	Endpoint state					Setup lockout
	Stall	Not primed	Primed	Underflow	Overflow	
Setup	ACK	ACK	ACK	N/A	SYSERR[1]	
In	STALL	NAK	Transmit	BS Error[2]	N/A	N/A
Out	STALL	NAK	Receive + NYET/ACK[3]	N/A	NAK	N/A
Ping	STALL	NAK	ACK	N/A	N/A	N/A
Invalid	Ignore	Ignore	Ignore	Ignore	Ignore	Ignore

1. SYSERR-System error should never occur when the latency FIFOs are correctly sized and the DCD is responsive.
2. Force Bit Stuff Error.
3. NYET/ACK-NYET unless the Transfer Descriptor has packets remaining according to the USB variable length protocol then ACK.

32.8.3.6 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 USB_DR controller is accomplished by the following:

- Exactly MULT Packets per (micro)Frame are transmitted/received. Note that 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 is held ready until executed or canceled by the DCD.

The USB_DR controller in host mode 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 is 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. 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
- 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.

32.8.3.6.1 Isochronous pipe synchronization

When it is necessary to synchronize an isochronous data pipe to the host, the (micro)frame number (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 FRINDEX = N - 1, the DCD must write the prime bit. The USB_DR 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.

32.8.3.6.2 Isochronous endpoint bus response matrix

The table below shows the isochronous endpoint bus response matrix.

Table 32-63. Isochronous endpoint bus response matrix

	Stall	Not Primed	Primed	Underflow	Overflow
Setup	STALL	STALL	STALL	N/A	N/A
In	NULL[1] Packet	NULL Packet	Transmit	BS Error[2]	N/A
Out	Ignore	Ignore	Receive	N/A	Drop Packet
Ping	Ignore	Ignore	Ignore	Ignore	Ignore
Invalid	Ignore	Ignore	Ignore	Ignore	Ignore

1. Zero Length Packet.
2. Force Bit Stuff Error.

32.8.4 Managing queue heads

The figure below shows the endpoint queue head diagram.

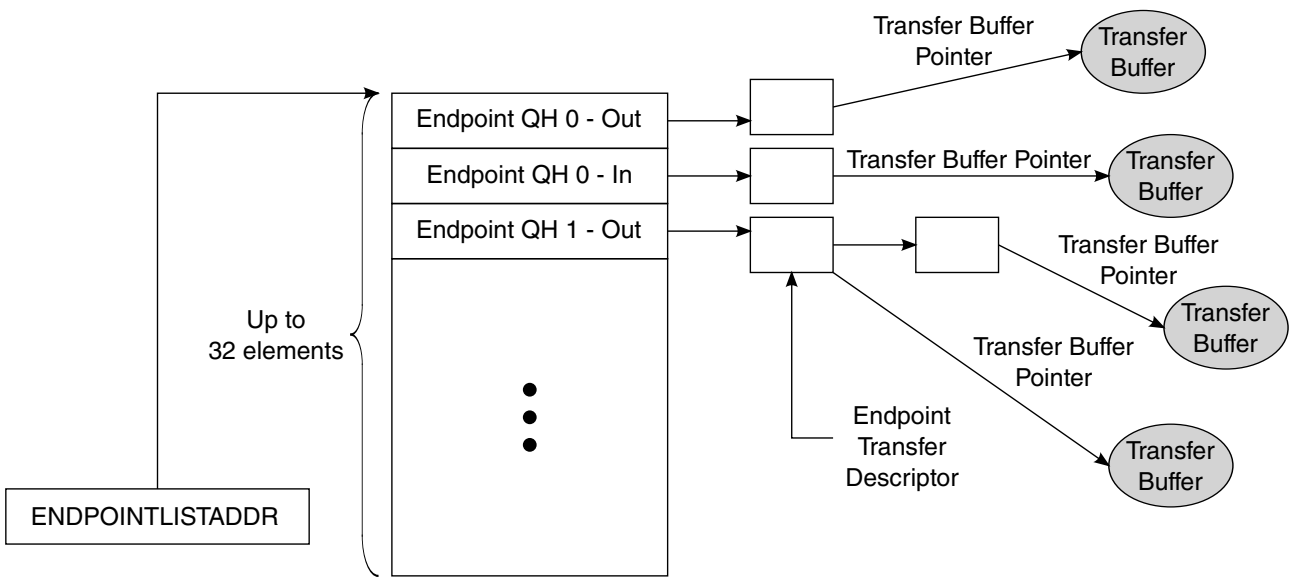


Figure 32-23. Endpoint queue head diagram

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 ENDPOINTLISTADDR contains a group of all dQHs in a sequential list as shown in Figure 32-23. 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 since pointers will no longer exist within the queue head once the dTD is retired (see Software link pointers).

In addition to the current and next pointers and the dTD overlay, 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.

32.8.4.1 Queue head initialization

One pair of device queue heads 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 that 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 dTDs.

32.8.4.2 Operational model for setup transfers

As discussed in 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.

NOTE

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 Flushing/depriming an endpoint.

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.

4. Decode setup packet and prepare data phase [optional] and status phase transfer as required by the USB Chapter 9 or application specific protocol.

32.8.5 Managing transfers with transfer descriptors

This section describes software link pointers, transfer descriptors, transfer completion and the device error matrix.

32.8.5.1 Software link pointers

It is necessary for the DCD software to maintain head and tail pointers 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.

NOTE

To conserve memory, the reserved fields at the end of the dQH can be used to store the Head and Tail pointers but it still remains the responsibility of the DCD to maintain the pointers.

The figure below shows the software link pointers.

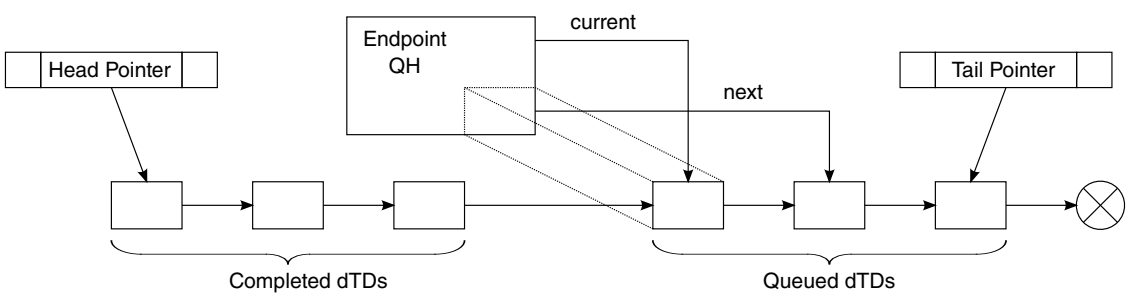


Figure 32-24. Software link pointers

32.8.5.2 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 seven 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.

32.8.5.3 Executing a transfer descriptor

To safely add a dTD, the DCD must account for the event in which 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.

First, determine whether the link list is empty by checking the DCD driver to see if the pipe is empty (internal representation of linked-list should indicate if any packets are outstanding). Then follow the sequence of actions in the following list as appropriate, depending on whether the link list is empty or not empty.

Case 1: Link list is empty

1. Write dQH next pointer AND dQH terminate bit to '0' as a single DWord operation.
2. Clear active and halt bit in dQH (in case set from a previous error).
3. Prime endpoint by writing '1' to correct bit position in ENDPTPRIME.

Case 2: Link list is not empty

1. Add dTD to end of linked list.

2. Read correct prime bit in ENDPTPRIME-if '1' DONE.
3. Set ATDTW bit in USBCMD register to '1'.
4. Read correct status bit in ENDPTSTATUS. (store in tmp. variable for later).
5. Read ATDTW bit in USBCMD register.

   If '0' goto 3.

   If '1' continue to 6.

6. Write ATDTW bit in USBCMD register to '0'.
7. If status bit read in (4) is '1' DONE.
8. If status bit read in (4) is '0' then Goto Case 1: Step 1.

32.8.5.4 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 is 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.

32.8.5.5 Flushing/depriming an endpoint

It is necessary for the DCD to flush to deprime one or 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 ENDPTFLUSH.
2. Wait until all bits in ENDPTFLUSH are '0'.
3. 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.
4. Read ENDPTSTATUS 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 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.

32.8.5.6 Device error matrix

The table below summarizes packet errors that are not automatically handled by the USB controller.

Table 32-64. 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 provides the error descriptions.

Table 32-65. Error descriptions

Overflow	Number of bytes received exceeded max. packet size or total buffer length. **NOTE:** This error also sets the Halt bit in the dQH. If there are dTDs remaining in the linked list for the endpoint, they 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.

32.8.6 Servicing interrupts

The interrupt service routine must consider that there are high-frequency, low-frequency operations, and error operations and order accordingly.

32.8.6.1 High-frequency interrupts

High frequency interrupts in particular should be handed in the order shown in the table below. The most important of these is listed first because the DCD must acknowledge a setup buffer in the timeliest manner possible.

Table 32-66. Interrupt handling order

Execution order	Interrupt	Action
1a	USB Interrupt[1] ENDPTSETUPSTATUS	Copy contents of setup buffer and acknowledge setup packet (as indicated in Managing queue heads). Process setup packet according to USB 2.0 Chapter 9 or application specific protocol.
1b	USB Interrupt ENDPTCOMPLETE	Handle completion of dTD as indicated in Managing queue heads.
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.

32.8.6.2 Low-frequency interrupts

The low frequency events include the interrupts shown in the table below.

These interrupts can be handled in any order because they do not occur often in comparison to the high-frequency interrupts.

Table 32-67. 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.

32.8.6.3 Error interrupts

Error interrupts are the least frequently occurring events. They should be placed last in the interrupt service routine.

The table below shows the error interrupt events.

Table 32-68. 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/ ENDPTCOMPLETE).
System Error	Unrecoverable error. Immediate Reset of core; free transfers buffers in progress and restart the DCD.

32.9 Deviations from the EHCI specifications

The host mode operation of the USB DR module is nearly EHCI-compatible with few minor differences. For the most part, the module conforms to the data structures and operations described in Section 3, "Data Structures," and Section 4, "Operational Model," in the EHCI specification. The particulars of the deviations occur in the following areas:

- 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-The module does not have a PCI interface and therefore the PCI configuration registers described in the EHCI specification are not applicable.

32.9.1 Embedded transaction translator function

The DR module 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.

32.9.1.1 Capability registers

The following additions have been added to the capability registers to support the embedded transaction translator function:

- N_TT added to HSCPARAMS-Host Controller Structural Parameters
- N_PTT added to HSCPARAMS-Host Controller Structural Parameters

See USB_HCSPARAMS for usage information.

32.9.1.2 Operational registers

The following additions have been added to the operational registers to support the embedded TT:

- ASYNCTTSTS is a new register.
- Addition of two-bit Port Speed (PSPD) to the PORTSC register.

32.9.1.3 Discovery differences

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).

The module always sets the port enable after the port reset operation regardless of the result of the host device chirp result. The resulting port speed is indicated by the PSPD field in PORTSC. Therefore, the standard EHCI host controller driver requires an alteration to handle directly connected full- and low-speed devices or hubs. The table below summarizes the functional differences between EHCI and EHCI with embedded TT.

Table 32-69. Functional differences between EHCI and EHCI with embedded TT

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 PORTSC.
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 PORTSC.
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 (that is, 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 (that is, Split target hub is the root hub)]

32.9.1.4 Data structures

The same data structures used for FS/LS transactions though a HS hub are also used for transactions through the root hub.

The following list demonstrates 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 PORTSC[PSPD] = 00 (FS), a LS-pre-pid is sent before the transmitting LS traffic.

Maximum Packet Size must be less than or equal 64 or undefined behavior 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 behavior may result.

32.9.1.5 Operational model

The operational models are well defined for the behavior of the transaction translator (see *Universal Serial Bus Revision 2.0 Specification*) and for the EHCI controller moving packets between system memory and a USB-HS hub.

Since the embedded transaction translator exists within the DR module 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.

32.9.1.5.1 Microframe 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 *Universal Serial Bus Revision 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 microframe 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.

NOTE

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 are 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 cannot babble through the SOF (start of B-frame 0).

32.9.1.5.2 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 table below summarizes the conditions where handshakes are emulated from internal state instead of actual handshakes to HS split bus traffic.

Table 32-70. Emulated handshakes

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 FS/LS device]

32.9.1.5.3 Asynchronous transaction scheduling and buffer management

The following *Universal Serial Bus Revision 2.0 Specification* items are implemented in the embedded transaction translator:

- USB 2.0-11.17.3
  - Sequencing is provided and a packet length estimator ensures no full-speed/low-speed packet babbles into SOF time.
- USB 2.0-11.17.4
  - Transaction tracking for 2 data pipes.
- USB 2.0-11.17.5
  - Clear_TT_Buffer capability provided

32.9.1.5.4 Periodic transaction scheduling and buffer management

The following *Universal Serial Bus Revision 2.0 Specification* items are implemented in the embedded transaction translator:

- USB 2.0-11.18.6.[1-2]
  - Abort of pending start-splits
    - EOF (and not started in microframes 6)
    - Idle for more than 4 microframes
  - Abort of pending complete-splits
    - EOF
    - Idle for more than 4 microframes

NOTE

There is no data schedule mechanism for these transactions other than the microframe pipeline. The embedded TT assumes the number of packets scheduled in a frame does not exceed the frame duration (1 msec) or else undefined behavior may result.

32.9.1.5.5 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 HCSPARAMS register.

See USB_HCSPARAMS for more information.

32.9.2 Device operation

The co-existence of a device operational controller within the DR module has little effect on EHCI compatibility for host operation except as noted in this section.

32.9.3 Non-zero fields the register file

Some of the reserved fields and reserved addresses in the capability registers and operational registers have use in device mode, the following must be adhered to:

- Write operations to all EHCI reserved fields (some of which are device fields in the DR module) in 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 module must properly mask EHCI reserved fields (some of which are device fields in the DR module registers).

32.9.4 SOF interrupt

The SOF interrupt is a free running 125 μsec interrupt for host mode.

EHCI does not specify this interrupt, but it has been added for convenience and as a potential software time base. The free running interrupt is shared with the device-mode start-of-frame interrupt. See USB_USBSTS and USB_USBINTR for more information.

32.9.5 Embedded design

This is an Embedded USB Host Controller as defined by the EHCI specification and thus does not implement the PCI configuration registers.

32.9.5.1 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 those provided by the Frame Adjust register in the PCI configuration registers.

Starts of microframes are timed precisely to 125 μsec using the transceiver clock as a reference clock. That is, 60-MHz transceiver clock for 8-bit physical interfaces and full-speed serial interfaces or 30-MHz transceiver clock.

32.9.6 Miscellaneous variations from EHCI

The modules support multiple physical interfaces which can operate in different modes when the module is configured with the software programmable Physical Interface Modes.

The control bits for selecting the PHY operating mode have been added to the PORTSC register providing a capability that is not defined by the EHCI specification.

32.9.6.1 Discovery

This section discusses port reset and port speed detection.

32.9.6.1.1 Port reset

The port connect methods specified by EHCI require setting the port reset bit in the register for a duration of 10 msec.

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 10 msec 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 msec.
    - 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.

32.9.6.1.2 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-HS 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 HS vs. FS/LS

Chapter 33
Universal Serial Bus Interface 3.0

33.1 Overview

The USB module is a USB 3.0-compliant serial interface engine for implementing a USB interface. This module may be connected to an external port. Collectively the module and external port are called the USB 3.0 interface. USB 3.0 supports super-speed (SS), high-speed (HS), full-speed (FS), and low-speed (LS) operations.

- The upper layer is common for USB 2.0 and USB 3.0 operation. This has the bus interface, buffer management block, list processor for scheduling, and control and status register (CSR) functions
- USB 2.0 PHY and MAC layers
- USB 3.0 PHY, LINK, and MAC layers

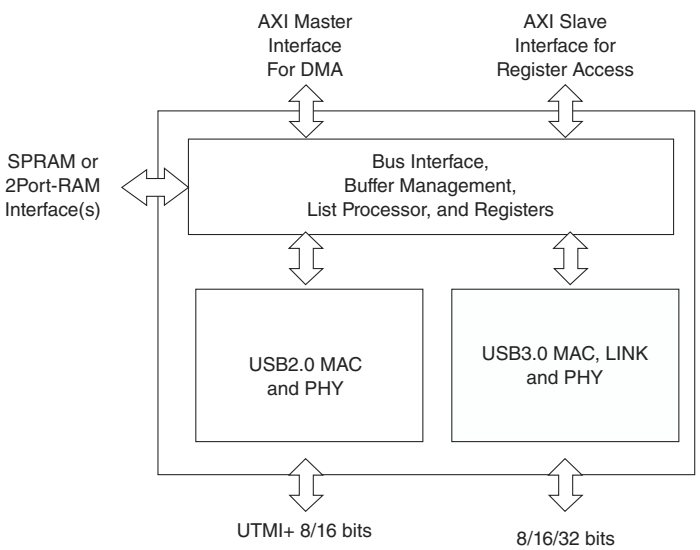


Figure 33-1. USB interface block diagram

33.1.1 Features

The USB 3.0 module includes the following features:

- Complies with USB specification rev 3.0 (xHCI compatible)
- Supports operation as a standalone USB host controller
- USB dual-role operation and can be configured as host or device
- Super-speed (5 Gbit/s), high-speed (480 Mbit/s), full-speed (12 Mbit/s), and low-speed (1.5 Mbit/s) operations.
- Supports operation as a standalone single port USB
- Supports four programmable, bidirectional USB endpoints
- OTG (on-the-go) 2.0 compliant, which includes both device and host capability. Super-speed operation is not supported when OTG is enabled.
- Supports system memory interface with 40-bit addressing capability
- Supports USB PHY power gating

33.1.2 Modes of Operation

The USB 3.0 module operates in following modes.

- Host Mode: SS/HS/FS/LS
- Device Mode: SS/HS/FS
- OTG: HS/FS/LS

33.1.3 External Signals

This section contains detailed descriptions of all the USB 3.0 controller signals. Many of the signals for the PHY interfaces are muxed onto the same pins in order to reduce pin count. The following table shows the USB signals, indicating which interface supports each signal.

Table 33-1. USB 3.0 External Signals

Signal	I/O	Description
USB1_D_P	IO	USB PHY Data Plus
USB1_D_M	IO	USB PHY Data Minus
USB1_VBUS	IO	USB PHY VBUS
USB1_ID	IO	USB PHY ID Detect
USB1_TX_P	O	USB PHY 3.0 Transmit Data (positive)
USB1_TX_M	O	USB PHY 3.0 Transmit Data (negative)
USB1_RX_P	I	USB PHY 3.0 Receive Data (positive)

Table continues on the next page...

Table 33-1. USB 3.0 External Signals (continued)

Signal	I/O	Description
USB1_RX_M	I	USB PHY 3.0 Receive Data (negative)
USB1_RESREF	IO	USB PHY Impedance Calibration
USB1_DRVVBUS	O	VBus power enable.
USB1_PWRFAULT	I	Indicates that a Vbus fault has occurred.

33.2 USB3.0 register descriptions

USB3.0 registers are 32-bit wide, and the addresses are 32-bit block aligned. To avoid hardware or software incompatibility, the driver must access the registers as 32-bit units. The driver must not access these registers as 8 bit or 16 bit. CSRs are classified as follows:

Table 33-2. Registers

Control and status segisters	Reference
Global registers	Capability registers length and HC interface version number (CAPLENGTH) to Global frame length adjustment register (GFLADJ)
Device register	Device configuration register (DCFG) to Device physical endpoint-n command parameter 0 register (DEPCMDPAR0_0 - DEPCMDPAR0_7)
OTG register	OTG configuration register (OCFG) to ADP event enable register (ADPEVTEN)
xHCI host register	See xHCI Spec for details. Refer Table 33-3 for list of registers.

The following table lists the supported xHCI registers. For register definitions, refer to the xHCI specification.

Table 33-3. Host registers

Offset	Register	Offset	Register
Register Set's Base= 0x0000		Register Set's Base= DBOFF	
0x0000	CAPLENGTH	0x0000	DB
0x0004	HCSPARAMS1	Register Set's Base= xECP * 4	
0x0008	HCSPARAMS2	0x0000	USBLEGSUP
0x000C	HCSPARAMS3	0x0004	USBLEGCTLSTS
0x0010	HCCPARAMS	Addr1	
0x0014	DBOFF	0x0000	SUPTPRT2_DW0
0x0018	RTSOFF	0x0004	SUPTPRT2_DW1
0x001C	Reserved	0x0008	SUPTPRT2_DW2

Table continues on the next page...

Table 33-3. Host registers (continued)

Offset	Register	Offset	Register
0x0020	USBCMD	0x000C	SUPTPRT2_DW3
0x0024	USBSTS	Addr2	
0x0028	PAGESIZE	0x0000	SUPTPRT3_DW0
0x002C-0x0033	Reserved	0x0004	SUPTPRT3_DW1
0x0034	DNCTRL	0x0008	SUPTPRT3_DW2
0x0038	CRCR	0x000C	SUPTPRT3_DW3
0x003C	CRCR	Addr2 + 10h	
0x0040-0x004F	Reserved	0x0000	DCID
0x0050	DCBAAP	0x0004	DCDB
0x0054	DCBAAP	0x0008	DCERSTSZ
0x0058	CONFIG	0x000C	Reserved
0x005C-0x041F	Reserved	0x0010	DCERSTBA
0x0420	PORTSC PORTPMSC_SS1	0x0014	DCERSTBA
0x0424	PORTPMSC_201	0x0018	DCERDP
0x0428	PORTLI	0x001C	DCERDP
0x042C	PORTHLPMC_SS2	0x0020	DCCTRL
0x042C	PORTHLPMC_202	0x0024	DCST
Register Set's Base= RTSOFF		0x0028	DCPORTSC
0x0000	MFINDEX	0x002C	Reserved
0x0004	Reserved	0x0030	DCCP
Register Set's Base= RTSOFF + 20h		0x0034	DCCP
0x0000	IMAN	0x0038	DCDDI1
0x0004	IMOD	0x003C	DCDDI2
0x0008	ERSTSZ		
0x000C	Reserved		
0x0010	ERSTBA		
0x0014	ERSTBA		
0x0018	ERDP		
0x001C	ERDP		

33.2.1 USB3 Memory map

USB3 base address: 2F0_0000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Capability registers length and HC interface version number (CAPLENGTH)	32	RO	0100_0020h
4h	Host controller structural parameters 1 (HCSPARAMS1)	32	RO	0200_017Fh
8h	Host controller structural parameters 2 (HCSPARAMS2)	32	RO	1400_00F1h
Ch	Host controller structural parameters 3 (HCSPARAMS3)	32	RO	07FF_000Ah
10h	Host controller capability parameters 1 (HCCPARAMS1)	32	RO	0220_F66Dh
14h	Doorbell offset (DBOFF)	32	RO	0000_0480h
18h	Runtime register space offset (RTSOFF)	32	RO	0000_0440h
1Ch	Host controller capability parameters 2 (HCCPARAMS2)	32	RO	0000_000Bh
C100h	Global SoC bus configuration register 0 (GSBUSCFG0)	32	RW	0010_0080h
C104h	Global SoC bus configuration register 1 (GSBUSCFG1)	32	RW	0000_0700h
C108h	Global Tx threshold control register (GTXTHRCFG)	32	RW	0000_0000h
C10Ch	Global Rx threshold control register (GRXTHRCFG)	32	RW	0000_0000h
C110h	Global core control register (GCTL)	32	RW	30C1_3004h
C118h	Global status register (GSTS)	32	RW	3E80_0000h
C11Ch	Global user control register 1 (GUCTL1)	32	RW	0004_018Ah
C128h	Global user ID register (GUID)	32	RW	0130_290Ah
C12Ch	Global user control register (GUCTL)	32	RW	0200_8010h
C130h	Global SoC bus error address register low (GBUSERRADDRLO)	32	RO	0000_0000h
C134h	Global SoC bus error address register high (GBUSERRADDRHI)	32	RO	0000_0000h
C138h	Global SS port to bus instance mapping register - low (GPRTBIMAPLO)	32	RW	0000_0000h
C13Ch	Global SS port to bus instance mapping register - high (GPRTBIMAPHI)	32	RW	0000_0000h
C140h	Global hardware parameters register 0 (GHWPARAMS0)	32	RO	4020_400Ah
C144h	Global hardware parameters register 1 (GHWPARAMS1)	32	RO	81E0_C93Bh
C148h	Global hardware parameters register 2 (GHWPARAMS2)	32	RO	0130_290Ah
C14Ch	Global hardware parameters register 3 (GHWPARAMS3)	32	RO	0410_8485h
C150h	Global hardware parameters register 4 (GHWPARAMS4)	32	RO	4782_0004h
C154h	Global hardware parameters register 5 (GHWPARAMS5)	32	RO	0420_4108h
C158h	Global hardware parameters register 6 (GHWPARAMS6)	32	RO	0904_9C20h
C15Ch	Global hardware parameters register 7 (GHWPARAMS7)	32	RO	0308_044Dh
C180h	Global high-speed port to bus instance mapping register - low (GPRTBIMAP_HSLO)	32	RW	0000_0000h
C184h	Global high-speed port to bus instance mapping register - high (GPRTBIMAP_HSHI)	32	RW	0000_0000h
C200h	Global USB2 PHY configuration register (GUSB2PHYCFG)	32	RW	4010_2400h
C2C0h	Global USB 3.0 PIPE control register (GUSB3PIPECTL)	32	RW	010C_0002h
C300h	Global transmit FIFO size register (GTXFIFOSIZ_0)	32	RW	0000_0042h
C310h	Global transmit FIFO size register (GTXFIFOSIZ_1)	32	RW	0000_0042h
C320h	Global transmit FIFO size register (GTXFIFOSIZ_2)	32	RW	0000_0042h

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
C330h	Global transmit FIFO size register (GTXFIFOSIZ_3)	32	RW	0000_0042h
C380h	Global receive FIFO size register (GRXFIFOSIZ_0)	32	RW	0000_0305h
C390h	Global receive FIFO size register (GRXFIFOSIZ_1)	32	RW	0000_0305h
C3A0h	Global receive FIFO size register (GRXFIFOSIZ_2)	32	RW	0000_0305h
C400h	Global event buffer address (low) register (GEVNTADRLO)	32	RW	0000_0000h
C404h	Global event buffer address (high) register (GEVNTADRHI)	32	RW	0000_0000h
C408h	Global event buffer size register (GEVNTSIZ)	32	RW	0000_0000h
C40Ch	Global event buffer count register (GEVNTCOUNT)	32	RW	0000_0000h
C600h	Global hardware parameters register 8 (GHWPARAMS8)	32	RO	0000_0904h
C610h	Global device TXFIFO DMA priority register (GTXFIFOPRIDEV)	32	RW	0000_0000h
C618h	Global host TXFIFO DMA priority register (GTXFIFOPRIHST)	32	RW	0000_0000h
C61Ch	Global host RXFIFO DMA priority register (GRXFIFOPRIHST)	32	RW	0000_0000h
C624h	Global host FIFO DMA high-low priority ratio register (GDMAHLRATIO)	32	RW	0000_0808h
C630h	Global frame length adjustment register (GFLADJ)	32	RW	0000_0000h
C700h	Device configuration register (DCFG)	32	RW	0008_0804h
C704h	Device control register (DCTL)	32	RW	00F0_0000h
C708h	Device event enable register (DEVTEN)	32	RW	0000_0000h
C70Ch	Device status register (DSTS)	32	RO	0052_0004h
C710h	Device generic command parameter register (DGCMDPAR)	32	RW	0000_0000h
C714h	Device generic command register (DGCMD)	32	RW	0000_0000h
C720h	Device active USB endpoint enable register (DALEPENA)	32	RW	0000_0000h
C800h	Device physical endpoint-n command parameter 2 register (DEPCMDPAR2_0)	32	RW	0000_0000h
C804h	Device physical endpoint-n command parameter 1 register (DEPCMDPAR1_0)	32	RW	0000_0000h
C808h	Device physical endpoint-n command parameter 0 register (DEPCMDPAR0_0)	32	RW	0000_0000h
C80Ch	Device physical endpoint-n command register (DEPCMD_0)	32	RW	0000_0000h
C810h	Device physical endpoint-n command parameter 2 register (DEPCMDPAR2_1)	32	RW	0000_0000h
C814h	Device physical endpoint-n command parameter 1 register (DEPCMDPAR1_1)	32	RW	0000_0000h
C818h	Device physical endpoint-n command parameter 0 register (DEPCMDPAR0_1)	32	RW	0000_0000h
C81Ch	Device physical endpoint-n command register (DEPCMD_1)	32	RW	0000_0000h
C820h	Device physical endpoint-n command parameter 2 register (DEPCMDPAR2_2)	32	RW	0000_0000h
C824h	Device physical endpoint-n command parameter 1 register (DEPCMDPAR1_2)	32	RW	0000_0000h
C828h	Device physical endpoint-n command parameter 0 register (DEPCMDPAR0_2)	32	RW	0000_0000h

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
C82Ch	Device physical endpoint-n command register (DEPCMD_2)	32	RW	0000_0000h
C830h	Device physical endpoint-n command parameter 2 register (DEPCMDPAR2_3)	32	RW	0000_0000h
C834h	Device physical endpoint-n command parameter 1 register (DEPCMDPAR1_3)	32	RW	0000_0000h
C838h	Device physical endpoint-n command parameter 0 register (DEPCMDPAR0_3)	32	RW	0000_0000h
C83Ch	Device physical endpoint-n command register (DEPCMD_3)	32	RW	0000_0000h
C840h	Device physical endpoint-n command parameter 2 register (DEPCMDPAR2_4)	32	RW	0000_0000h
C844h	Device physical endpoint-n command parameter 1 register (DEPCMDPAR1_4)	32	RW	0000_0000h
C848h	Device physical endpoint-n command parameter 0 register (DEPCMDPAR0_4)	32	RW	0000_0000h
C84Ch	Device physical endpoint-n command register (DEPCMD_4)	32	RW	0000_0000h
C850h	Device physical endpoint-n command parameter 2 register (DEPCMDPAR2_5)	32	RW	0000_0000h
C854h	Device physical endpoint-n command parameter 1 register (DEPCMDPAR1_5)	32	RW	0000_0000h
C858h	Device physical endpoint-n command parameter 0 register (DEPCMDPAR0_5)	32	RW	0000_0000h
C85Ch	Device physical endpoint-n command register (DEPCMD_5)	32	RW	0000_0000h
C860h	Device physical endpoint-n command parameter 2 register (DEPCMDPAR2_6)	32	RW	0000_0000h
C864h	Device physical endpoint-n command parameter 1 register (DEPCMDPAR1_6)	32	RW	0000_0000h
C868h	Device physical endpoint-n command parameter 0 register (DEPCMDPAR0_6)	32	RW	0000_0000h
C86Ch	Device physical endpoint-n command register (DEPCMD_6)	32	RW	0000_0000h
C870h	Device physical endpoint-n command parameter 2 register (DEPCMDPAR2_7)	32	RW	0000_0000h
C874h	Device physical endpoint-n command parameter 1 register (DEPCMDPAR1_7)	32	RW	0000_0000h
C878h	Device physical endpoint-n command parameter 0 register (DEPCMDPAR0_7)	32	RW	0000_0000h
C87Ch	Device physical endpoint-n command register (DEPCMD_7)	32	RW	0000_0000h
CC00h	OTG configuration register (OCFG)	32	RW	0000_0000h
CC04h	OTG control register (OCTL)	32	RW	0000_0000h
CC08h	OTG events register (OEVT)	32	RW	0000_0000h
CC0Ch	OTG events enable register (OEVTEN)	32	RW	0000_0000h
CC10h	OTG status register (OSTS)	32	RO	0000_0019h
CC20h	ADP configuration register (ADPCFG)	32	RW	0000_0001h
CC24h	ADP control register (ADPCTL)	32	RW	0000_0001h
CC28h	ADP event register (ADPEVT)	32	RW	0000_0000h

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
CC2Ch	ADP event enable register (ADPEVTEN)	32	RW	0200_0001h

33.2.2 Capability registers length and HC interface version number (CAPLENGTH)

33.2.2.1 Offset

Register	Offset
CAPLENGTH	0h

33.2.2.2 Diagram

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

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								CAPLENGTH							
W																
Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

33.2.2.3 Fields

Field	Function
31-16 HCIVERSION	HC interface version number The value is set as 32'h100.
15-8 —	Reserved
7-0 CAPLENGTH	Capability registers length The value is 32'h20.

33.2.3 Host controller structural parameters 1 (HCSPARAMS1)

33.2.3.1 Offset

Register	Offset
HCSPARAMS1	4h

33.2.3.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	MAXPORTS								Reserved					MAXINTRS		
W																
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MAXINTRS								MAXSLOTS							
W																
Reset	0	0	0	0	0	0	0	1	0	1	1	1	1	1	1	1

33.2.3.3 Fields

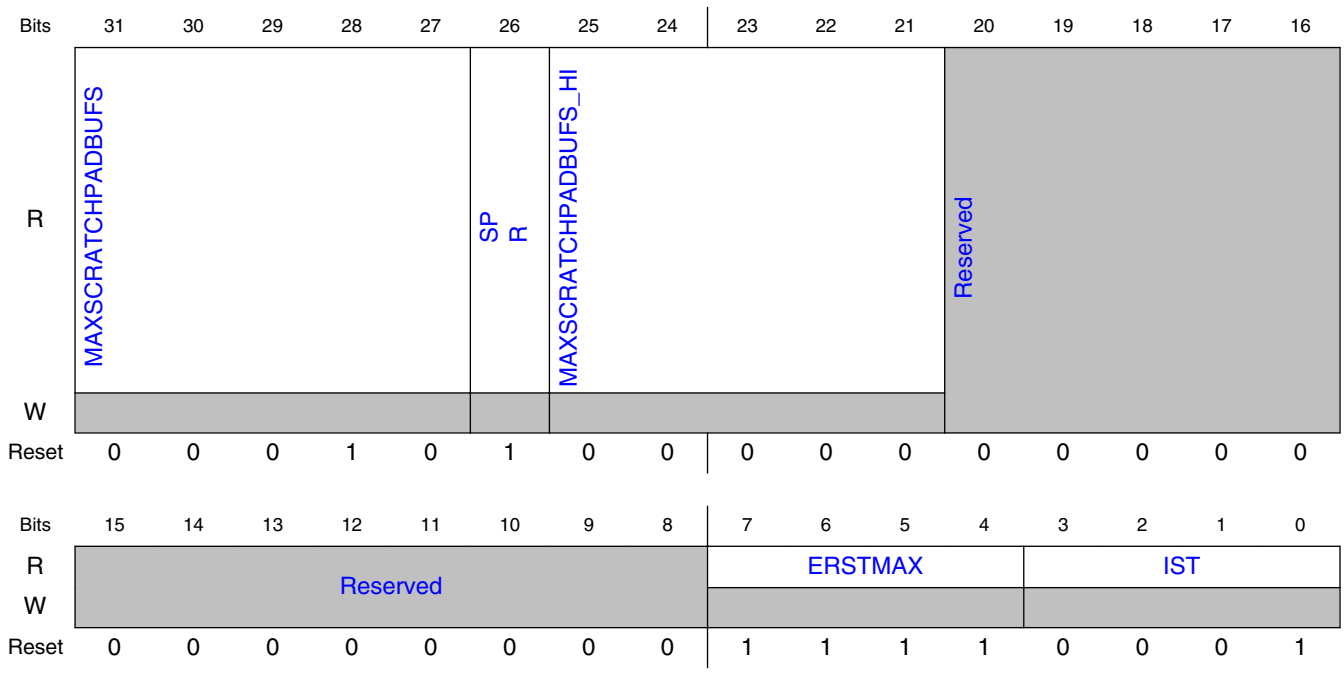
Field	Function
31-24 MAXPORTS	Maximum number of ports set as 2.
23-19 —	Reserved
18-8 MAXINTRS	Number of interrupters set as 1.
7-0 MAXSLOTS	Number of device slots set as 127.

33.2.4 Host controller structural parameters 2 (HCSPARAMS2)

33.2.4.1 Offset

Register	Offset
HCSPARAMS2	8h

33.2.4.2 Diagram



33.2.4.3 Fields

Field	Function
31-27 MAXSCRATCHPADBUFS	Max scratchpad buffers low set as 2
26 SPR	Scratchpad restore
25-21	Max scratchpad buffers high

Table continues on the next page...

Field	Function
MAXSCRATCHPADBUFS_HI	
20-8 —	Reserved
7-4 ERSTMAX	Event ring segment table max set as 15
3-0 IST	Isochronous scheduling threshold set as 1.

33.2.5 Host controller structural parameters 3 (HCSPARAMS3)

33.2.5.1 Offset

Register	Offset
HCSPARAMS3	Ch

33.2.5.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	U2_DEVICE_EXIT_LAT															
W																
Reset	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								U1_DEVICE_EXIT_LAT							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0

33.2.5.3 Fields

Field	Function
31-16	U2 device exit latency set as 32'h7FF

Table continues on the next page...

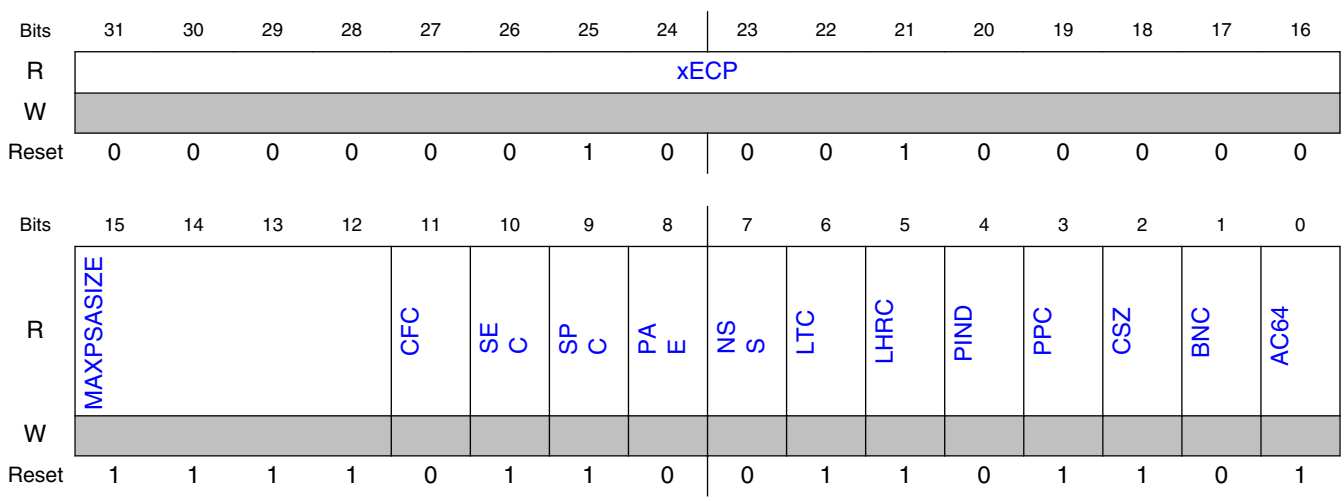
Field	Function
U2_DEVICE_EXIT_LAT	
15-8 —	Reserved
7-0 U1_DEVICE_EXIT_LAT	U1 device exit latency set as 32'hA

33.2.6 Host controller capability parameters 1 (HCCPARAMS1)

33.2.6.1 Offset

Register	Offset
HCCPARAMS1	10h

33.2.6.2 Diagram



33.2.6.3 Fields

Field	Function
31-16	xHCI extended capabilities pointer set as 544

Table continues on the next page...

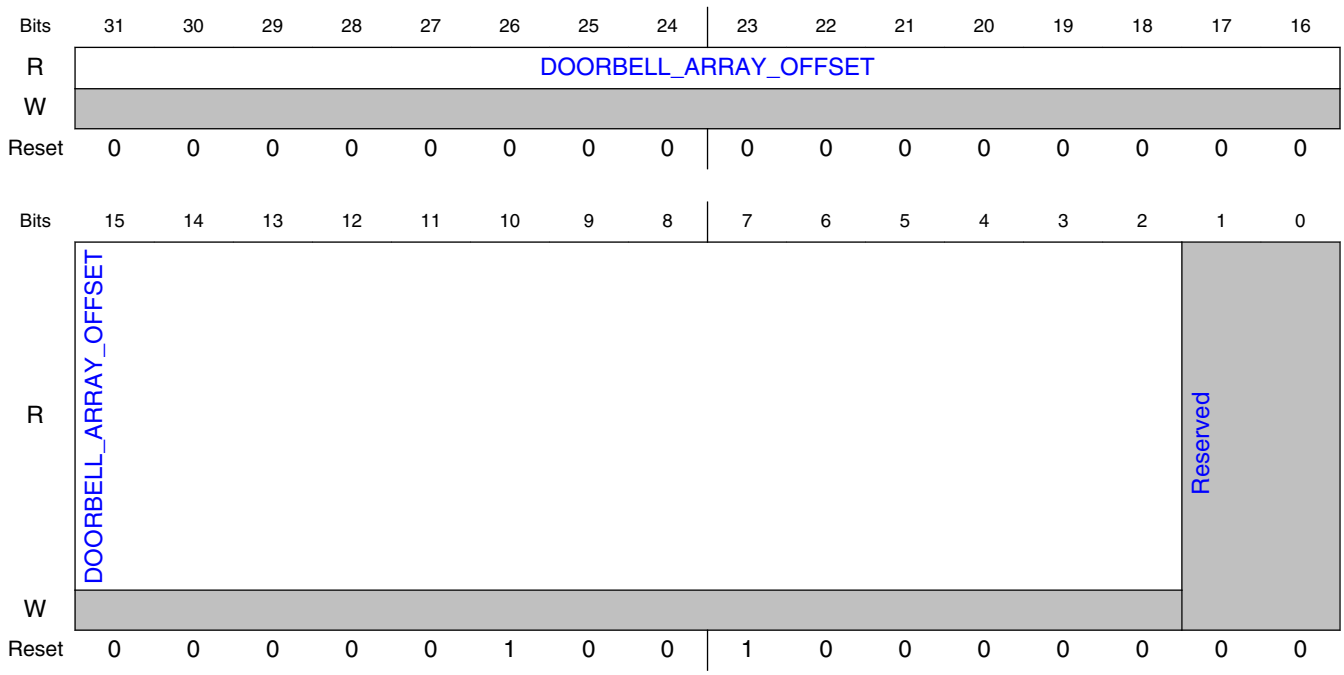
Field	Function
xECP	
15-12 MAXPSASIZE	Maximum primary stream array size set as 15
11 CFC	Contiguous frame ID capability
10 SEC	Stopped EDLTA capability
9 SPC	Short packet capability
8 PAE	Parse all event data
7 NSS	No secondary SID support
6 LTC	Latency tolerance messaging capability
5 LHRC	Light HC reset capability
4 PIND	Port indicators
3 PPC	Port power control
2 CSZ	Context size
1 BNC	BW negotiation capability
0 AC64	64-bit addressing capability

33.2.7 Doorbell offset (DBOFF)

33.2.7.1 Offset

Register	Offset
DBOFF	14h

33.2.7.2 Diagram



33.2.7.3 Fields

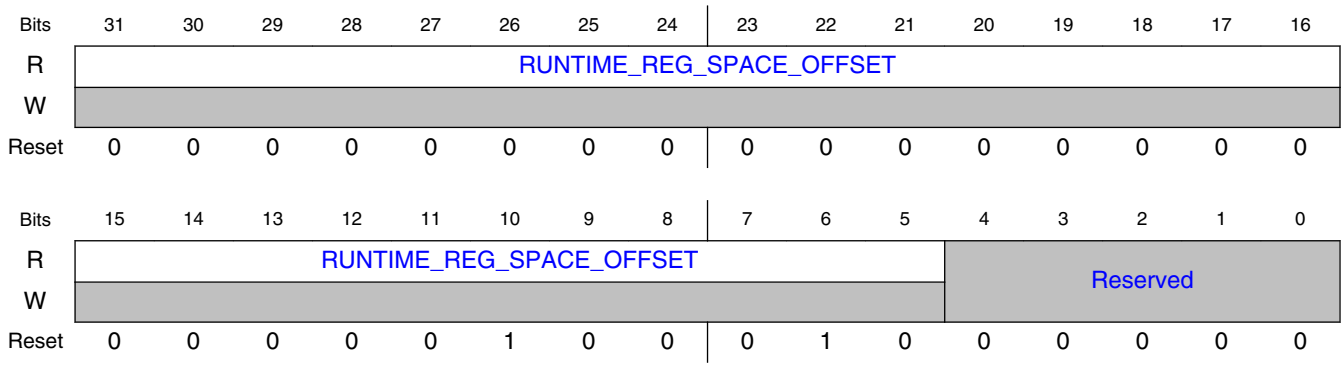
Field	Function
31-2 DOORBELL_ARRAY_OFFSET	Doorbell array offset set as 1152
1-0 —	Reserved

33.2.8 Runtime register space offset (RTSOFF)

33.2.8.1 Offset

Register	Offset
RTSOFF	18h

33.2.8.2 Diagram



33.2.8.3 Fields

Field	Function
31-5 RUNTIME_REG_SPACE_OFFSET	Runtime register space offset set as 1088
4-0 —	Reserved

33.2.9 Host controller capability parameters 2 (HCCPARAMS2)

33.2.9.1 Offset

Register	Offset
HCCPARAMS2	1Ch

33.2.9.2 Diagram

Bits	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

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved										CIC	LEC	CTC	FSC	CMC	U3C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1

33.2.9.3 Fields

Field	Function
31-6 —	Reserved
5 CIC	Configuration information capability
4 LEC	Large ESIT payload capability
3 CTC	Compliance transition capability
2 FSC	Force save context capability
1 CMC	Configure endpoint command max exit latency too large capability
0 U3C	U3 entry capability

33.2.10 Global SoC bus configuration register 0 (GSBUSCFG0)

33.2.10.1 Offset

Register	Offset
GSBUSCFG0	C100h

33.2.10.2 Function

This register cconfigures system bus DMA options for the AXI master bus. Options include burst length and cache type (bufferable/posted, cacheable/snoop, and so on). The application can program this register upon power-on, or a change in mode of operation after the DMA engine is halted.

xHCI register power-on value

While using a standard xHCI host driver, make sure to set the register's power-on value (current values are 32'h100080 and 32'h700) because the standard xHCI driver does not access this register.

Burst length

For a given DMA transfer, the burst length is set according to the largest enabled burst length. Note that the undefined length INCR burst, if enabled, has priority over all other burst lengths.

Cache type

For a given DMA transfer, the cache type is set according to the 4-bit cache type register bit corresponding to the transfer type:

- Data read
- Descriptor read
- Data write
- Descriptor write

The definition of the 4-bit cache type register bit corresponds to the cache type definition of the configured master bus type (AXI3) as defined below:

Table 33-4. Cache type bit assignments

MBUS_TYPE	Cache type output	[3]	[2]	[1]	[0]
AXI3	ar/awcache[3:0]	Write allocate	Read allocate	Cacheable	Bufferable

Cache type in AXI

AXI3: CSR cache type refers to AXI cache type (awcache[3:0] and arcache[3:0]); or in ACE (AXI coherency extensions) to AXI memory type.

Posted requests

The native interface DMA read request gmr_mcmd[2:0] indicates posted vs non-posted requests depending on the bufferable/posted bit of the cache type.

Table 33-5. Cache type bit definitions and usage

Signal name	Definition	Usage
Bufferable, Posted	The transaction response can be returned quickly, but the transaction itself can take an arbitrary number of clock cycles to reach the final destination.	• Improves bus utilization of DMA writes with a high response latency, such as off-chip memory or bridges. • Allows the core to start a new DMA write transaction while a DMA write is in progress. • For descriptor write transactions, "Bufferable/ Posted" must be 0 to avoid a DMA write race condition with a software interrupt or subsequent DMA read. When the core writes back a descriptor or event, it waits for the DMA response to indicate that it can safely re-fetch that descriptor or set the core's interrupt. If the descriptor DMA write is posted, the data integrity is not ensured for future transactions.
Cacheable, Modifiable, Snoop (negation of No Snoop)	The characteristics of the transaction at the destination may not match the original.	• For DMA writes, multiple write transactions may be merged. • For DMA reads, a location can be pre-fetched or fetched just once for multiple read transactions. • For AXI, this bit should be used in conjunction with the read allocate / write allocate to indicate upon a cache miss whether the transaction should be cached, or other allocate bits to indicate how to handle cache misses.

Connections from GSBUSCFG0 to master cache type

In device mode, the GSBUSCFG0 cache type bits are connected to the two (write and read) system bus master cache type ports through a pair of 4-bit wide 2x1 multiplexers; the MUX select signal is the descriptor access indication (as opposed to data).

In host mode, the GSBUSCFG0 cache type bits are connected as in device mode with these exceptions:

- For data accesses, the core sets cacheable to the inverse of the xHCI TRB's (transfer request block) no snoop flag; the cacheable bit in GSBUSCFG0 is disregarded.
- For descriptor accesses, the core sets cacheable to the inverse of the internal DMA command's no snoop flag. When the LSP performs a Scratchpad DMA write, it sets the internal DMA command's No Snoop flag thereby ensuring cacheable=0, as required by the xHCI specification.

The master bus type is AXI, the GSBUSCFG0 to cache type connections are reflected by following diagrams.

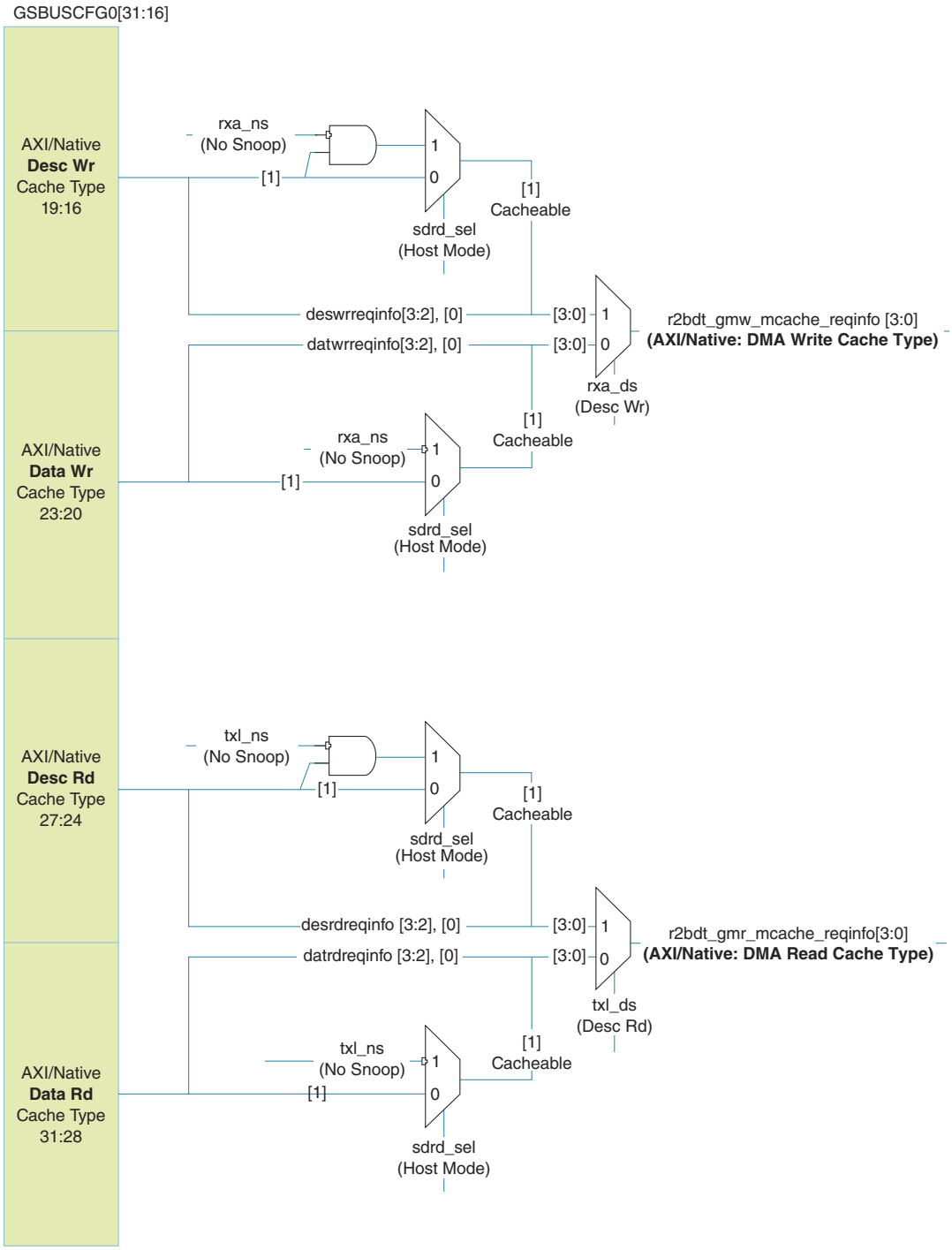
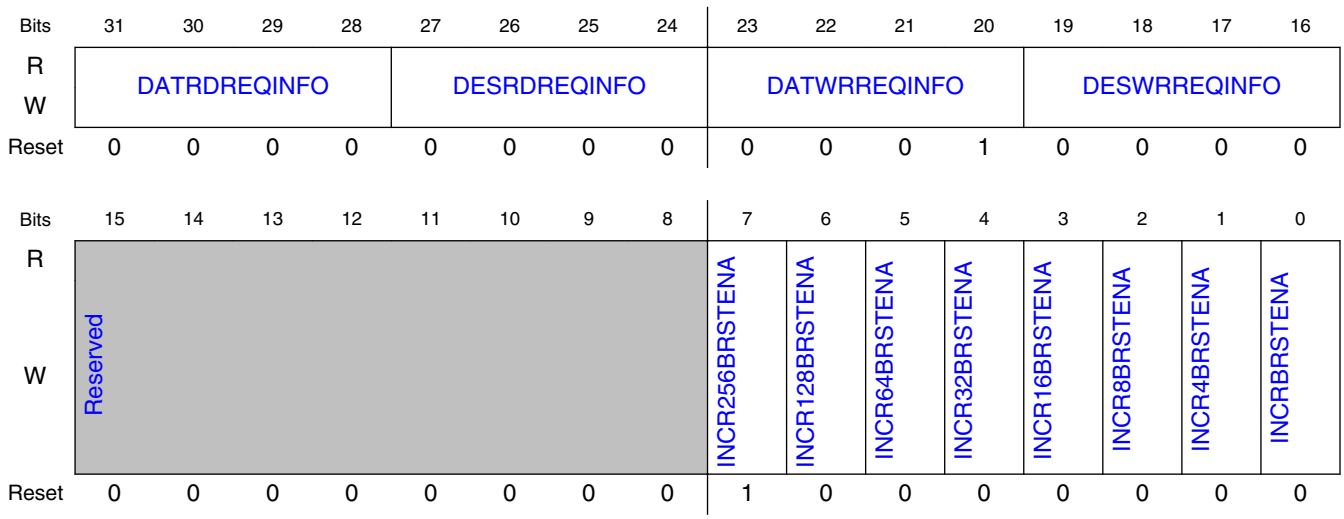


Figure 33-2. GSBUSCFG0 cache type connections for AXI

NOTE

For master interface DMA access, program the GSBUSCFG0 register to 0x2222000F for better performance.

33.2.10.3 Diagram



33.2.10.4 Fields

Field	Function
31-28 DATRDREQINFO	AXI-cache for data read Input to BUS-GM.
27-24 DESRDREQINFO	AXI-cache for descriptor read Input to BUS-GM.
23-20 DATWRREQINFO	AXI-cache for data write Input to BUS-GM.
19-16 DESWRREQINFO	AXI-cache for descriptor write Input to BUS-GM.
15-8 —	Reserved
7 INCR256BRSTENA	NCR256 burst type enable Input to BUS-GM. For the AXI configuration, if software set this bit to 1, the AXI master uses INCR to do the 256-beat burst. **NOTE:** The bit is not supported for now.
6 INCR128BRSTENA	NCR128 burst type enable Input to BUS-GM. For the AXI configuration, if software set this bit to 1, the AXI master uses INCR to do the 128-beat burst. **NOTE:** The bit is not supported for now.

Table continues on the next page...

Field	Function
5 INCR64BRSTENA	INCR64 burst type enable Input to BUS-GM. For the AXI configuration, if software set this bit to 1, the AXI master uses INCR to do the 64-beat burst. **NOTE:** The bit is not supported for now.
4 INCR32BRSTENA	INCR32 burst type enable Input to BUS-GM. For the AXI configuration, if software set this bit to 1, the AXI master uses INCR to do the 32-beat burst. **NOTE:** The bit is not supported for now.
3 INCR16BRSTENA	INCR16 burst type enable Input to BUS-GM. For the AXI configuration, if software sets this bit to 1, the AXI master uses INCR to do the 16-beat burst.
2 INCR8BRSTENA	INCR8 burst type enable Input to BUS-GM. For the AXI configuration, if software sets this bit to 1, the AXI master uses INCR to do the 8-beat burst.
1 INCR4BRSTENA	INCR4 burst type enable Input to BUS-GM. For the AXI configuration, when this bit is enabled, the controller is allowed to do bursts of beat length 1, 2, 3, and 4. It is highly recommended that this bit is enabled to prevent descriptor reads and writes from being broken up into separate transfers.
0 INCRBRSTENA	Undefined length INCR burst type enable Input to BUS-GM This bit determines the set of burst lengths the master interface uses. It works in conjunction with the GSBUSCFG0[7-1] enables (INCR256/128/64/32/16/8/4). 0b - INCRX burst mode ARLEN/AWLEN (for AXI configurations) use only the following burst lengths: • 1 (if GSBUSCFG0[1:7] = 0 • 4 (if GSBUSCFG0[INCR4BRSTENA] = 1) • 8 (if GSBUSCFG0[INCR8BRSTENA] = 1) • 16 (if GSBUSCFG0[INCR16BRSTENA] = 1) • 32 (if GSBUSCFG0[INCR32BRSTENA] = 1) • 64 (if GSBUSCFG0[INCR64BRSTENA] = 1) • 128 (if GSBUSCFG0[INCR128BRSTENA] = 1) • 256 (if GSBUSCFG0[INCR256BRSTENA] = 1) They do not use INCR. 1b - INCR (undefined length) burst mode AXI configurations- ARLEN/AWLEN uses any length less than or equal to the largest-enabled burst length of INCR4/8/16/32/64/128/256. For cache line-aligned applications, this bit is typically set to 0 to ensure that the master interface uses only power-of-2 burst lengths (as enabled through GSBUSCFG0[7-0]).

33.2.11 Global SoC bus configuration register 1 (GSBUSCFG1)

33.2.11.1 Offset

Register	Offset
GSBUSCFG1	C104h

33.2.11.2 Function

NOTE

For master interface DMA access, program the GSBUSCFG1 register to 0x00000F00 for better performance.

33.2.11.3 Diagram

Bits	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

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved			EN1KPAGE	PIPETRANSLIMIT				Reserved							
Reset	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0

33.2.11.4 Fields

Field	Function
31-13 —	Reserved
12 EN1KPAGE	1K page boundary enable By default (this bit is disabled), the AXI breaks transfers at the 4K page boundary. When this bit is enabled, the AXI master (DMA data) breaks transfers at the 1K page boundary.
11-8	AXI pipelined transfers burst request limit

Table continues on the next page...

Field	Function
PIPETRANSLIMIT	The bit controls the number of outstanding pipelined transfers requests the AXI master pushes to the AXI slave. Once the AXI master reaches this limit, it does not make more requests on the AXI ARADDR and AWADDR buses until the associated data phases complete. 0000b - 1 request 0001b - 2 requests 0010b - 3 requests 0011b - 4 requests ... 1111b - 16 requests
7-0 —	Reserved

33.2.12 Global Tx threshold control register (GTXTHRCFG)

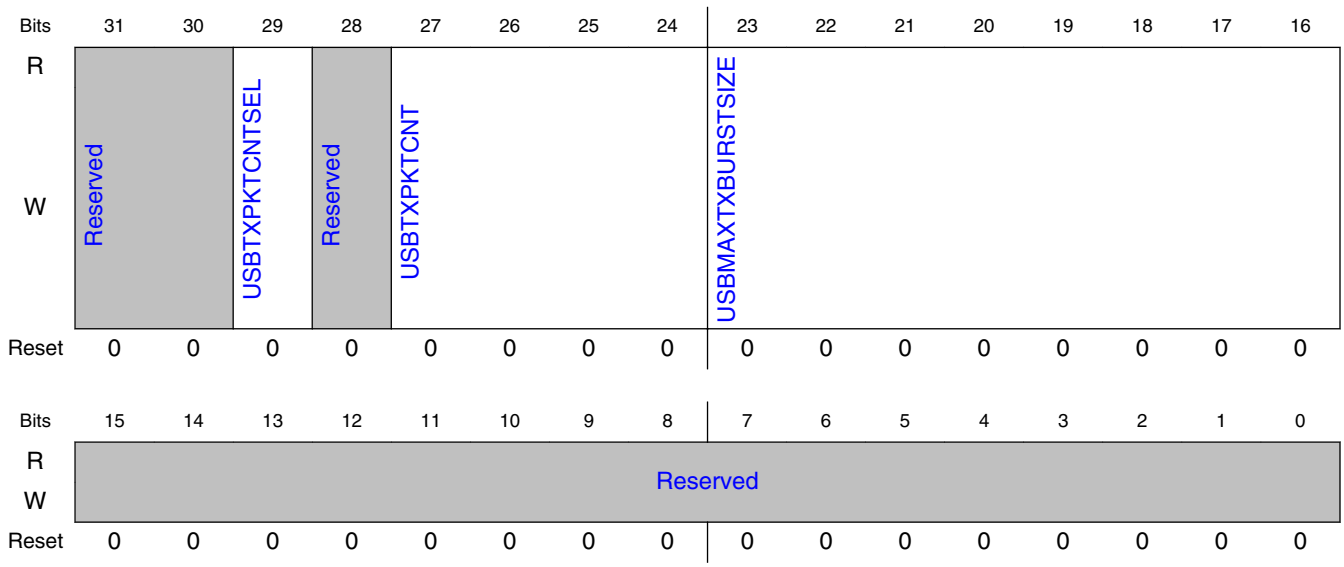
33.2.12.1 Offset

Register	Offset
GTXTHRCFG	C108h

33.2.12.2 Function

All bits in the GTXTHRCFG register are valid only in host mode. This register is not applicable for debug target and USB 2.0-only mode.

33.2.12.3 Diagram



33.2.12.4 Fields

Field	Function
31-30 —	Reserved
29 USBTXPKTCNTSEL	USB transmit packet count enable This bit enables/disables the USB transmission multi-packet thresholding. 0b - USB transmission multi-packet thresholding is disabled, the core can only start transmission on the USB after the entire packet has been fetched into the corresponding TXFIFO. 1b - USB transmission multi-packet thresholding is enabled. The core can only start transmission on the USB after USB transmit packet count amount of packets for the USB transaction (burst) are already in the corresponding TXFIFO. This mode is only valid in the host mode. It is only used for SuperSpeed.
28 —	Reserved
27-24 USBTXPKTCNT	USB transmit packet count This bit specifies the number of packets that must be in the TXFIFO before the core can start transmission for the corresponding USB transaction (burst). This bit is only valid when the USBTXPKTCNTSEL bit is set to 1. The valid values are from 1 to 15. **NOTE:** This bit must be less than or equal to the USBMAXTXBURSTSIZE bit.
23-16 USBMAXTXBURSTSIZE	USB maximum Tx burst size When USBTXPKTCNTSEL is set to 1, this bit specifies the maximum bulk OUT burst, the core can do. When the system bus is slower than the USB, TXFIFO can underrun during a long burst. User can program a smaller value to this bit to limit the TX burst size that the core can do. It only applies to SS Bulk, isochronous, and interrupt OUT endpoints in the host mode. Valid values are from 1 to 16.

Table continues on the next page...

Field	Function
15-0 —	Reserved

33.2.13 Global Rx threshold control register (GRXTHRCFG)

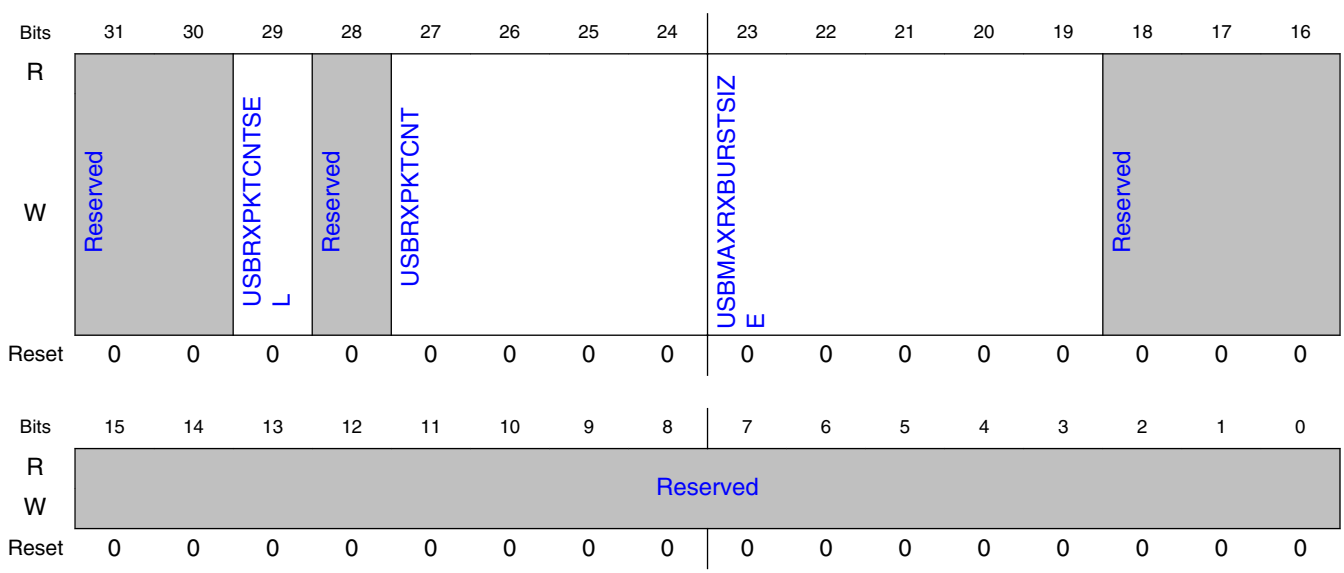
33.2.13.1 Offset

Register	Offset
GRXTHRCFG	C10Ch

33.2.13.2 Function

This register is not applicable for debug target and USB 2.0-only mode.

33.2.13.3 Diagram



33.2.13.4 Fields

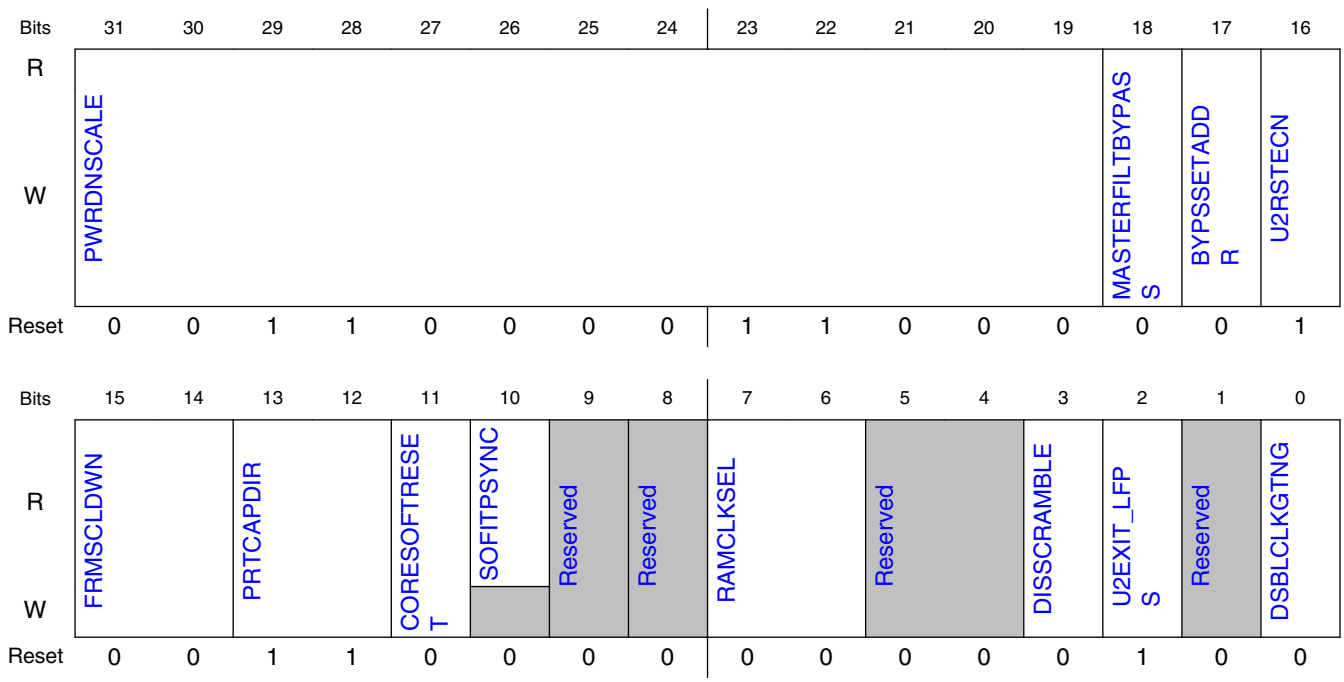
Field	Function
31-30 —	Reserved
29 USBRXPKTCNTSEL	USB receive packet count enable This bit enables/disables the USB reception multi-packet thresholding. In device mode, Setting this bit to 1 also enables the functionality of reporting NUMP in the ACK TP based on the RXFIFO space instead of reporting a fixed NUMP derived from DCFG[NUMP]. While using external buffer control (EBC) feature, disable this mode by setting USBRXPKTCNTSEL to 0. 0b - The core can only start reception on the USB when the RXFIFO has space for at least one packet. 1b - The core can only start reception on the USB when the RXFIFO has space for at least USBRXPKTCNT amount of packets. This mode is valid in both host and device mode. It is only used for SuperSpeed.
28 —	Reserved
27-24 USBRXPKTCNT	USB receive packet count In host mode, this bit specifies the space (in terms of the number of packets) that must be available in the RXFIFO before the core can start the corresponding USB RX transaction (burst). In device mode, this bit specifies the space (in terms of the number of packets) that must be available in the RXFIFO before the core can send ERDY for a flow-controlled endpoint. This bit is valid only when the USBRXPKTCNTSE bit is set to 1. The valid values for this bit are from 1 to 15. **NOTE:** This bit must be less than or equal to the USB maximum receive burst size (USBMAXRXBURSTSIZE) bit.
23-19 USBMAXRXBURSTSIZE	USB maximum receive burst size In host mode, this bit specifies the maximum bulk IN burst, the USB 3.0 core can perform. When the system bus is slower than the USB, RXFIFO can overrun during a long burst. User can program a smaller value to this bit to limit the RX burst size that the core can perform. It only applies to SS bulk, isochronous, and interrupt IN endpoints in the host mode. In device mode, this bit specifies the NUMP value that is sent in ERDY for an OUT endpoint. This bit is valid only when USBRXPKTCNTSEL is set to 1. The valid values for this bit are from 1 to 16.
18-0 —	Reserved

33.2.14 Global core control register (GCTL)

33.2.14.1 Offset

Register	Offset
GCTL	C110h

33.2.14.2 Diagram



33.2.14.3 Fields

Field	Function
31-19 PWRDNSCALE	Power down scale The USB3 suspend_clk input replaces pipe3_rx_pclk as a clock source to a small part of the USB3 core that operates when the SS PHY is in its lowest power (P3) state, and therefore does not provide a clock. The power down scale bit specifies how many suspend_clk periods fit into a 16 KHz clock period. When performing the division, round up the remainder. For example, when using an 8-/16-/32-bit PHY and 25 MHz suspend clock, Power down scale = 25000 KHz/16 KHz = 13'd1563 (rounder up) **NOTE:** • Minimum suspend clock frequency is 32 KHz • Maximum suspend clock frequency is 125 MHz The LTSSM uses suspend clock for 12 ms and 100 ms timers during suspend mode. According to the USB 3.0 specification, the accuracy on these timers is 0% to +50%.

Table continues on the next page...

Field	Function
	• 12 ms + 0~+50% accuracy = 18 ms (Range is 12 ms - 18 ms) • 100 ms + 0~+50% accuracy = 150 ms (Range is 100 ms - 150 ms) The suspend clock accuracy requirement is: • (12,000/62.5) * (GCTL[PWRDNSCALE]) * actual suspend_clk_period should be between 12,000 and 18,000 • (100,0000/62.5) * (GCTL[PWRDNSCALE]) * actual suspend_clk_period should be between 100,000 and 150,000 For example, if the suspend_clk frequency varies from 7.5 MHz to 10.5 MHz, then the value needs to programmed is: Power down scale = 10500/16 = 657 (rounded up; and fastest frequency used)
18 MASTERFILTBYPASS	Master filter bypass 0b - Enable all the filter modules 1b - Bypass all the filter modules
17 BYPSSETADDR	Bypass SetAddress in device mode When this bit is set, the device core uses the value in DCFG[DEVADDR] bits directly for comparing the device address in the tokens. **NOTE:** This bit must be set to 0.
16 U2RSTECN	If the SuperSpeed connection fails during POLL or LMP exchange, the device connects at non-SS mode. If this bit is set, then device attempts three more times to connect at SS, even if it previously failed to operate in SS mode. **NOTE:** This bit is applicable only in device mode.
15-14 FRMSCLDWN	This bit scales down device view of a SoF/USOF/ITP duration. For SS/HS mode: 11 Implements interval to be 15.625 µs 10 Implements interval to be 31.25 µs 01 Implements interval to be 62.5 µs 00 Implements interval to be 125 µs For FS mode, the scale-down value is multiplied by 8. 00 1024 bytes 01 512 bytes 10 256 bytes 11 128 bytes
13-12 PRTCAPDIR	Port capability direction The sequence for switching modes in DRD configuration is as follows: Switching from device to host: 1. Reset the controller using GCTL[CORESOFTRESET]. 2. Set GCTL[PRTCAPDIR] to 2'b01 (Host mode). 3. Reset the host using USBCMD[HCRESET]. 4. Follow the steps in "Initializing Host Registers". Switching from host to device: 1. Reset the controller using GCTL[CORESOFTRESET]. 2. Set GCTL[PRTCAPDIR] to 2'b10 (Device mode). 3. Reset the device by setting DCTL[CSFTRST]. 4. Follow the steps in "Register Initialization".

Table continues on the next page...

Field	Function
	01b - For host configurations 10b - For device configurations 11b - For OTG configurations For OTG, if PRTCAPDIR is 2'b11, it acts as an OTG 2.0 device with A-device or B-device determined by the IDDIG input, and host or peripheral role based on HNP. If PRTCAPDIR is 2'b01, it acts as a DRD in host mode. If PRTCAPDIR is 2'b10, it acts as a DRD in device mode. The OTG device can be programmed to enable/disable SRP and HNP by using the bits present in the OCFG register.
11 CORESOFTRESET	Core Soft Reset 0 No soft reset 1 Soft reset to core Clears the interrupts and all the CSRs except the following registers: GCTL GUCTL GSTS GGPIO GUID GUSB2PHYCFGn registers GUSB3PIPECTLn registers DCFG DCTL DEVTEN DSTS While resetting PHYs (using GUBS3PHYCFG or GUSB3PIPECTL registers), the core must be in reset state until PHY clocks are stable. This controls the bus, ram, and mac domain resets. **NOTE:** This bit is for debug purpose only. Use USBCMD.HCRESET in xHCI Mode and DCTL.SoftReset in device mode for soft reset.
10 SOFITPSYNC	The bit is set to 0. The core keeps the UTMI PHY on the first port in a non-suspended state whenever there is a SuperSpeed port that is not in Rx.Detect, SS.Disable and U3.
9 —	Reserved
8 —	Reserved
7-6 RAMCLKSEL	RAM clock select On USB-reset, hardware clears these bits to 2'b00. **NOTE:** In host mode, this bit must be set to 2'b00, that is, the ram_clk must be assigned to bus_clk only. The reason is if the SS port0 goes to P3, the pipe_clk is shutdown, and the USB 2.0 ports cannot operate. 00b - Bus clock 01b - Pipe clock 10b - Pipe/2 clock 11b - Reserved
5-4 —	Reserved
3	Disable scrambling

Table continues on the next page...

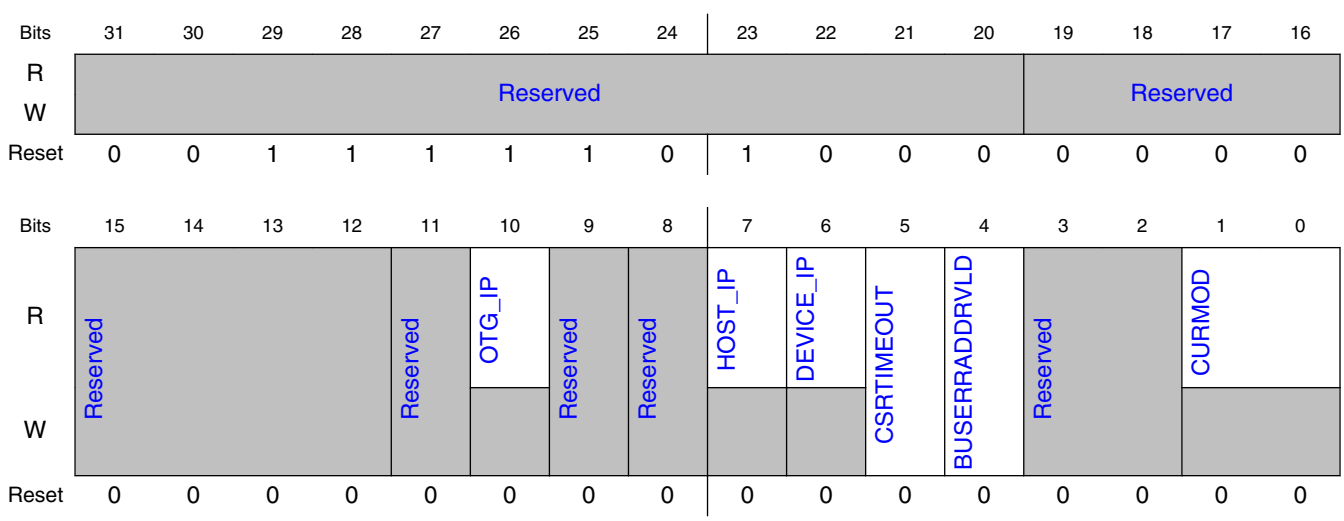
Field	Function
DISSCRAMBLE	Transmit request to link partner on next transition to recovery or polling.
2 U2EXIT_LFPS	This bit is added to improve interoperability with a third party host controller. This host controller in U2 state while performing receiver detection generates an LFPS glitch of about 4 µs duration. This causes the device to exit from U2 state because the LFPS filter value is 248 ns. With the new functionality enabled, the device can stay in U2 while ignoring this glitch from the host controller. 0b - The link treats 248 ns LFPS as a valid U2 exit. 1b - The link waits for 8 µs of LFPS before it detects a valid U2 exit.
1 —	Reserved
0 DSBLCLKGTNG	Disable clock gating When this bit is set to 1 and the core is in low power mode, internal clock gating is disabled. Set this bit to 1 after power-on-reset.

33.2.15 Global status register (GSTS)

33.2.15.1 Offset

Register	Offset
GSTS	C118h

33.2.15.2 Diagram



33.2.15.3 Fields

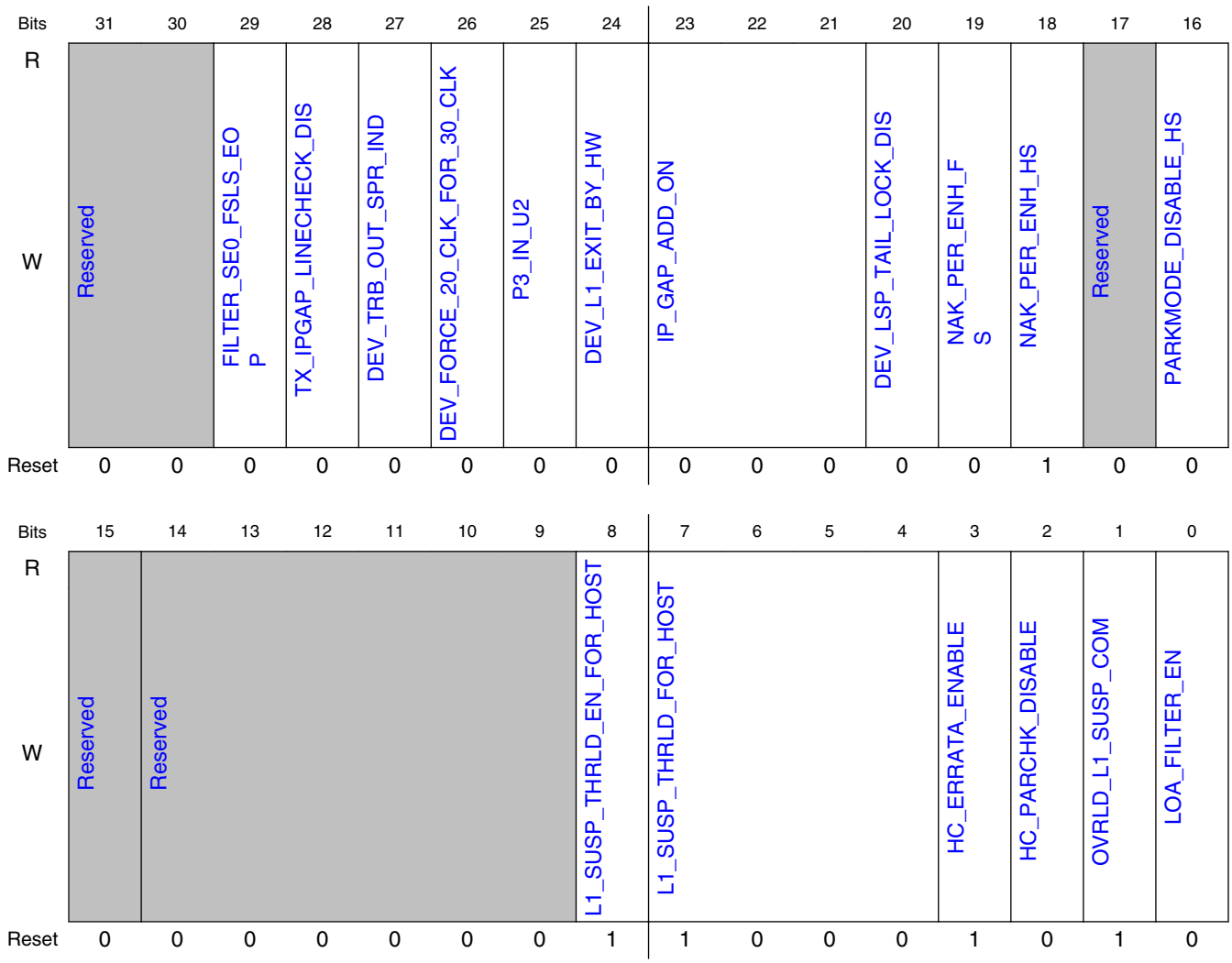
Field	Function
31-20 —	Reserved
19-12 —	Reserved
11 —	Reserved
10 OTG_IP	OTG interrupt pending This bit indicates that there is a pending interrupt pertaining to OTG in OEVT register.
9 —	Reserved
8 —	Reserved
7 HOST_IP	Host interrupt pending This bit indicates that there is a pending interrupt pertaining to xHC in the host event queue.
6 DEVICE_IP	Device interrupt pending This bit indicates that there is a pending interrupt pertaining to peripheral (device) operation in the device event queue.
5 CSRTIMEOUT	CSR timeout When this bit is set to 1, it indicates that software performed a write or read to a core register that could not be completed within 17'h1FFFF bus clock cycles.
4 BUSERRADDRVLD	Bus error address valid Indicates that the GBUSERRADDR register is valid and reports the first bus address that encounters a bus error.
3-2 —	Reserved
1-0 CURMOD	Current mode of operation 00b - Device mode 01b - Host mode

33.2.16 Global user control register 1 (GUCTL1)

33.2.16.1 Offset

Register	Offset
GUCTL1	C11Ch

33.2.16.2 Diagram



33.2.16.3 Fields

Field	Function
31-30 —	Reserved
29 FILTER_SE0_FSLS_EOP	This bit is applicable for FS/LS operation. If this feature is enabled, then SE0 on the linestate is validated for 2 consecutive utmi clock edges for EOP detection. This feature is applicable only in FS in device mode and FS/LS mode of operation in host mode.

Table continues on the next page...

Field	Function
	• Device mode: FS - If GUCTL1[FILTER_SE0_FSLS_EOP] is set, then for device LPM hanshake, the core ignores single SE0 glitch on the linestate during transmit. Only 2 or more SE0 is considered as a valid EOP on FS. • Host mode: FS/LS - If GUCTL1[FILTER_SE0_FSLS_EOP] is set, then the core ignores single SE0 glitch on the linestate during transmit. Only 2 or more SE0 is considered as a valid EOP on FS/LS port. Enable this feature if the LineState has SE0 glitches during transmission. This bit is quasi-static, i.e., should not be changed during device operation. 0b - No change in linestate check for SE0 detection in FS/LS (default) 1b - Feature enabled, FS/LS SE0 is filtered for 2 clocks for detecting EOP
28 TX_IPGAP_LINECHECK_DIS	This bit is applicable for HS operation of u2mac. If this feature is enabled, then the 2.0 mac operating in HS ignores the UTMI Linestate during the transmit of a token (during token-to-token and token-to-data IPGAP). When enabled, the controller implements a fixed 40-bit TxEndDelay after the packet is given on UTMI and ignores the linestate during this time. This fetaure is applicable only in HS mode of operation. Device mode: If GUCTL1[TX_IPGAP_LINECHECK_DIS] is set, then for device LPM handshake, the core ignores the linestate after TX and wait for a fixed clocks (40 bit times equivalent) after transmiting ACK on utmi. Host mode: If GUCTL1[TX_IPGAP_LINECHECK_DIS] is set, then the ipgap between (tkn to tkn/data) is added by 40 bit times of TXENDDELAY, and linestate is ignored during this 40 bit times delay. Enable this bit if the linestate does not reflect the expected line state (J) during transmission. This bit is quasi-static, i.e., should not be changed during device operation. 0b - No change in linestate check (default) 1b - Feature enabled, 2.0 MAC disables linestate check during HS transmit
27 DEV_TRB_OUT_SPR_IND	This bit is applicable for device mode only (and ignored in host mode). If the device application (SW/HW) wants to know if a short packet was received for an OUT in the TRB status itself, then this feature can be enabled, so that a bit is set in the TRB writeback in the buf_size DWORD. Bit[26] - SPR of the {trbstatus, RSVD, SPR, PCM1, bufsize} DWORD is set during an OUT transfer TRB write back if this is the last TRB used for that transfer descriptor. This bit is quasi-static, i.e., should not be changed during device operation. 0b - No change in TRB status DWORD (default) 1b - Feature enabled, OUT TRB status indicates short packet
26 DEV_FORCE_20_CLK_FOR_30_CLK	This bit is applicable (and to be set) for device mode (DCFG[SPEED] != SS) only. In the 3.0 device core, if the core is programmed to operate in 2.0 only (i.e., device speed is programmed to 2.0 speeds in DCFG[SPEED]), then setting this bit makes the internal 2.0 (utmi) clock to be routed as the 3.0 (pipe) clock. Enabling this feature allows the pipe3 clock to be not-running when forcibily operating in 2.0 device mode. **NOTE:** When using this feature, all pipe3 inputs must be in inactive mode, esp. pipe3 clocks not running and pipe3_phystatus_async must be tied to 0. This bit should not be set if the core is programmed to operate in SuperSpeed mode (even when it falls back to 2.0). This bit is quasi-static, i.e., should not be changed during operation. 0b - Uses 3.0 clock when operating in 2.0 mode (default) 1b - Feature enabled
25 P3_IN_U2	Setting this bit enables P3 power state when the SuperSpeed link is in U2. Another power saving option. Check with the PHY vendor before enabling this option. When setting this bit to 1 to enable P3 in P2, GUSB3PIPECTL[27] should be set to 0 to make sure that the U2 exit is attempted in P0. This bit should be set only when GCTL[SOFITPSYNC]=1 or GFLADJ[GFLADJ_REFCLK_LPM_SEL]=1. 0b - When SuperSpeed link is in U2 , PowerState P2 is attempted on the PIPE interface (default) 1b - When SuperSpeed link is in U2, PowerState P3 is attempted if GUSB3PIPECTL[17] is set
24	This bit is applicable for device mode (2.0) only. This bit enables device controller sending remote wakeup for L1 if the device becomes ready for sending/accepting data when in L1 state. If the host expects the device to send remote wakeup signalling to resume after going into L1 in flow controlled

Table continues on the next page...

Field	Function
DEV_L1_EXIT_BY_HW	state, then this bit can be set to send the remote wake signal automatically when the device controller becomes ready. This hardware remote wake feature is applicable only to bulk and interrupt transfers, and not for Isoch/Control. When control transfers are in progress, the LPM is rejected (NYET response). Only after control transfers are completed (either with ACK/STALL), LPM is accepted. For Isoch transfers, the host needs to do the wake-up and start the transfer. The device controller does not do remote-wakeup when Isoch endpoints get ready. The device software needs to keep the GUSB2PHYCFG[ENBLSLPM] reset in order to keep the PHY clock to be running for keeping track of SOF intervals. This bit is quasi-static, i.e., should not be changed during device operation. 0b - Disables device L1 hardware exit logic (default) 1b - Feature enabled
23-21 IP_GAP_ADD_ON	This bit is used to add on to the default inter packet gap setting in the USB 2.0 MAC.
20 DEV_LSP_TAIL_LOCK_DIS	This is a bug fix for STAR 9000716195 that affects the CSP mode for OUT endpoints in device mode. The issue is that tail TRB index is not synchronized with the cache scratchpad bytecount update. If the fast-forward request comes in-between the bytecount update ona newly fetched TRB and the tail-index write update in TPF, the RDP works on an incorrect tail index and misses the byte count decrement for the newly fetched TRB in the fast-forwarding process. This fix needs to be present all the times. 0b - Enables device lsp lock logic for tail TRB update (default) 1b - Fix disabled
19 NAK_PER_ENH_FS	If a periodic endpoint is present , and if a bulk endpoint which is also active is being NAKed by the device, then this could result in a decrease in performance of other full-speed bulk endpoint which is ACked by the device. Setting this bit to 1 enables the host controller to schedule more transactions to the async endpoints (bulk/ control) and hence improves the performance of the bulk endpoint. This control bit should be enabled only if the existing performance with the default setting is not sufficient for the full-speed application. Setting this bit only controls, and is only required for full-speed transfers. 0b - Enhancement not applied 1b - Enables performance enhancement for FS async endpoints in the presence of NAKs
18 NAK_PER_ENH_HS	If a periodic endpoint is present , and if a bulk endpoint which is also active is being NAKed by the device, then this could result in a decrease in performance of other high-speed bulk endpoint which is ACKed by the device. Setting this bit to 1, enables the host controller to schedule more transactions to the async endpoints (bulk/ control) and hence improves the performance of the bulk endpoint. This control bit should be enabled only if the existing performance with the default setting is not sufficient for the high-speed application. Setting this bit only controls, and is only required for high-speed transfers. 0b - Enhancement not applied 1b - Enables performance enhancement for HS async endpoints in the presence of NAKs
17 —	Reserved
16 PARKMODE_DISABLE_HS	This bit is used only in host mode. When this bit is set to 1, all HS bus instances park mode are disabled. To improve performance in park mode, the xHCI scheduler queues in three requests of 4 packets each for high-speed asynchronous endpoints in a micro-frame. But if a device is slow and if it NAKs more than 3 times, then it is rescheduled only in the next micro-frame. This could decrease the performance of a slow device even further. In a few high-speed devices (such as Sandisk Cruzer Blade 4GB VID:1921, PID:21863 and Flex Drive VID:3744, PID:8552) when an IN request is sent within 900 ns of the ACK of the previous packet, these devices send a NAK. When connected to these devices, if required, the software can disable the park

Table continues on the next page...

Field	Function
	mode if performance drop is found in your system. When park mode is disabled, pipelining of multiple packet is disabled and instead one packet at a time is requested by the scheduler. This allows up to 12 NAKs in a micro-frame and improves performance of these slow devices.
15 —	Reserved
14-9 —	Reserved
8 L1_SUSP_THRLD_EN_FOR_HOST	This bit is used only in host mode. The host controller asserts the utmi_l1_suspend_n and utmi_sleep_n output signals to the PHY in the L1 state. 0b - Disable 1b - Enable
7-4 L1_SUSP_THRLD_FOR_HOST	This bit is effective only when the L1_SUSP_THRLD_EN_FOR_HOST bit is set to 1. 1111 State is normal working 0110 State is L2 Suspend 0101 State is L1 Suspend 1011 State is L1 Sleep
3 HC_ERRATA_ENABLE	Host ELD enable When this bit is set to 1, it enables the exit latency delta (ELD) support defined in the xHCI 1.0 errata. This bit is used only in the host mode.
2 HC_PARCHK_DISABLE	Host parameter check disable 0b - The xHC checks that the input slot/EP context bits comply to the xHCI Specification. Upon detection of a parameter error during command execution, the xHC generates an event TRB with completion code indicating 'PARAMETER ERROR'. (default) 1b - The xHC does not perform parameter checks and does not generate 'PARAMETER ERROR' completion code.
1 OVRLD_L1_SUSP_COM	If this bit is set, the utmi_l1_suspend_com_n is overloaded with the utmi_sleep_n signal. This bit is usually set if the PHY stops the port clock during L1 sleep condition. **NOTE:** The recommended connection for the SUSPENDM/SLEEPM signals to the PHY with respect to this bit is as follows. For non-0 ports, connect • utmi_sleep_n[n] to SLEEPM[n] • (utmi_suspend_n[n] & utmi_l1_suspend_n[n]) to SUSPENDM[n] • USB2 PHYCLK[n] to utmi_clk[n] GUCTL1[OVRLD_L1_SUSP_COM] impacts only port 0. For port 0: • For PHY, GUSB2PHYCFGn[U2_FREECLK_EXISTS]=1: With this connection, the PHY keeps PLL active so that FREECLK is always available irrespective of suspend/sleep. • Connect USB2 PHY COMMONONN to 0. • Connect utmi_sleep_n[0] to SLEEPM[0]. • Connect (utmi_suspend_n[0] & utmi_l1_suspend_n[0]) to SUSPENDM[0]. • Connect USB2 PHY FREECLK to utmi_clk[0].

Table continues on the next page...

Field	Function
	• Leave utmi_suspend_com_n, utmi_l1_suspend_com_n unconnected. • GUCTL1[OVRLD_L1_SUSP_COM] can be set to any value. • For third-party PHY, GUSB2PHYCFGn[U2_FREECLK_EXISTS]=0: With this connection the PHY can shut off all the clocks when the required conditions are met (like, GUSB2PHYCFGn[8,6], GUCTL1[1], GFLADJ[23], GCTL[10], Suspend condition, HW LPM enable etc). • Connect ~utmi_suspend_com_n to SUSPENDM[0] (or equivalent). • Connect ~utmi_l1_suspend_com_n to SLEEPM[0] (or equivalent). • Connect PHYCLK0 (first port clock) to utmi_clk[0]. • Leave utmi_suspend_n[0], utmi_l1_suspend_n[0], utmi_sleep_n[0] unconnected. • Set GUCTL1[OVRLD_L1_SUSP_COM] to 1'b1.
0 LOA_FILTER_EN	If this bit is set, the USB 2.0 port babble is checked at least three consecutive times before the port is disabled. This prevents false triggering of the babble condition when using low quality cables. **NOTE:** This bit is valid only in host mode.

33.2.17 Global user ID register (GUID)

33.2.17.1 Offset

Register	Offset
GUID	C128h

33.2.17.2 Function

This register contains the user ID.

33.2.17.3 Diagram

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

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

33.2.17.4 Fields

Field	Function
31-0 USERID	User ID

33.2.18 Global user control register (GUCTL)

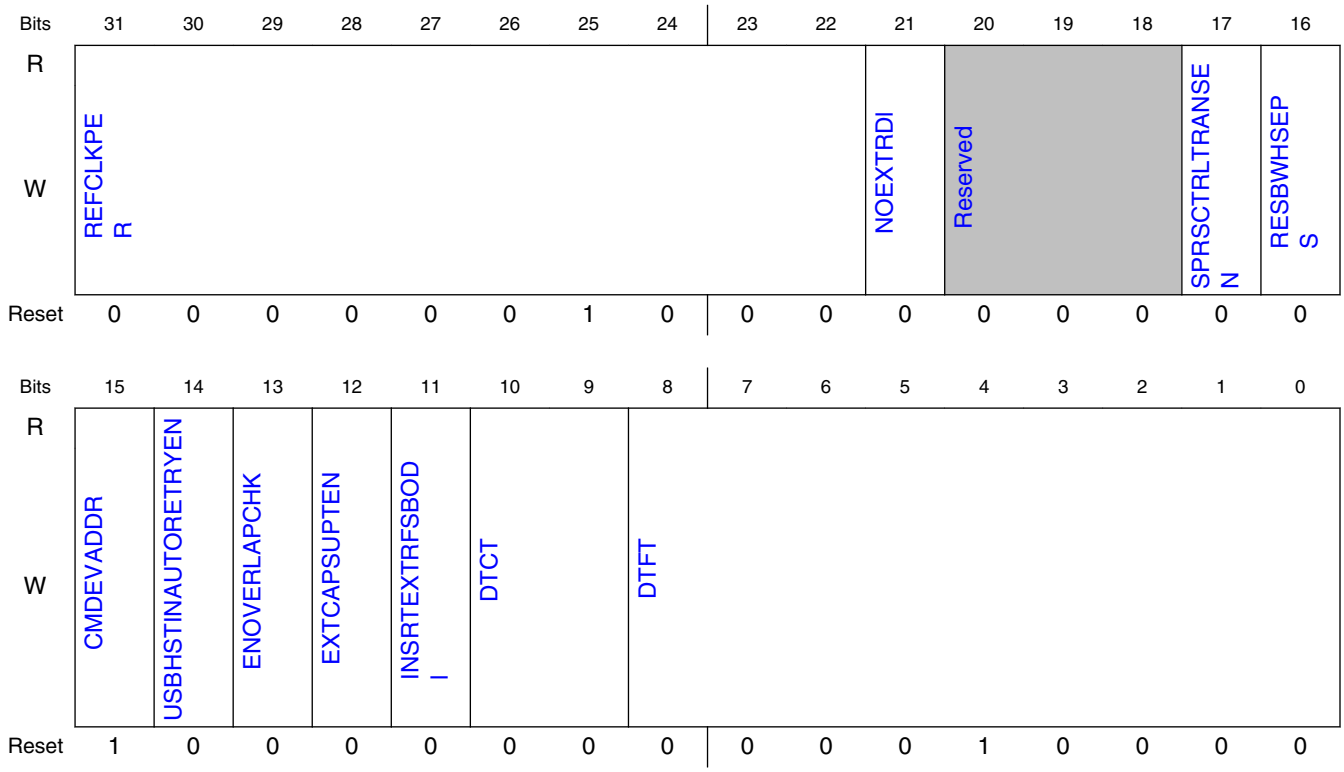
33.2.18.1 Offset

Register	Offset
GUCTL	C12Ch

33.2.18.2 Function

This register provides a few options for the software to control the core behavior in the host mode. Most of the options are used to improve host inter-operability with different devices.

33.2.18.3 Diagram



33.2.18.4 Fields

Field	Function
31-22 REFCLKPER	This bit indicates in terms of nano seconds the period of ref_clk. The default value of this register is set to 10'h8 (8 ns/125 MHz). This bit needs to be updated during power-on-initialization, if GCTL[SOFITPSYNC] or GFLADJ[GFLADJ_REFCLK_LPM_SEL] is set to 1. The programmable maximum value is 62 ns, and the minimum value is 8 ns. Use the reference clock with a period that is an integer multiple, so that ITP can meet the jitter margin of 32 ns. The allowable ref_clk frequencies whose period is not integer multiples are 16/17/19.2/24/39.7 MHz. This bit should not be set to 0 at any time. If this feature is not planned for future use, then set this bit to the default value (10'h8).
21 NOEXTRDI	No extra delay between SOF and the first packet Some HS devices misbehave when the host sends a packet immediately after a SOF. However, adding an extra delay between a SOF and the first packet can reduce the USB data rate and performance. This bit is used to control whether the host should wait for 2 µs before it sends the first packet after a SOF, or not. User can set this bit to 1 to improve the performance if those problematic devices are not a concern in the user's host environment. 0b - Host waits for 2 µs after a SOF before it sends the first USB packet 1b - Host doesn't wait after a SOF before it sends the first USB packet

Table continues on the next page...

Field	Function
20-18 —	Reserved
17 SPRSCTRLTRANSEN	Sparse control transaction enable Some devices are slow in responding to control transfers. Scheduling multiple transactions in one microframe/frame can cause these devices to misbehave. If this bit is set to 1'b1, the host controller schedules transactions for a Control transfer in different microframes/frames.
16 RESBWHSEPS	Reserving 85% bandwidth for HS periodic EPs By default, HC reserves 80% of the bandwidth for periodic EPs. If this bit is set, the bandwidth is relaxed to 85% to accommodate two high-speed, high bandwidth ISOC EPs. USB 2.0 required 80% bandwidth allocated for ISOC traffic. If two high-bandwidth ISOC devices (HD Webcams) are connected, and if each requires 1024-bytes X 3 packets per micro-frame, then the bandwidth required is around 82%. If this bit is set, then it is possible to connect two Webcams of 1024 bytes X 3 paylod per micro-frame each. Otherwise, the resolution of the Webcams needs to be reduced. This bit is valid in host and DRD configuration and is used in host mode operation only. Ignore this bit in device mode.
15 CMDEVADDR	Compliance mode for device address When this bit is 1'b1, Slot ID may have different value than device address if max_slot_enabled < 128. The xHCI compliance requires this bit to be set to 1. The 0 mode is for debug purpose only. This allows to easily identify a device connected to a port in the Lecroy or Eliisys trace during hardware debug. This bit is valid in host and DRD configuration and is used in host mode operation only. Ignore this bit in device mode. 0b - Device address is equal to Slot ID 1b - Increment device address on each address device command
14 USBHSTINAUTORETRYEN	Host IN auto retry When set, this bit enables the auto retry feature. For IN transfers (non-isochronous) that encounter data packets with CRC errors or internal overrun scenarios, the auto retry feature causes the host core to reply to the device with a non-terminating retry ACK (that is, an ACK transaction packet with RETRY = 1 and NUMP != 0). If the auto retry feature is disabled, the core responds with a terminating retry ACK (that is, an ACK transaction packet with RETRY = 1 and NUMP = 0). **NOTE:** This bit is also applicable to the device mode. 0b - Auto retry disabled (default) 1b - Auto retry enabled
13 ENOVERLAPCHK	Enable check for LFPS overlap during remote Ux exit 0b - When the link exists U1/U2/U3 because of a remote exit, it does not look for an LFPS overlap. 1b - The SuperSpeed link when exiting U1/U2/U3 waits for either the remote link LFPS or TS1/TS2 training symbols before it confirms that the LFPS handshake is complete. This is done to handle the case where the LFPS glitch causes the link to start exiting from the low power state. Looking for the LFPS overlap makes sure that the link partner also sees the LFPS.
12 EXTCAPSUPTEN	External extended capability support enable When set, this bit enables extended capabilities to be implemented outside the core. A read to the first DWORD of the last internal extended capability (the "xHCI supported protocol capability for USB 3.0 when the Debug Capability is not enabled, or the "Debug Capability" when it is enabled) returns a value of 4 in the Next Capability Pointer bit. This indicates to software that there is another capability for DWORDs after this capability (for example, at address N+16 where N is the address of this DWORD). If enabled, an external address decoder that snoops the xHC slave interface needs to be implemented. If it

Table continues on the next page...

Field	Function
	sees an access to N+16 or greater, the slave access is re-routed to a piece of hardware which returns the external capability pointer register of the new capability and also handles reads/writes to this new capability and the side effects. If disabled, a read to the first DWORD of the last internal extended capability returns 0 in the 'Next Capability Pointer' bit. This indicates there are no more capabilities.
11 INSRTEXTRFSBODI	Insert extra delay between FS bulk OUT transactions Some FS devices are slow to receive bulk OUT data and can get stuck when there are consecutive bulk OUT transactions with short inter-transaction delays. This bit is used to control whether the host inserts extra delay between consecutive bulk OUT transactions to a FS endpoint. 0b - Host doesn't insert extra delay between consecutive bulk OUT transactions to a FS Endpoint 1b - Host inserts about 12 µs extra delay between consecutive bulk OUT transactions to a FS Endpoint to work around the device issue. Setting this bit to 1 reduces the bulk OUT transfer performance for most of the FS devices.
10-9 DTCT	Device timeout coarse tuning This bit is a host mode parameter which determines how long the host waits for a response from device before considering a timeout. The core first checks the DTCT value. If it is 0, then the timeout value is defined by the DTFT. If it is non-zero, then it uses the following Timeout values: 00b - 0 µs -> use DTFT value instead 01b - 500 µs 10b - 1.5 µs 11b - 6.5 µs
8-0 DTFT	Device timeout fine tuning This bit is a host mode parameter which determines how long the host waits for a response from device before considering a timeout. For DTFT bit to take effect, DTCT must be set to 2'b00. The DTFT value is the number of 125 MHz clocks * 256 to count before considering a device timeout. For the 125 MHz clock (8 ns period), this is calculated as follows- (DTFT value) * 256 * (8 ns) Quick Reference: • if DTFT = 0x2, 2*256*8 = 4 µs timeout • if DTFT = 0x5, 5*256*8 = 10 µs timeout • if DTFT = 0xA, 10*256*8 = 20 µs timeout • if DTFT = 0x10, 16*256*8 = 32 µs timeout • if DTFT = 0x19, 25*256*8 = 51 µs timeout • if DTFT = 0x31, 49*256*8 = 100 µs timeout • if DTFT = 0x62, 98*256*8 = 200 µs timeout

33.2.19 Global SoC bus error address register low (GBUSERRADDRLO)

33.2.19.1 Offset

Register	Offset
GBUSERRADDRLO	C130h

33.2.19.2 Function

When the AXI master bus returns "Error" response, the "SoC Bus Error" is generated. In the host mode, the host_system_err port indicates this condition. In addition, it is also indicated in the USBSTS[HSE] bit. In the device mode, the GSTS[BUSERRADDRVLD] bit is the only indication of the SoC bus error.

Due to the nature of AXI, it is possible that multiple AXI transactions are active at a time. The USB 3.0 controller does not keep track of the start address of all outstanding transactions. Instead, it keeps track of the start address of the DMA transfer associated with all active transactions. It is this address that is reported in the GBUSERRADDR when a bus error occurs.

For example, if the USB 3.0 controller initiates a DMA transfer to write 1K of packet data starting at buffer address 0xABCD0000, and this DMA is broken up into multiple 256B bursts on the AXI, then if a bus error occurs on any of these associated AXI transfers, the GBUSERRADDR reflects the DMA start address of 0xABCD0000 regardless of which AXI transaction received the error.

33.2.19.3 Diagram

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

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

33.2.19.4 Fields

Field	Function
31-0 BUSERRADDR	Bus address - low This 64-bit register contains the lower 32 bits of the first bus address that encountered a SoC bus error. It is valid when the GSTS[BUSERRADDRVLD] bit is 1. It can only be cleared by resetting the core.

33.2.20 Global SoC bus error address register high (GBUSERRADDRHI)

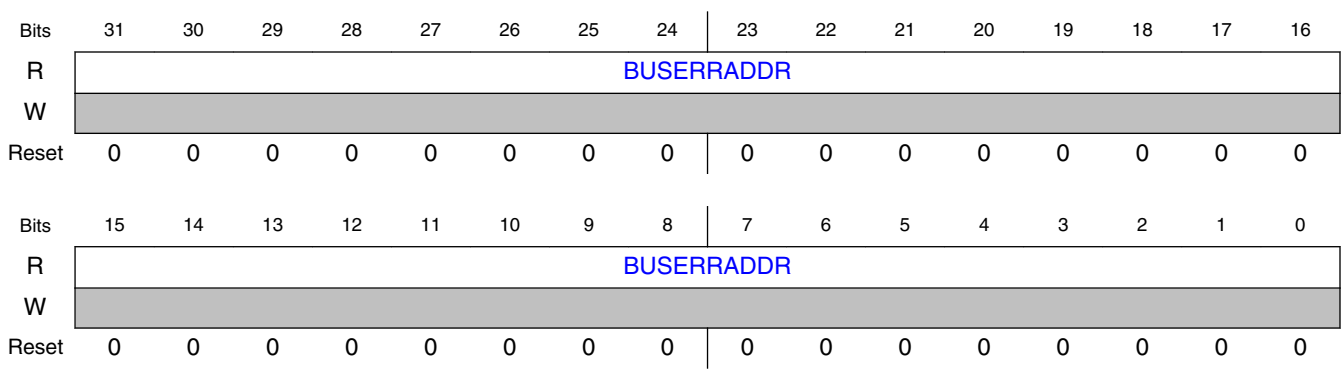
33.2.20.1 Offset

Register	Offset
GBUSERRADDRHI	C134h

33.2.20.2 Function

It represents the remanining bits of global SoC bus error address register.

33.2.20.3 Diagram



33.2.20.4 Fields

Field	Function
31-0 BUSERRADDR	Bus address - high This 64-bit register contains the higher 32 bits of the first bus address that encountered a SoC bus error. It is valid when the GSTS[BUSERRADDRVLD] bit is 1. It can only be cleared by resetting the core.

33.2.21 Global SS port to bus instance mapping register - low (GPRTBIMAPLO)

33.2.21.1 Offset

Register	Offset
GPRTBIMAPLO	C138h

33.2.21.2 Function

This is an alternate register for the GPRTBIMAP register.

NOTE

For reset values, refer to the corresponding values in the GPRTBIMAP register.

33.2.21.3 Diagram

Bits	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

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved												BINUM1			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

33.2.21.4 Fields

Field	Function
31-4 —	Reserved
3-0 BINUM1	SS USB instance number for port. Value set as 0.

33.2.22 Global SS port to bus instance mapping register - high (GPRTBIMAPHI)

33.2.22.1 Offset

Register	Offset
GPRTBIMAPHI	C13Ch

33.2.22.2 Function

This is an alternate register for the GPRTBIMAP register.

NOTE

For reset values, refer to the corresponding values in the GPRTBIMAP register.

33.2.22.3 Diagram

Bits	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

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved												BINUM9			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

33.2.22.4 Fields

Field	Function
31-4 —	Reserved
3-0	SS USB instance number for port 9.

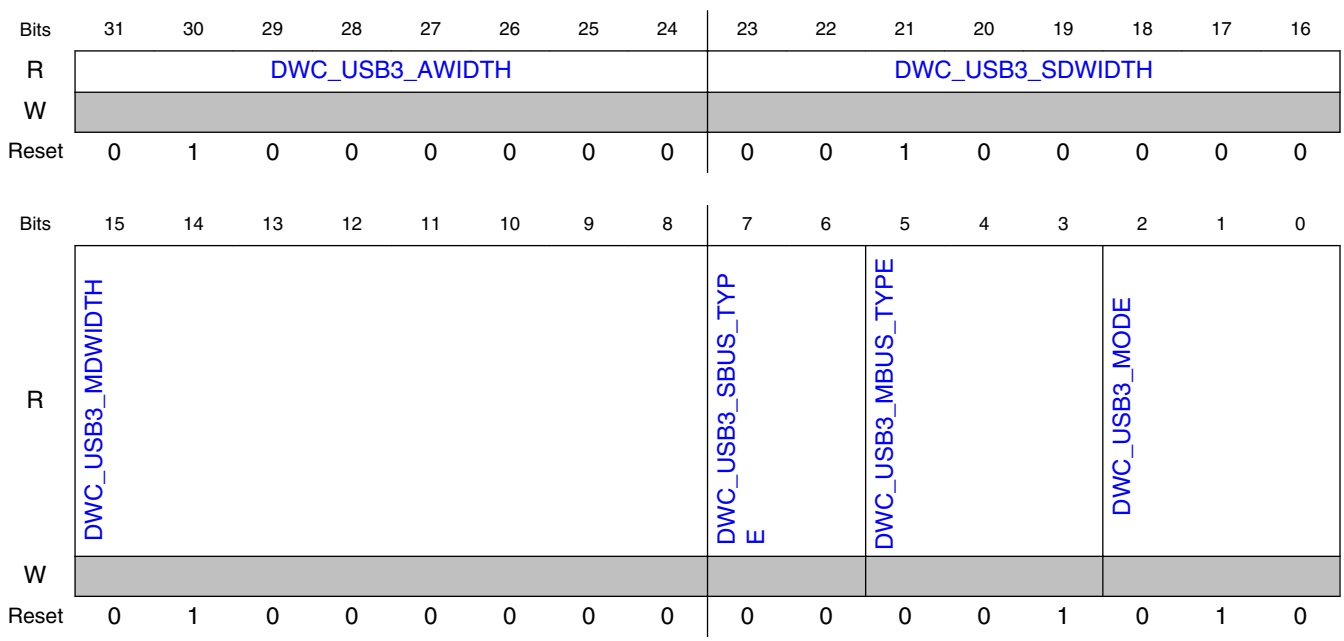
Field	Function
BINUM9	

33.2.23 Global hardware parameters register 0 (GHWPARAMS0)

33.2.23.1 Offset

Register	Offset
GHWPARAMS0	C140h

33.2.23.2 Diagram



33.2.23.3 Fields

Field	Function
31-24	Master/Slave address bus width: 64 bit

Table continues on the next page...

Field	Function
DWC_USB3_AWIDTH	
23-16 DWC_USB3_SDWIDTH	Slave bus (Register access bus) data bus width: 32 bit
15-8 DWC_USB3_MDWIDTH	Master bus (DMA bus) data bus width It selects the data bus width of the master bus interface. 33-bit option is used only for Hub configuration. The possible values are: • 32 32-bits • 33 33-bits • 64 64-bits • 128 128-bits
7-6 DWC_USB3_SBUS_TYPE	Slave bus (Register access bus) interface type It selects the chip slave bus interface type. The slave bus is used for register programming. The settings not shown are reserved. 00b - AHB
5-3 DWC_USB3_MBUS_TYPE	Master bus (DMA bus) interface type It selects the chip master bus interface type. The master bus is used for DMA. The settings not shown are reserved. 001b - AXI
2-0 DWC_USB3_MODE	Mode of operation It selects the controller mode for USB 3.0. **NOTE:** It is configurable based on license(s) purchased. The settings not shown are resereved. 010b - DRD

33.2.24 Global hardware parameters register 1 (GHWPARAMS1)

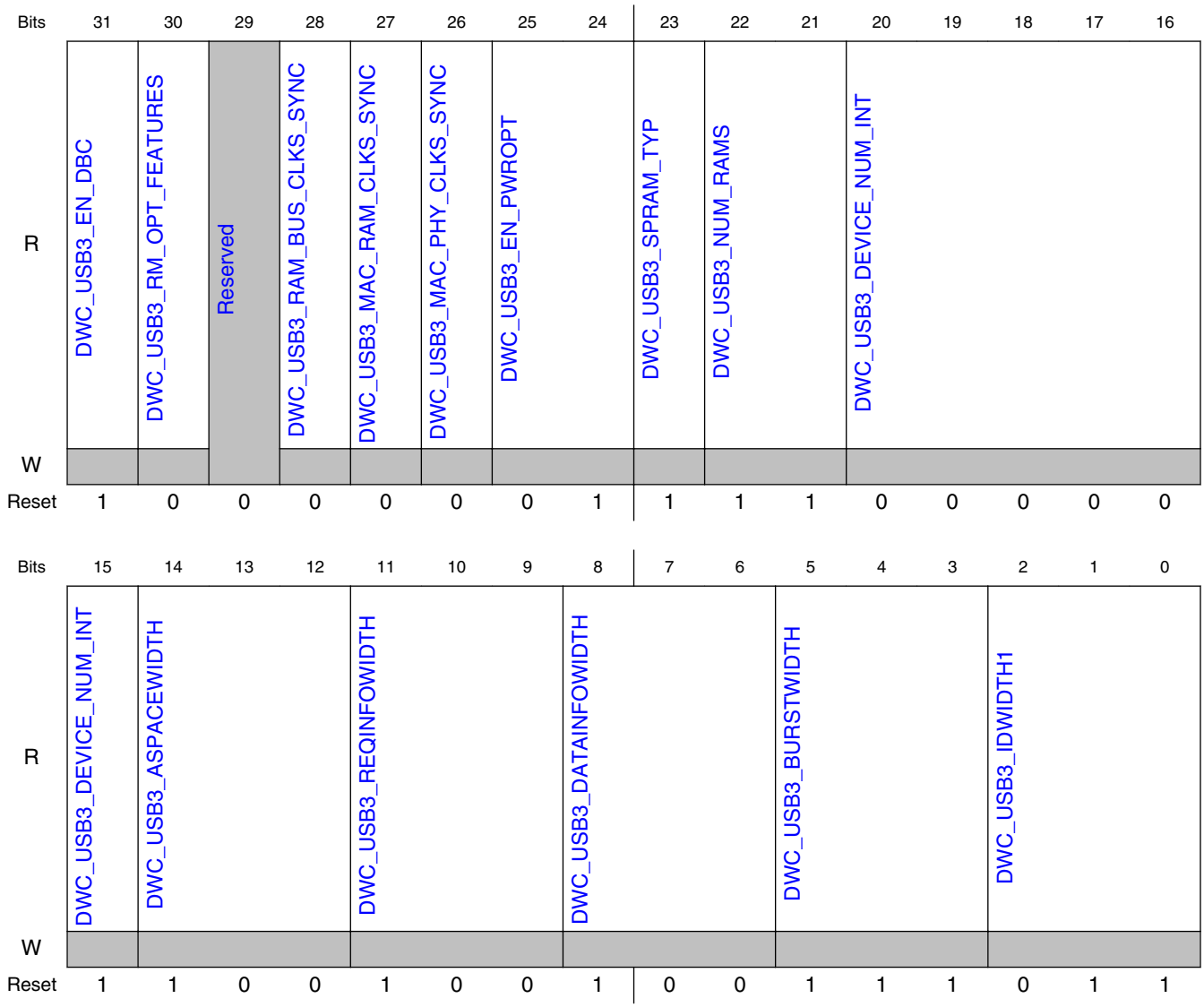
33.2.24.1 Offset

Register	Offset
GHWPARAMS1	C144h

33.2.24.2 Function

This register contains the hardware configuration options.

33.2.24.3 Diagram



33.2.24.4 Fields

Field	Function
31 DWC_USB3_EN_DBC	Enables xHCI debug capability 0b - No 1b - Yes
30	It specifies whether to remove optional features.

Table continues on the next page...

Field	Function
DWC_USB3_RM_OPT_FEATURES	When this parameter is enabled, the user ID register, General Purpose Input/Output ports, and SOF toggle and counter ports are removed. 0b - No 1b - Yes
29 —	Reserved
28 DWC_USB3_RAM_BUS_CLKS_SYNC	It specifies whether the RAM clock and the Bus clock are synchronous to each other. 0b - No 1b - Yes
27 DWC_USB3_MAC_RAM_CLKS_SYNC	It specifies whether the MAC clock and the RAM clock are synchronous to each other. 0b - No 1b - Yes
26 DWC_USB3_MAC_PHY_CLKS_SYNC	It specifies whether the MAC clock and the PHY clock are synchronous to each other. 0b - No 1b - Yes
25-24 DWC_USB3_EN_PWROPT	Power optimization mode It specifies the power optimization mode. If clock gating only is selected, RAM and PHY clocks are gated when the core is inactive during U1, U2, or U3 states. 00b - No power optimization 01b - Clock gating only 10b - Reserved 11b - Reserved
23 DWC_USB3_SPRAM_TYP	Synchronous static RAM type It selects the FIFO synchronous static RAM type. 0b - 2-port RAM (2Port-RAM) 1b - Single-port RAM (SPRAM)
22-21 DWC_USB3_NUM_RAMS	Number of RAMs It selects the number of RAMs. The possible values are 1, 2 and 3.
20-15 DWC_USB3_DEVICE_NUM_INT	Number of device mode event buffers It selects the number of event buffers in device mode. The possible values are 1, 2,..., and 32.
14-12 DWC_USB3_ASPACEWIDTH	It selects the address space port width of the master and slave bus interfaces. The possible values are 1, 2, 3, 4, 5 and 6.
11-9 DWC_USB3_REQINFOWIDTH	It selects the request/response info port width of the master and slave bus interfaces. The possible values are 4, 5 and 6.
8-6	It selects the data info port width of the master and slave bus interfaces. The possible values are 1, 2, 3, 4, 5 and 6.

Table continues on the next page...

Field	Function
DWC_USB3_DATAINFOWIDTH	
5-3 DWC_USB3_BURSTWIDTH	DWC_USB3_BURSTWIDTH 1 It selects the burst port width of the master and slave bus interfaces. The possible values are 1,2, 3, 4, 5, 6, 7 and 8.
2-0 DWC_USB3_IDWIDTH1	Master ID port width It selects the ID port width of the master bus interface. This parameter limits the number of pipelined AXI transfers. The GSBUSCFG1[PIPETRANSLIMIT] bit can only be programmed to a value less than or equal to two to the power of DWC_USB3_IDWIDTH. The possible values are 4, 5, 6, 7 and 8.

33.2.25 Global hardware parameters register 2 (GHWPARAMS2)

33.2.25.1 Offset

Register	Offset
GHWPARAMS2	C148h

33.2.25.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DWC_USB3_USERID															
W																
Reset	0	0	0	0	0	0	0	1	0	0	1	1	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DWC_USB3_USERID															
W																
Reset	0	0	1	0	1	0	0	1	0	0	0	0	1	0	1	0

33.2.25.3 Fields

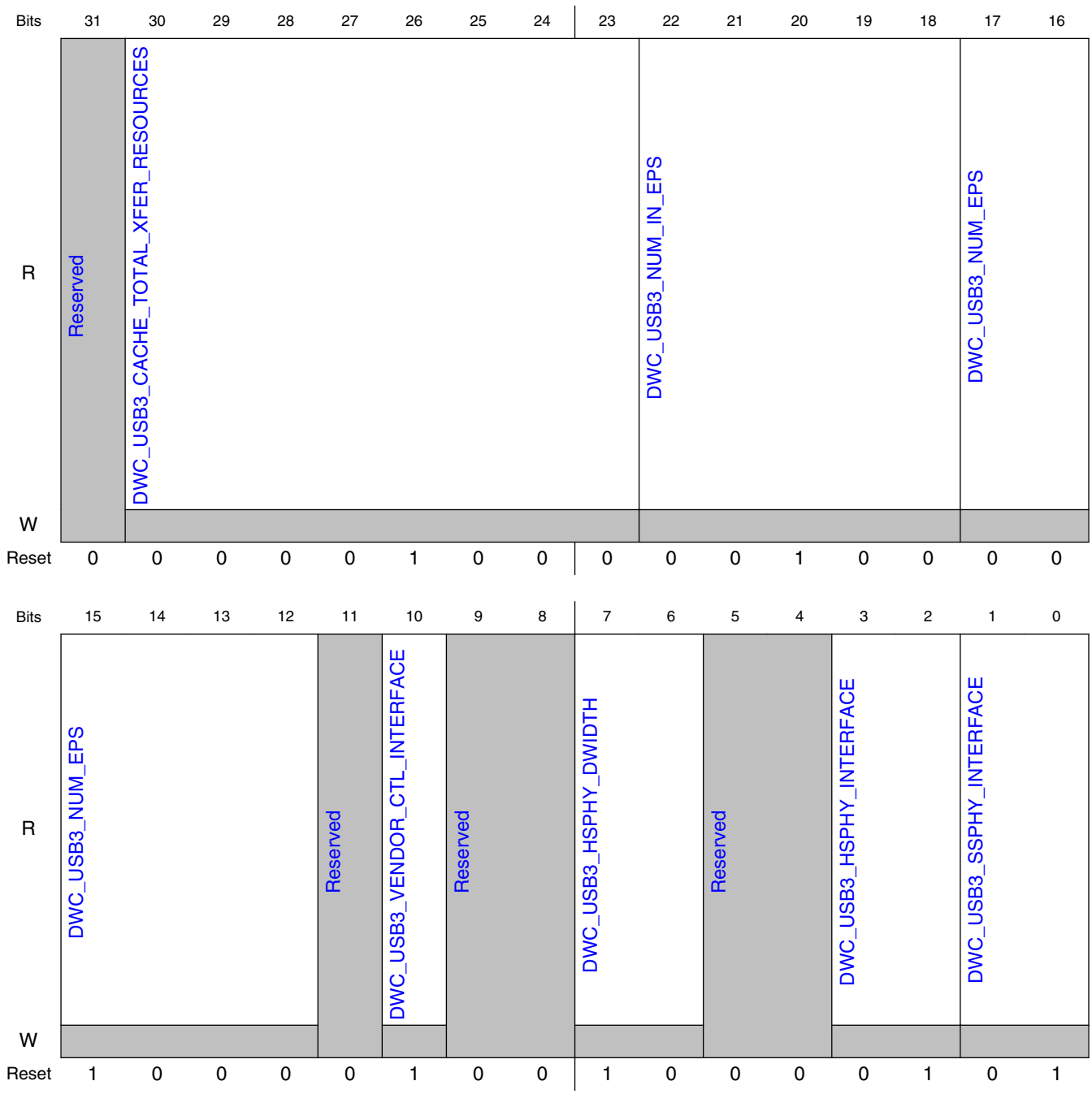
Field	Function
31-0 DWC_USB3_USERID	Global user ID (GUID) register's power-on-initialization value It specifies the global user ID (GUID) register's power-on-initialization value. The value is set as 32'h130290A.

33.2.26 Global hardware parameters register 3 (GHWPARAMS3)

33.2.26.1 Offset

Register	Offset
GHWPARAMS3	C14Ch

33.2.26.2 Diagram



33.2.26.3 Fields

Field	Function
31	Reserved

Table continues on the next page...

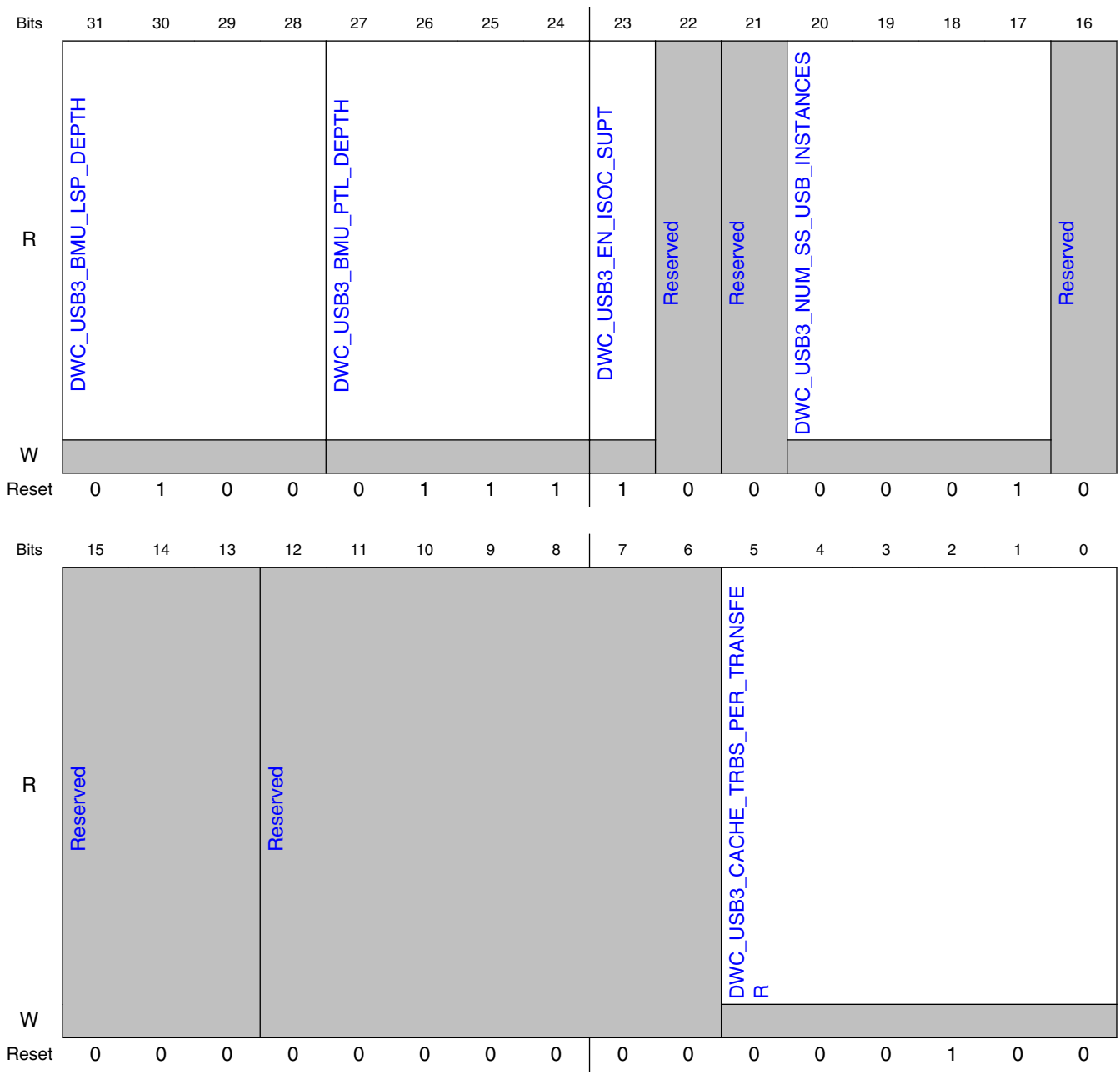
Field	Function
—	
30-23 DWC_USB3_CACHE_TOTAL_XFER_RESOURCES	It selects the maximum number of transfer resources in the core. The value is set as 8.
22-18 DWC_USB3_NUM_IN_EPS	Number of device mode active IN endpoints It specifies the maximum number of device mode IN endpoints active at any time, including control endpoint 0, which is always present. The value is set as 4.
17-12 DWC_USB3_NUM_EPS	Number of device mode endpoints It specifies the number of device mode single directional endpoints, including OUT and IN endpoint. The value is set as 8.
11 —	Reserved
10 DWC_USB3_VENDOR_CTL_INTERFACE	The bit enables the UTMI+ PHY vendor control interface. The value is enabled and value is set as 1.
9-8 —	Reserved
7-6 DWC_USB3_HSPHY_DWIDTH	It specifies the data width of the UTMI+ PHY interface. All other settings are reserved. 10b - 8-/16-bits
5-4 —	Reserved
3-2 DWC_USB3_HSPHY_INTERFACE	It specifies the high-speed PHY interface(s). The value is set as 1 for UTMI+.
1-0 DWC_USB3_SSPHY_INTERFACE	It specifies the superSpeed PHY interface. The value is set as 1 for PIPE3. In USB 2.0 only mode, set to 0 else select the PIPE3 interface.

33.2.27 Global hardware parameters register 4 (GHWPARAMS4)

33.2.27.1 Offset

Register	Offset
GHWPARAMS4	C150h

33.2.27.2 Diagram



33.2.27.3 Fields

Field	Function
31-28 DWC_USB3_BMU_LSP_DEPTH	It specifies the depth of the BMU-LSP status buffer. The value is set as 4.
27-24 DWC_USB3_BMU_PTL_DEPTH	It specifies the depth of the BMU-PTL source/sink buffers. The value is set as 8.
23 DWC_USB3_EN_ISOC_SUPT	It enables isochronous endpoint capability. By default, the value is set to 1 to enable this capability.
22 —	Reserved
21 —	Reserved
20-17 DWC_USB3_NUM_SS_USB_INSTANCES	Number of SuperSpeed USB bus instances It specifies the number of SuperSpeed USB bus instances. The value is set as 1.
16-13 —	Reserved
12-6 —	Reserved
5-0 DWC_USB3_CACHE_TRBS_PER_TRANSFER	Number of cached TRBs per transfer It selects the number of transfer request blocks (TRBs) per transfer that can be cached within the core. The value is set as 4.

33.2.28 Global hardware parameters register 5 (GHWPARAMS5)

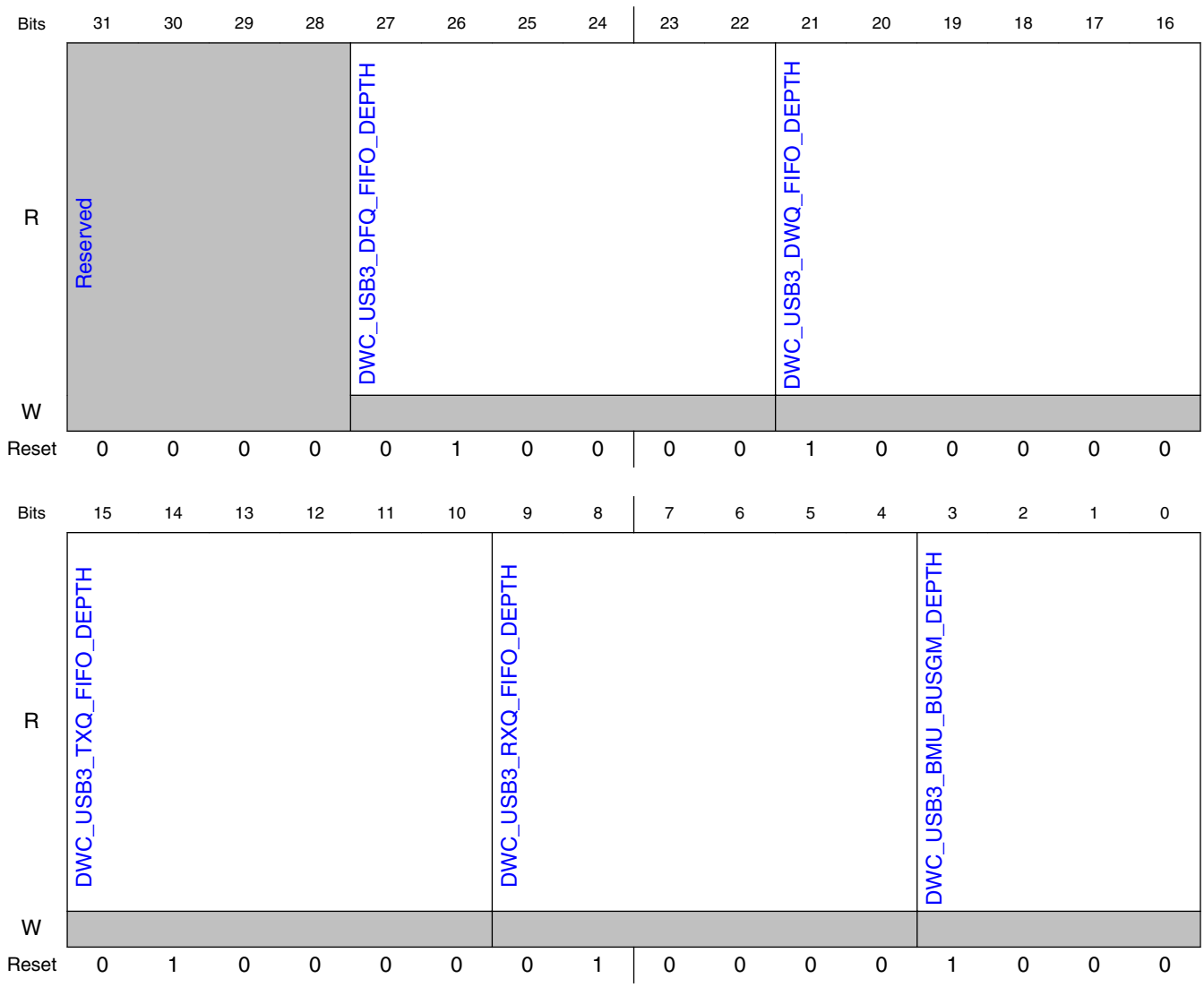
33.2.28.1 Offset

Register	Offset
GHWPARAMS5	C154h

33.2.28.2 Function

The register is read-only.

33.2.28.3 Diagram



33.2.28.4 Fields

Field	Function
31-28	Reserved

Table continues on the next page...

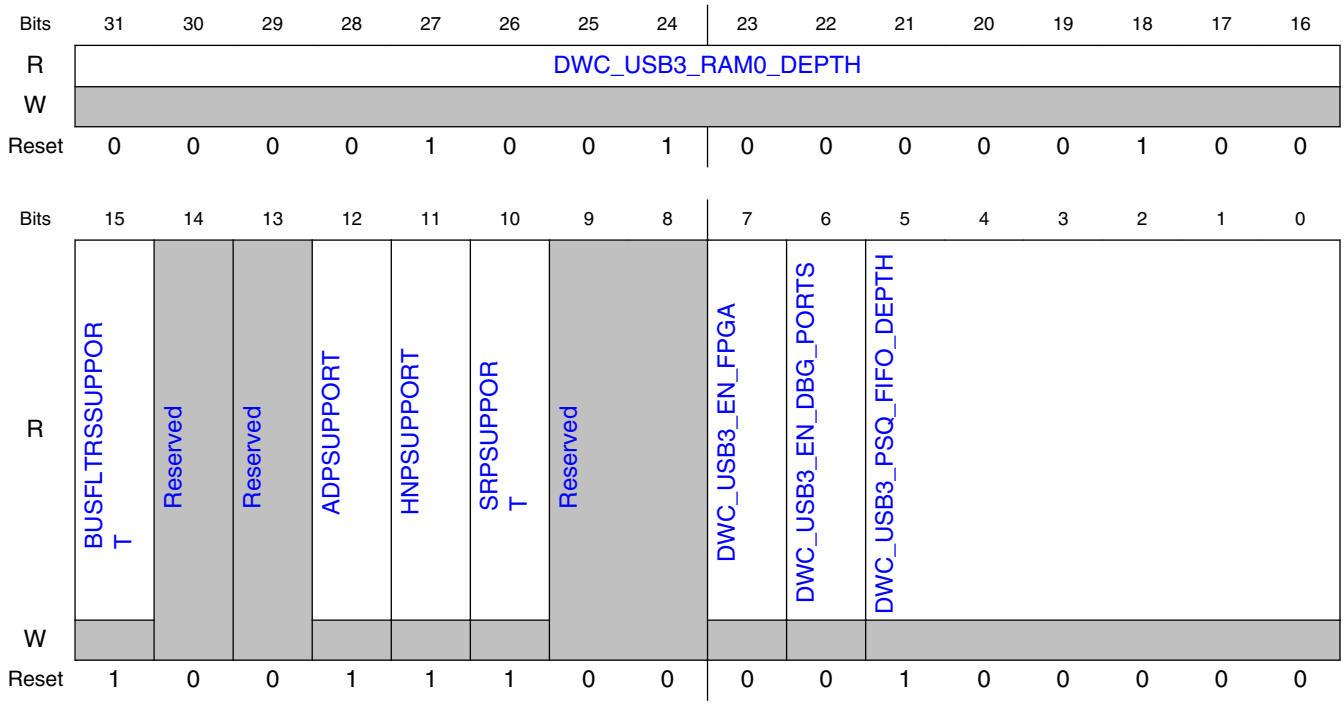
Field	Function
—	
27-22 DWC_USB3_DFQ_FIFO_DEPTH	It specifies the size of the BMU descriptor fetch request queue. The specified depth is allocated in the data FIFO RAM and defines the number of descriptor fetch commands the scheduler can queue to the BMU. The value is set as 16.
21-16 DWC_USB3_DWQ_FIFO_DEPTH	It specifies the size of the BMU descriptor write queue. The specified depth is allocated in the data FIFO RAM (32-/64-/128-bit wide) and defines the number of descriptor write commands the scheduler can queue to the BMU. value is set as 32.
15-10 DWC_USB3_TXQ_FIFO_DEPTH	It specifies the size of the BMU Tx request queue. The specified depth is allocated in the data FIFO RAM (32-/64-/128-bit wide) and defines the number of Tx commands the scheduler can queue to the BMU. The value is set as 16.
9-4 DWC_USB3_RXQ_FIFO_DEPTH	It specifies the size of the BMU Rx request queue. The specified depth is allocated in the data FIFO RAM (32-/64-/128-bits wide) and defines the number of Rx commands the scheduler can queue to the BMU. The value is set as 16.
3-0 DWC_USB3_BMU_BUSGM_DEPTH	It specifies the depth of the BMU-BUSGM source/sink buffers. The FIFOs are 32-/64-/128-bit wide, matching the bus master data width. The value is set as 8.

33.2.29 Global hardware parameters register 6 (GHWPARAMS6)

33.2.29.1 Offset

Register	Offset
GHWPARAMS6	C158h

33.2.29.2 Diagram



33.2.29.3 Fields

Field	Function
31-16 DWC_USB3_RAM0_DEPTH	Total RAM0 depth. The value is set as 2308. It specifies the depth of RAM0. In device, host, and DRD configuration, RAM0 contains: • 3-RAM configuration: Descriptor cache • 2-RAM configuration: Descriptor cache and RXFIFOs • 1-RAM configuration: Descriptor cache, TXFIFOs, and RXFIFOs
15 BUSFLTRSSUPPORT	It specifies whether to add a filter for VBUS and ID related control inputs from the PHY. • UTMI+ PHY: This signal is from the PHY. 0b - No 1b - Yes
14 —	Reserved
13 —	Reserved
12 ADPSUPPORT	It enables internal ADP capability of the USB 3.0 core. When it is enabled, the core incorporates ADP controller logic and provides ADP control signals.

Table continues on the next page...

Field	Function
11 HNPSUPPORT	HNP support enabled The application uses this bit to determine the USB 3.0 core's HNP support. 0b - HNP support is not enabled 1b - HNP support is enabled
10 SRPSUPPORT	SRP support enabled The application uses this bit to determine the USB 3.0 core's SRP support. 0b - SRP support is not enabled 1b - SRP support is enabled
9-8 —	Reserved
7 DWC_USB3_EN_FPGA	Hardware validation/driver development with an FPGA platform 0b - No 1b - Yes
6 DWC_USB3_EN_DBG_PORTS	It is used for FPGA hardware validation of the core. 0b - No 1b - Yes
5-0 DWC_USB3_PSQ_FIFO_DEPTH	It specifies the size of the BMU protocol status queue. The value is set as 32. The specified depth is allocated in the data FIFO RAM 32-/64-/128-bit wide) and defines the number of header and status DWORDs the PTL can queue to the LSP.

33.2.30 Global hardware parameters register 7 (GHWPARAMS7)

33.2.30.1 Offset

Register	Offset
GHWPARAMS7	C15Ch

33.2.30.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DWC_USB3_RAM2_DEPTH															
W																
Reset	0	0	0	0	0	0	1	1	0	0	0	0	1	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DWC_USB3_RAM1_DEPTH															
W																
Reset	0	0	0	0	0	1	0	0	0	1	0	0	1	1	0	1

33.2.30.3 Fields

Field	Function
31-16 DWC_USB3_RAM2_DEPTH	Total RAM2 depth. It specifies the depth of RAM2. The value is set as 776.
15-0 DWC_USB3_RAM1_DEPTH	Total RAM1 depth. It specifies the depth of RAM1. The value is set as 1101.

33.2.31 Global high-speed port to bus instance mapping register - low (GPRTBIMAP_HSLO)

33.2.31.1 Offset

Register	Offset
GPRTBIMAP_HSLO	C180h

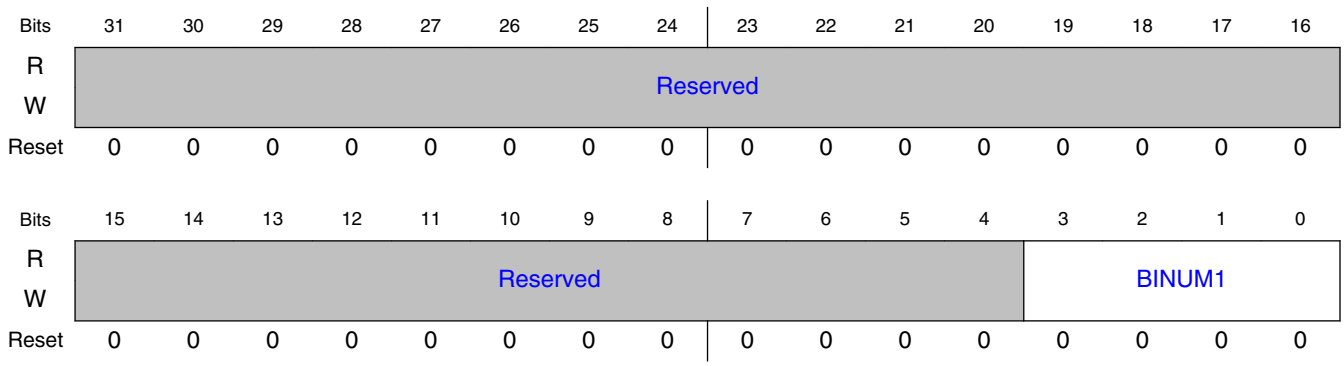
33.2.31.2 Function

This is an alternate register for the GPRTBIMAP_HS register.

NOTE

For reset values, refer to the corresponding values in the GPRTBIMAP_HS register.

33.2.31.3 Diagram



33.2.31.4 Fields

Field	Function
31-4 —	Reserved
3-0 BINUM1	HS USB instance number for port 1. The value is set as 0.

33.2.32 Global high-speed port to bus instance mapping register - high (GPRTBIMAP_HSHI)

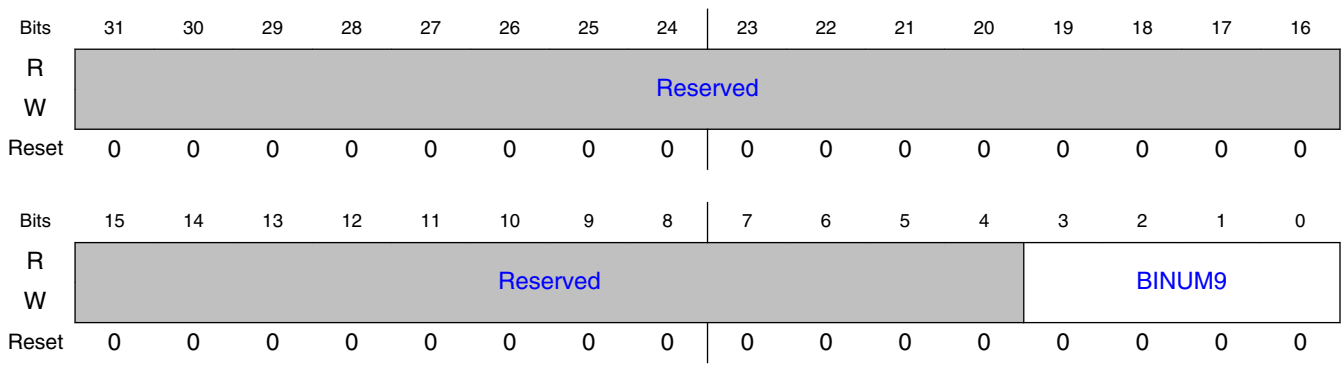
33.2.32.1 Offset

Register	Offset
GPRTBIMAP_HSHI	C184h

33.2.32.2 Function

This is an alternate register for the GPRTBIMAP_HS register.

33.2.32.3 Diagram



33.2.32.4 Fields

Field	Function
31-4 —	Reserved
3-0 BINUM9	HS USB instance number for port 9.

33.2.33 Global USB2 PHY configuration register (GUSB2PHYCFG)

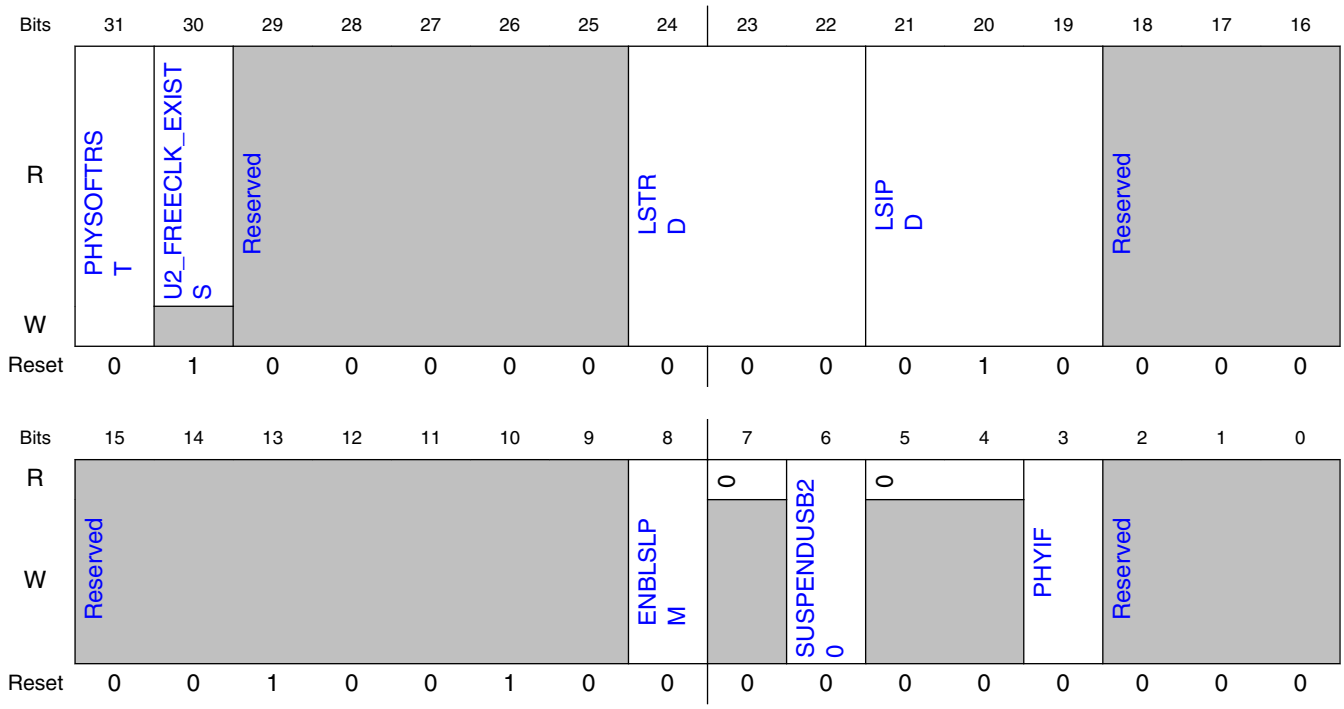
33.2.33.1 Offset

Register	Offset
GUSB2PHYCFG	C200h

33.2.33.2 Function

The SoC must program this register before starting any transaction.

33.2.33.3 Diagram



33.2.33.4 Fields

Field	Function
31 PHYSOFTRST	UTMI PHY soft reset. It causes the usb2phy_reset signal to be asserted to reset a UTMI PHY.
30 U2_FREECLK_EXISTS	Specifies USB 2.0 PHY free-running PHY clock exists. The value is set as 1.
29-25 —	Reserved
24-22 LSTRD	LS turnaround time. This bit indicates the value of the Rx-to-Tx packet gap for LS devices. **NOTE:** This bit is applicable only in host mode. For normal operation, to work with most LS devices, the default value is set as 0 (2-bit times).

Table continues on the next page...

Field	Function
	The programmable LS device inter-packet gap and turnaround delays are provided to support some legacy LS devices that might require different delays than the default/fixed ones. For instance, the Open LS mouse requires 3-bit times of inter-packet gap to work correctly. Include the PHY delays when programming the LSIPD/LSTRDTIM values. For example, if the PHY's TxEndDelay in LS mode is 30 UTMI CLKs, then subtract this delay (~1 LS bit time) from the device's delay requirement. 000b - 2-bit times 001b - 2.5-bit times 010b - 3-bit times 011b - 3.5-bit times 100b - 4-bit times 101b - 4.5-bit times 110b - 5-bit times 111b - 5.5-bit times
21-19 LSIPD	LS inter-packet time. This bit indicates the value of Tx-to-Tx packet gap for LS devices. **NOTE:** This bit is applicable only in host mode. For normal operation to work with most LS devices the bit is set to 2 (3-bit times). The programmable LS device inter-packet gap and turnaround delays are provided to support some legacy LS devices that might require different delays than the default/fixed ones. For instance, the Open LS mouse requires 3-bit times of inter-packet gap to work correctly. Include the PHY delays when programming the LSIPD/LSTRDTIM values. For example, if the PHY's TxEndDelay in LS mode is 30 UTMI clocks, then subtract this delay (~1 LS bit time) from the device's delay requirement. 000b - 2-bit times 001b - 2.5-bit times 010b - 3-bit times 011b - 3.5-bit times 100b - 4-bit times 101b - 4.5-bit times 110b - 5-bit times 111b - 5.5-bit times
18-9 —	Reserved
8 ENBLSLPM	Enable utmi_sleep_n and utmi_l1_suspend_n. The application uses this bit to control utmi_sleep_n and utmi_l1_suspend_n assertion to the PHY in the L1 state. 0b - utmi_sleep_n and utmi_l1_suspend_n assertion from the core is not transferred to the external PHY 1b - utmi_sleep_n and utmi_l1_suspend_n assertion from the core is transferred to the external PHY
7 —	Reserved This bit should always read 0.
6 SUSPENDUSB20	Suspend USB2.0 HS/FS/LS PHY When set, USB2.0 PHY enters suspend mode if suspend conditions are valid. If it is set to 1, then the application should clear this bit after power-on-reset. Application needs to set it to 1 after the core initialization is completed. **NOTE:** In host mode, on reset, this bit is set to 1. Software can override this bit after reset.
5-4 —	Reserved This bit should always read 0.
3 PHYIF	PHY interface

Table continues on the next page...

Field	Function
	If UTMI+ is selected, the application uses this bit to configure the core to support a UTMI+ PHY with an 8-bit or 16-bit interface. 0b - 8-bit interface 1b - 16-bit interface
2-0 —	Reserved

33.2.34 Global USB 3.0 PIPE control register (GUSB3PIPECTL)

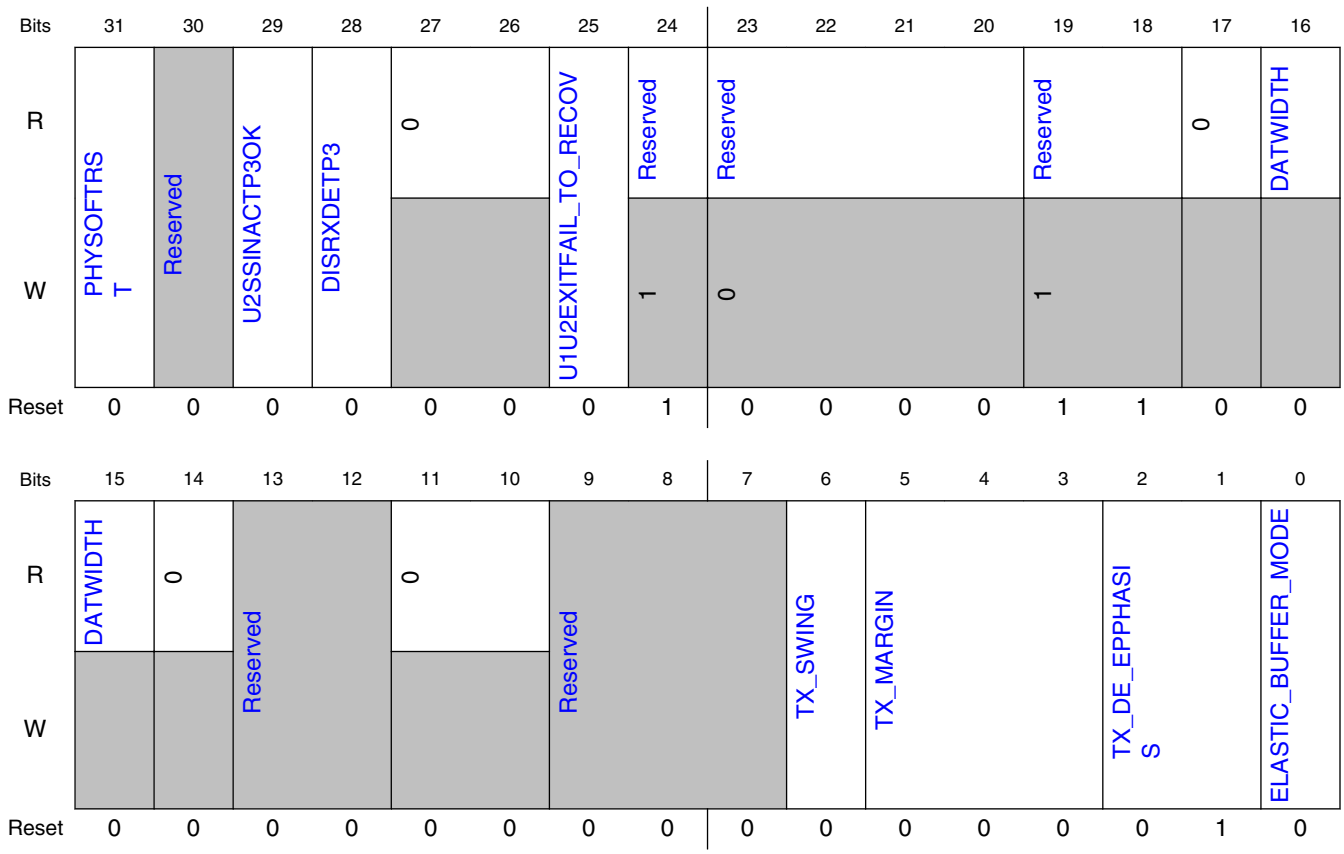
33.2.34.1 Offset

Register	Offset
GUSB3PIPECTL	C2C0h

33.2.34.2 Function

The application uses this register to configure the USB3 PHY and PIPE interface.

33.2.34.3 Diagram



33.2.34.4 Fields

Field	Function
31 PHYSOFTRST	USB3 PHY soft reset. After setting this bit to 1, the software needs to clear this bit.
30 —	Reserved
29 U2SSINACTP3OK	P3 OK for U2/SSInactive 0b - During link state U2/SS.Inactive, put PHY in P2 (default) 1b - During link state U2/SS.Inactive, put PHY in P3
28 DISRXDETP3	Disabled receiver detection in P3 0b - If PHY is in P3 and core needs to perform receiver detection, the core performs receiver detection in P3 (default) 1b - If PHY is in P3 and core needs to perform receiver detection, the core changes PHY power state to P2 and then perform receiver detection. After receiver detection, the core changes PHY power state to P3
27-26	Reserved

Table continues on the next page...

Field	Function
—	The bit should always read 0.
25 U1U2EXITFAIL_TO_RECOV	U1U2exitfail to recovery When set, and U1/U2 LFPS handshake fails, the LTSSM transitions from U1/U2 to recovery instead of SS inactive. If recovery fails, then the LTSSM can enter SS.Inactive. This is an enhancement only. It prevents interoperability issue if the remote link does not do proper handshake.
24 —	Reserved This bit should be always set to 1.
23-20 —	Reserved
19-18 —	Reserved This bit is read-only and should be always set to 1.
17 SUSPENDENABLE	Suspend USB3.0 SS PHY When set, and if suspend conditions are valid, the USB 3.0 PHY enters suspend mode. The value is set as 0.
16-15 DATWIDTH	PIPE data width One clock after reset, these bits receive the value. 00b - 32 bits 01b - 16 bits 10b - 8 bits
14 —	Reserved. This bit should always read 0.
13-12 —	Reserved
11-10 —	Reserved This bit should always read 0.
9-7 —	Reserved
6 TX_SWING	Tx swing. Refer to the PIPE3 specification.
5-3 TX_MARGIN	Tx margin[2:0]. Refer to Table 5-3 of the PIPE3 specification.
2-1 TX_DE_EPPHASIS	Tx deemphasis The value driven to the PHY is controlled by the LTSSM during USB3 Compliance mode. (Refer to Table 5-3 of the PIPE3 specification.)
0 ELASTIC_BUFFER_MODE	Elastic buffer mode

33.2.35 Global transmit FIFO size register (GTXFIFOSIZ_0 - GTXFIFOSIZ_3)

33.2.35.1 Offset

Register	Offset
GTXFIFOSIZ_0	C300h
GTXFIFOSIZ_1	C310h
GTXFIFOSIZ_2	C320h
GTXFIFOSIZ_3	C330h

33.2.35.2 Function

This register specifies the RAM start address and depth

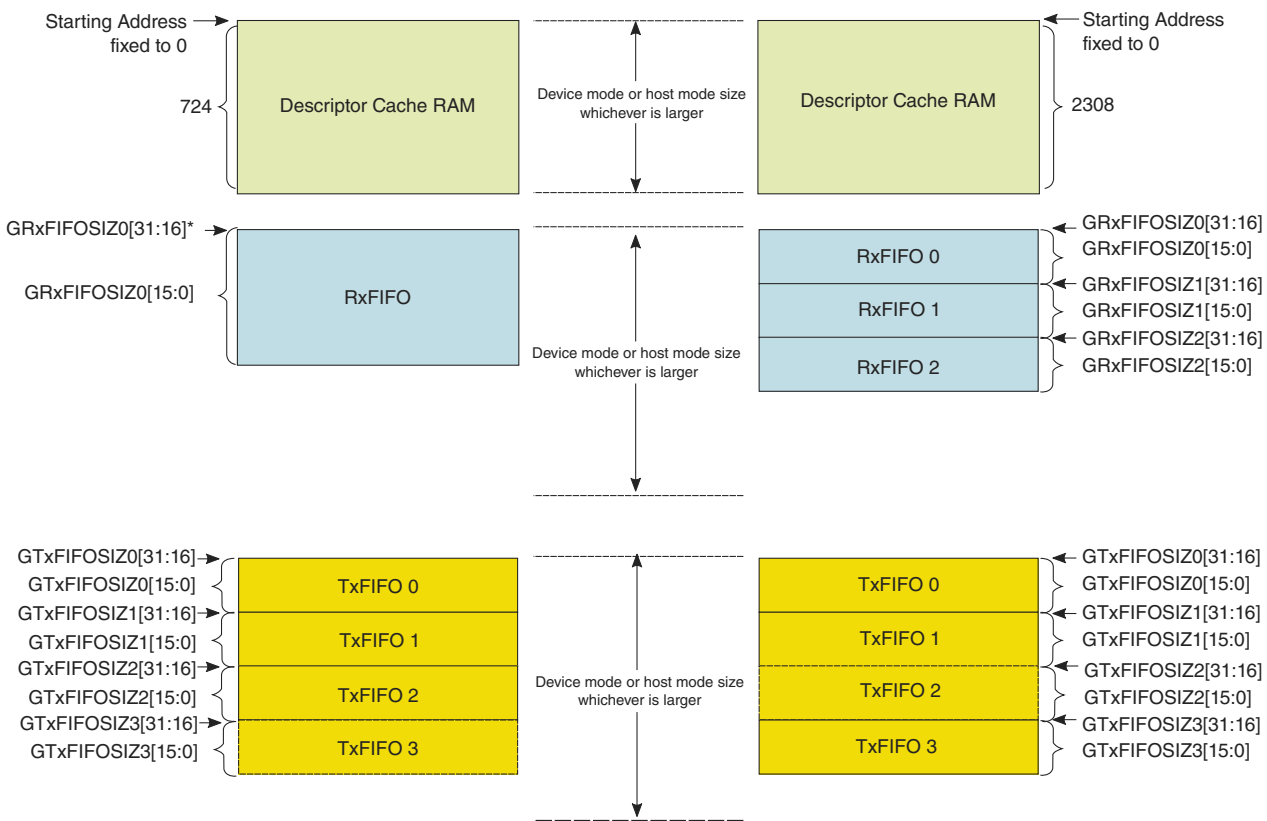


Figure 33-3. GTxFIFOSIZn and GRxFIFOSIZn usage in DRD mode without debug capability

33.2.35.3 Diagram

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

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

33.2.35.4 Fields

Field	Function
31-16 TXFSTADDR_N	Transmit FIFOn RAM start address This bit contains the memory start address for TXFIFOn in MDWIDTH-bit words.
15-0 TXFDEP_N	TXFIFO depth This bit contains the depth of TXFIFOn in MDWIDTH-bit words. Minimum value: 32 Maximum value: 32,768

33.2.36 Global receive FIFO size register (GRXFIFOSIZ_0 - GRXFIFOSIZ_2)

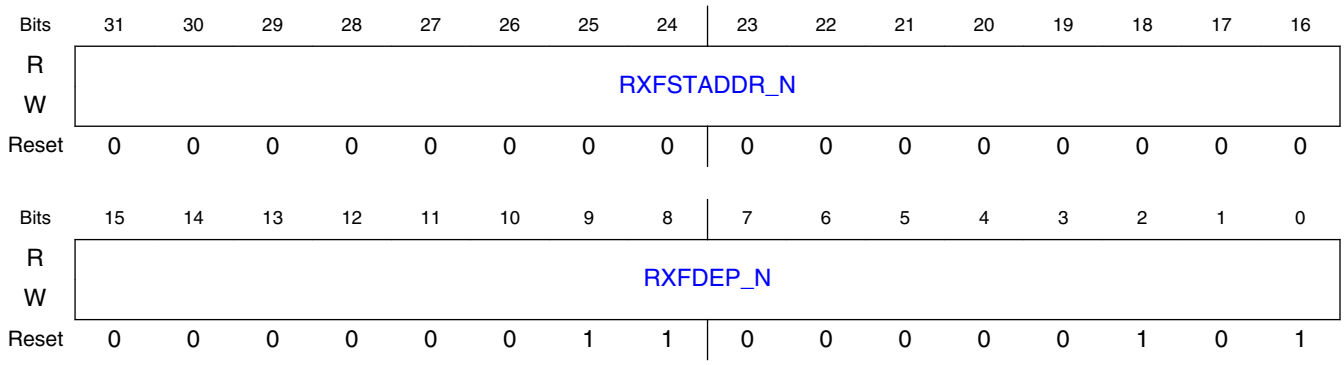
33.2.36.1 Offset

Register	Offset
GRXFIFOSIZ_0	C380h
GRXFIFOSIZ_1	C390h
GRXFIFOSIZ_2	C3A0h

33.2.36.2 Function

This register specifies the RAM start address and depth.

33.2.36.3 Diagram



33.2.36.4 Fields

Field	Function
31-16 RXFSTADDR_N	RXFIFOn RAM start address This bit contains the memory start address for RXFIFOn in MDWIDTH-bit words.
15-0 RXFDEP_N	RXFIFO depth This bits contains the depth of RXFIFOn in MDWIDTH-bit words. Minimum value: 32 Maximum value: 16,384

33.2.37 Global event buffer address (low) register (GEVNTADR LO)

33.2.37.1 Offset

Register	Offset
GEVNTADRLO	C400h

33.2.37.2 Function

This register holds the event buffer DMA address pointer. Software must initialize this address once during power-on-initialization. Software must not change the value of this register after it is initialized. Software must only use the GEVNTCOUNT register for event processing.

33.2.37.3 Diagram

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

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

33.2.37.4 Fields

Field	Function
31-0 EVNTADRLO	Event buffer address Holds the lower 32 bits of start address of the external memory for the event buffer. During operation, hardware does not update this address.

33.2.38 Global event buffer address (high) register (GEVNTADRHI)

33.2.38.1 Offset

Register	Offset
GEVNTADRHI	C404h

33.2.38.2 Function

This register holds the event buffer DMA address pointer. Software must initialize this address once during power-on-initialization. Software must not change the value of this register after it is initialized. Software must only use the GEVNTCOUNT register for event processing.

33.2.38.3 Diagram

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

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

33.2.38.4 Fields

Field	Function
31-0 EVNTADRHI	Event buffer address Holds the higher 32 bits of start address of the external memory for the event buffer. During operation, hardware does not update this address.

33.2.39 Global event buffer size register (GEVNTSIZ)

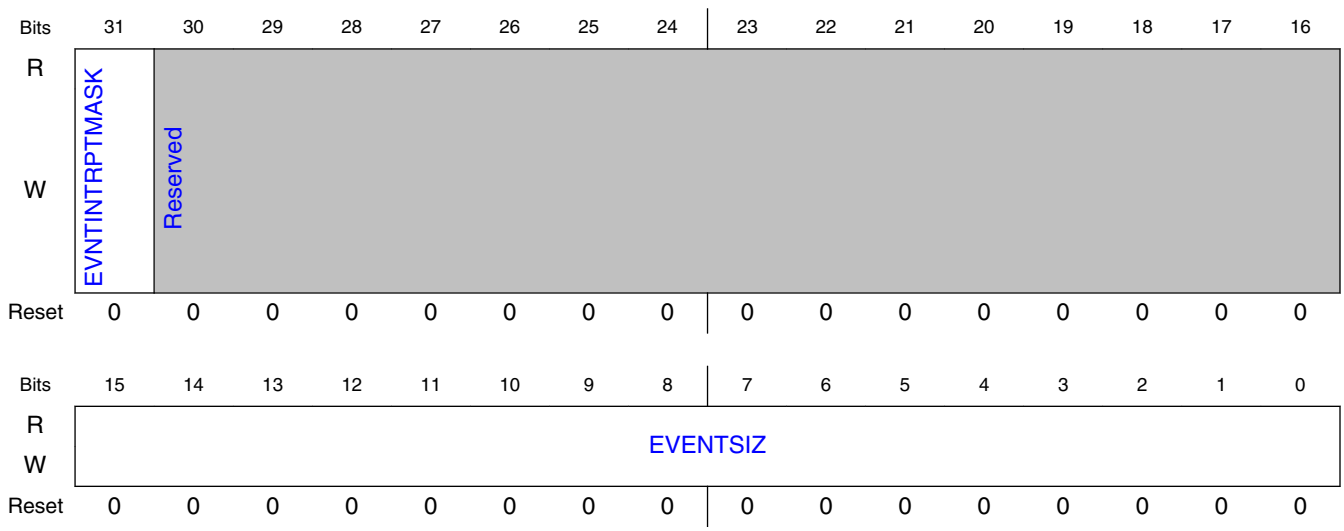
33.2.39.1 Offset

Register	Offset
GEVNTSIZ	C408h

33.2.39.2 Function

This register holds the event buffer size and the event interrupt mask bits. During power-on-initialization, software must initialize the size with the number of bytes allocated for the event buffer. The event interrupt mask bit masks the interrupt, but events are still queued. After configuration, software must preserve the event buffer size value when changing the event interrupt mask.

33.2.39.3 Diagram



33.2.39.4 Fields

Field	Function
31 EVNTINTRPTMASK	Event interrupt mask When set to 1, this prevents the interrupt from being generated. However, even when the mask is set, the events are queued.
30-16 —	Reserved
15-0 EVENTSIZ	Event buffer size in bytes Holds the size of the event buffer in bytes; must be a multiple of four. This is programmed by software once during initialization. The minimum size of the event buffer is 32 bytes.

33.2.40 Global event buffer count register (GEVNTCOUNT)

33.2.40.1 Offset

Register	Offset
GEVNTCOUNT	C40Ch

33.2.40.2 Function

This register holds the number of valid bytes in the event buffer. During initialization, software must initialize the count by writing 0 to the event count bit. Each time the hardware writes a new event to the event buffer, it increments this count. Most events are four bytes, but some events may span over multiple four byte entries. Whenever the count is greater than zero, the hardware raises the corresponding interrupt line (depending on the GEVNTSIZ[EVNTINTRPTMASK] bit). On an interrupt, software processes one or more events out of the event buffer. Afterwards, software must write the event count bit with the number of bytes it processed.

Clock crossing delays may result in the interrupt's continual assertion after software acknowledges the last event. Therefore, when the interrupt line is asserted, software must read the GEVNTCOUNT register and only process events if the GEVNTCOUNT is greater than 0.

33.2.40.3 Diagram

Bits	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

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

33.2.40.4 Fields

Field	Function
31-16 —	Reserved
15-0 EVNTCOUNT	Event count When read, returns the number of valid events in the event buffer (in bytes). When written, hardware decrements the count by the value written. The interrupt line remains high when count is not 0.

33.2.41 Global hardware parameters register 8 (GHWPARAMS8)

33.2.41.1 Offset

Register	Offset
GHWPARAMS8	C600h

33.2.41.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DWC_USB3_DCACHE_DEPTH_INFO															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DWC_USB3_DCACHE_DEPTH_INFO															
W																
Reset	0	0	0	0	1	0	0	1	0	0	0	0	0	1	0	0

33.2.41.3 Fields

Field	Function
31-0	The read-only value defines the minimum RAM0 requirement. The value of the bit is set as 2308.

Field	Function
DWC_USB3_DCACHE_DEPTH_INFO	

33.2.42 Global device TXFIFO DMA priority register (GTXFIFOPRIDEV)

33.2.42.1 Offset

Register	Offset
GTXFIFOPRIDEV	C610h

33.2.42.2 Function

This register specifies the relative DMA priority level among the Device TXFIFOs (one per IN endpoint). Each register bit[n] controls the priority (1: high, 0: low) of each TXFIFO[n]. When multiple TXFIFOs compete for DMA service at a given time (that is, multiple TXQs contain TX DMA requests and their corresponding TXFIFOs have space available), the TX DMA arbiter grants access on a packet-basis in the following manner:

1. High-priority TXFIFOs are granted access using round-robin arbitration
2. Low-priority TXFIFOs are granted access using round-robin arbitration only after the high-priority TXFIFOs have no further processing to do (that is, either the TXQs are empty or the corresponding TXFIFOs are full).

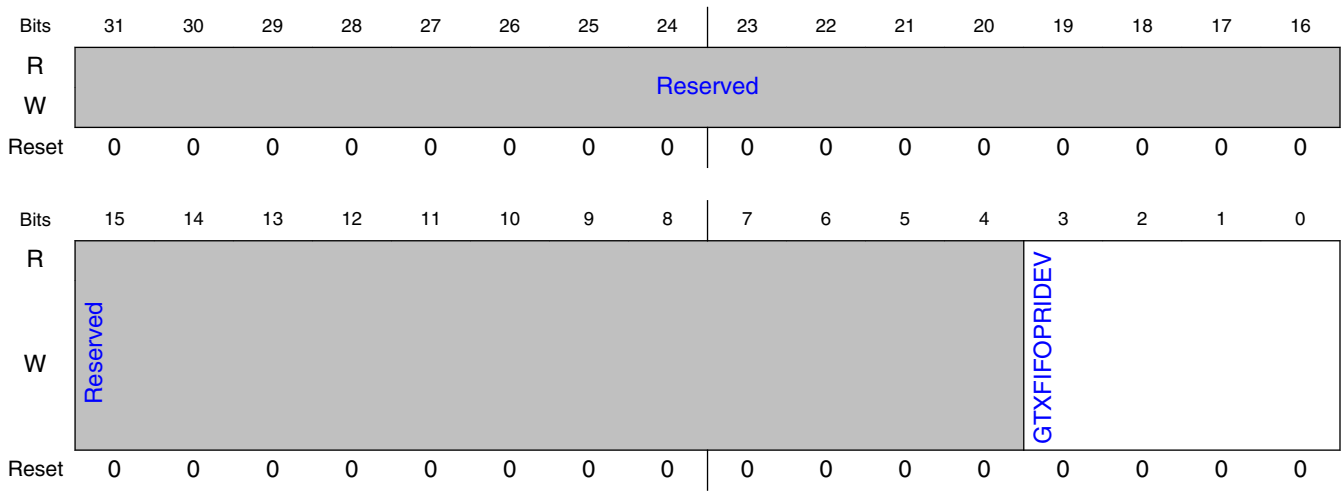
For scatter-gather packets, the arbiter grants successive DMA requests to the same FIFO until the entire packet is completed.

When configuring periodic IN endpoints, software must set register bit[n]=1, where n is the TXFIFO assignment. This ensures that the DMA for isochronous or interrupt IN endpoints are prioritized over bulk or control IN endpoints. The register size corresponds to the number of Device IN endpoints.

NOTE

Since the device mode uses only one RXFIFO, there is no device RXFIFO DMA priority register.

33.2.42.3 Diagram



33.2.42.4 Fields

Field	Function
31-4 —	Reserved
3-0 GTXFIFOPRIDEV	Device TXFIFO priority 0000b - Low (default) 0001b - High

33.2.43 Global host TXFIFO DMA priority register (GTXFIFOPRIHST)

33.2.43.1 Offset

Register	Offset
GTXFIFOPRIHST	C618h

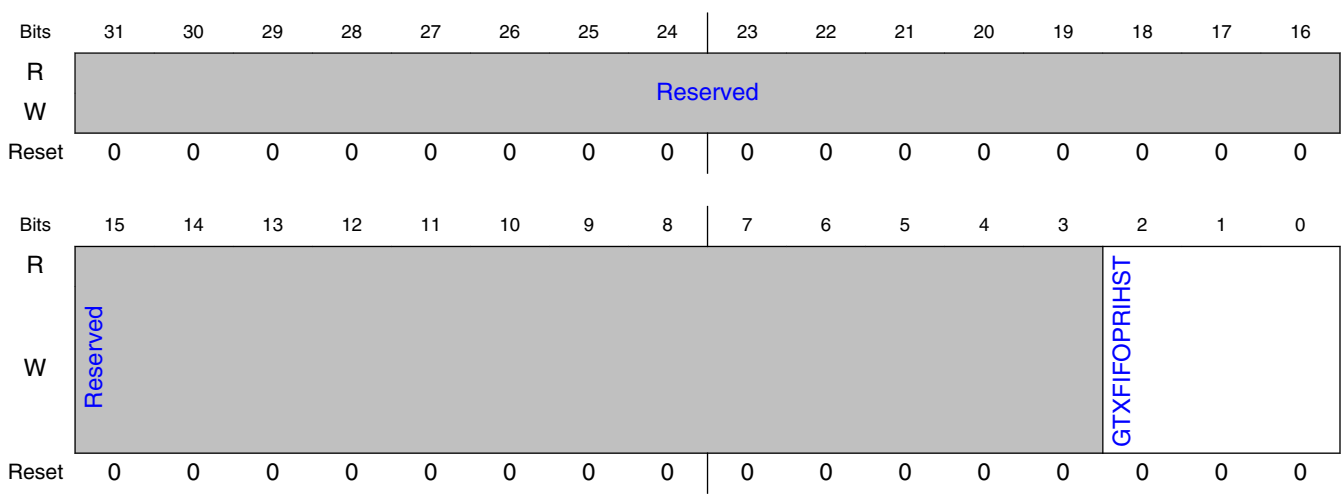
33.2.43.2 Function

This register specifies the relative DMA priority level among the host TXFIFOs (one per USB bus instance) within the associated speed group (SS or HS/FS/LS). Each register bit[n] controls the priority (1: high, 0: low) of TXFIFO[n] within a speed group. When multiple TXFIFOs compete for DMA service at a given time (that is, multiple TXQs contain TX DMA requests and their corresponding TXFIFOs have space available), the TX DMA arbiter grants access on a packet-basis in the following manner:

1. Among the FIFOs in the same speed group (SS or HS/FS/LS):
   a. High-priority TXFIFOs are granted access using round-robin arbitration
   b. Low-priority TXFIFOs are granted access using round-robin arbitration only after the high- priority TXFIFOs have no further processing to do (that is, either the TXQs are empty or the corresponding TXFIFOs are full).
2. The TX DMA arbiter prioritizes the SS speed group or HS/FS/LS speed group according to the ratio programmed in the GDMAHLRATIO register.

For scatter-gather packets, the arbiter grants successive DMA requests to the same FIFO until the entire packet is completed. The register size corresponds to the number of configured USB bus instances; for example, in the default configuration, there are 3 USB bus instances (1 SS, 1 HS, and 1 FS/LS).

33.2.43.3 Diagram



33.2.43.4 Fields

Field	Function
31-3 —	Reserved
2-0 GTXFIFOPRIHST	Host TXFIFO priority 000b - Low priority (default) 001b - High priority

33.2.44 Global host RXFIFO DMA priority register (GRXFIFOPRIHST)

33.2.44.1 Offset

Register	Offset
GRXFIFOPRIHST	C61Ch

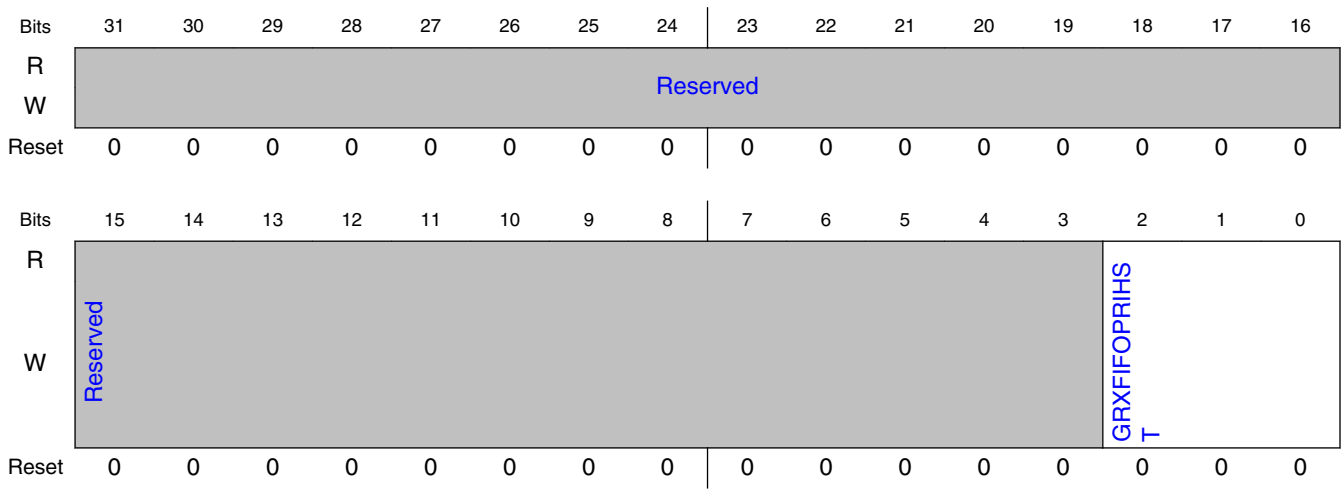
33.2.44.2 Function

This register specifies the relative DMA priority level among the host RXFIFOs (one per USB bus instance) within the associated speed group (SS or HS/FS/LS). Each register bit[n] controls the priority (1: high, 0: low) of RXFIFO[n] within a speed group. When multiple RXFIFOs compete for DMA service at a given time (that is, multiple RXQs contain RX DMA requests and their corresponding RXFIFOs have data available), the RX DMA arbiter grants access on a packet-basis in the following manner:

1. Among the FIFOs in the same speed group (SS or HS/FS/LS):
   a. High-priority RXFIFOs are granted access using round-robin arbitration
   b. Low-priority RXFIFOs are granted access using round-robin arbitration only after high-priority RXFIFOs have no further processing to do (that is, either the RXQs are empty or the corresponding RXFIFOs do not have the required data).
2. The RX DMA arbiter prioritizes the SS speed group or HS/FS/LS speed group according to the ratio programmed in the GDMAHLRATIO register.

For scatter-gather packets, the arbiter grants successive DMA requests to the same FIFO until the entire packet is completed. This register is present only when the core is configured to operate in the host mode (includes DRD and OTG modes). The register size corresponds to the number of configured USB bus instances; for example, in the default configuration, there are 3 USB bus instances (1 SS, 1 HS, and 1 FS/LS).

33.2.44.3 Diagram



33.2.44.4 Fields

Field	Function
31-3 —	Reserved
2-0 GRXFIFOPRIHST	Host RXFIFO priority 000b - Low priority (default) 001b - High priority

33.2.45 Global host FIFO DMA high-low priority ratio register (GDMAHLRATIO)

33.2.45.1 Offset

Register	Offset
GDMAHLRATIO	C624h

33.2.45.2 Function

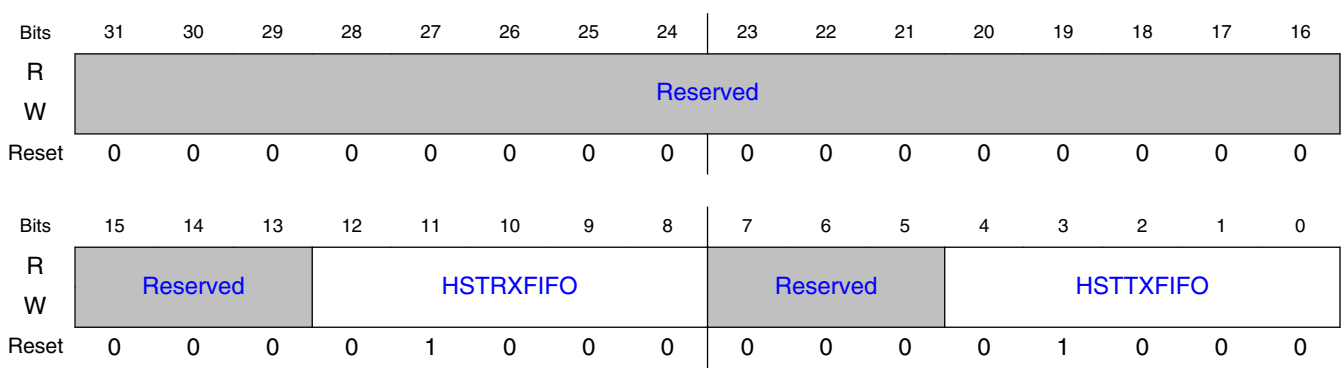
This register specifies the relative priority of the SS FIFOs with respect to the HS/FS/LS FIFOs. The DMA arbiter prioritizes the HS/FS/LS round-robin arbiter group every DMA high-low priority ratio grants as indicated in the register separately for TX and RX.

To illustrate, consider that all FIFOs are requesting access simultaneously, and the ratio is 4. SS gets priority for 4 packets, HS/FS/LS gets priority for 1 packet, SS gets priority for 4 packets, HS/FS/LS gets priority for 1 packet, and so on.

If FIFOs from both speed groups are not requesting access simultaneously then,

- If SS got grants 4 out of the last 4 times, then HS/FS/LS get the priority on any future request.
- If HS/FS/LS got the grant last time, SS gets the priority on the next request.
- If there is a valid request on either SS or HS/FS/LS, a grant is always awarded; there is no idle. This register is present if the core is configured to operate in host mode.

33.2.45.3 Diagram



33.2.45.4 Fields

Field	Function
31-13 —	Reserved
12-8 HSTRXFIFO	Host RXFIFO DMA high-low priority ratio
7-5 —	Reserved
4-0 HSTTXFIFO	Host TXFIFO DMA high-low priority ratio

33.2.46 Global frame length adjustment register (GFLADJ)

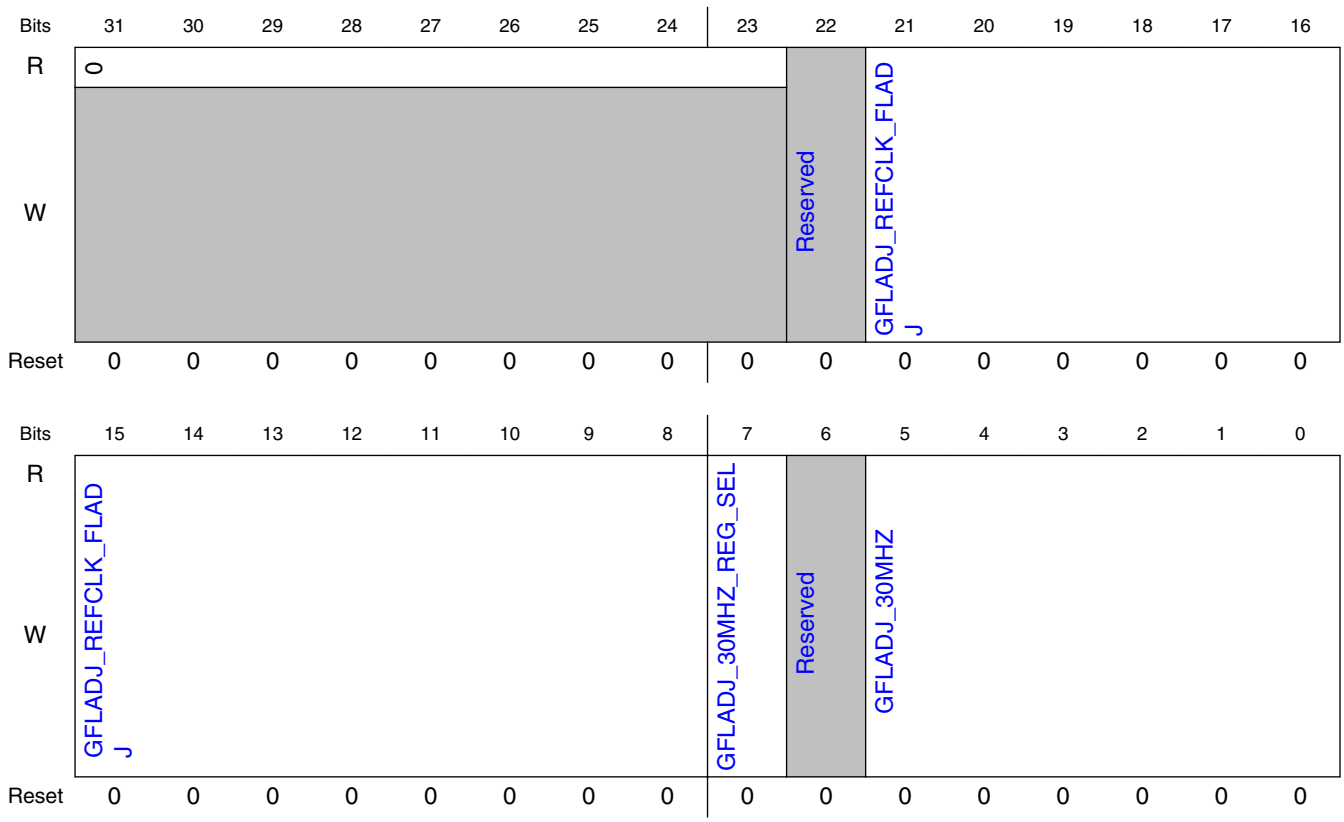
33.2.46.1 Offset

Register	Offset
GFLADJ	C630h

33.2.46.2 Function

It provides an option to override the fladj_30mhz_reg sideband signal.

33.2.46.3 Diagram



33.2.46.4 Fields

Field	Function
31-23 —	Reserved This bit should always read 0.
22 —	Reserved
21-8 GFLADJ_REFCLK_FLADJ	This bit indicates the frame length adjustment to be applied when SOF/ITP counter is running on the ref_clk.
7 GFLADJ_30MHZ_REG_SEL	This bit selects whether to use a hard-coded value of 20h (32 decimal) or the value in GFLADJ[GFLADJ_30MHZ] to adjust the frame length for the SOF/ITP. 0b - The controller uses the hard coded value 20h (32 decimal) . which gives a SOF cycle time of 60000. 1b - The controller uses the value in GFLADJ[GFLADJ_30MHZ].
6 —	Reserved

Table continues on the next page...

Field	Function
5-0 GFLADJ_30MHZ	This bit indicates the value that is used for frame length adjustment when GFLADJ_30MHZ_REG_SEL = 1. Each step of this field's value corresponds to 16 high-speed bit times. The SOF cycle time (the number of SOF counter clock periods to generate a SOF microframe length) is equal to 59488 + the value in this field. When GFLADJ_30MHZ_REG_SEL = 0, a hard-coded value of 20h (32 decimal) is used, which gives a SOF cycle time of 60000. For details on how to set this value, refer to section 5.2.4, "Frame Length Adjustment Register (FLADJ)," of the xHCI Specification. 000000b - Frame length is 59488 HS bit times 000001b - Frame length is 59504 HS bit times 000010b - Frame length is 59520 HS bit times ... 011111b - Frame length is 59984 HS bit times 100000b - Frame length is 60000 HS bit times ... 111110b - Frame length is 60480 HS bit times 111111b - Frame length is 60496 HS bit times

33.2.47 Device configuration register (DCFG)

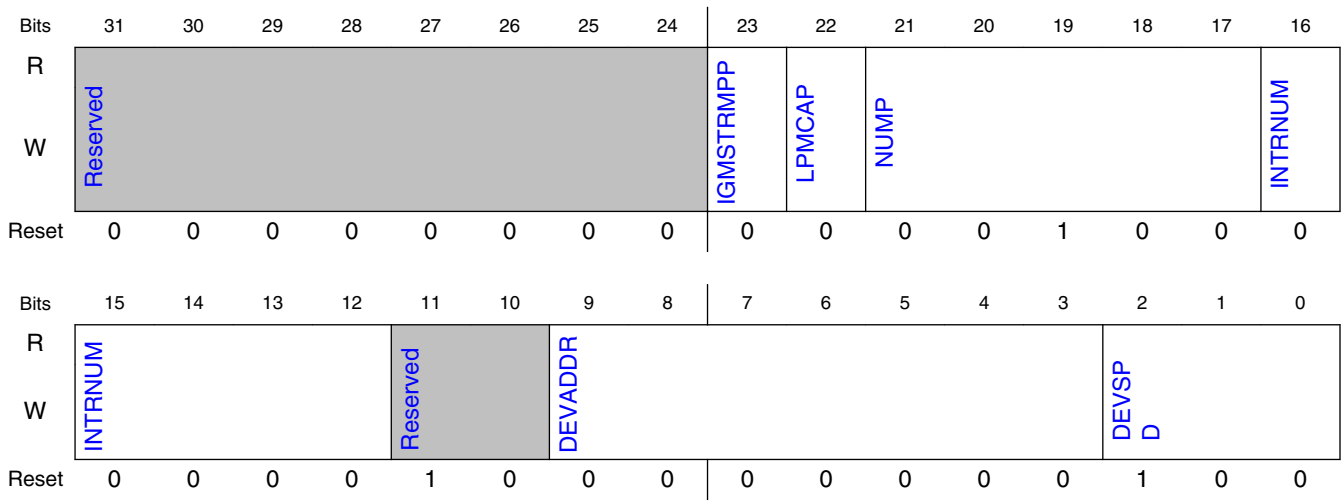
33.2.47.1 Offset

Register	Offset
DCFG	C700h

33.2.47.2 Function

This register configures the core in device mode after power-on or after certain control commands or enumeration. Do not make changes to this register after initial programming.

33.2.47.3 Diagram



33.2.47.4 Fields

Field	Function
31-24 —	Reserved
23 IGMSTRMPP	Ignore stream PP This bit only affects stream-capable bulk endpoints. When this bit is set to 0 and the controller receives a data packet with the Packet Pending (PP) bit set to 0 for OUT endpoints, or it receives an ACK with the NUMP bit set to 0 and PP set to 0 for IN endpoints, the core attempts to search for another stream (CStream) to initiate to the host. However, there are two situations where this behavior is not optimal: • When the host is setting PP=0 even though it has not finished the stream, or • When the endpoint on the device is configured with one transfer resource and therefore does not have any other streams to initiate to the host. When this bit is set to 1, the core ignores the Packet Pending bit for the purposes of stream selection and does not search for another stream when it receives DP(PP=0) or ACK(NUMP=0, PP=0). This can enhance the performance when the device system bus bandwidth is low or the host responds to the core's ERDY transmission very quickly.
22 LPMCAP	LPM capable The application uses this bit to control the USB 3.0 core LPM capabilities. If the core operates as a non-LPM-capable device, it cannot respond to LPM transactions. 0b - LPM capability is not enabled 1b - LPM capability is enabled
21-17 NUMP	Number of receive buffers This bit indicates the number of receive buffers to be reported in the ACK TP.

Table continues on the next page...

Field	Function
	The USB 3.0 controller uses this bit if GRXTHRCFG[USBRXPKTCNTSEL] is set to 0. The application can program this value based on RXFIFO size, buffer sizes programmed in descriptors, and system latency. For an OUT endpoint, this bit controls the number of receive buffers reported in the NUMP bit of the ACK TP transmitted by the core. **NOTE:** This bit is used in host mode when debug capability is enabled.
16-12 INTRNUM	Interrupt number Indicates interrupt/EventQ number on which non-endpoint-specific device-related interrupts (see DEVT) are generated.
11-10 —	Reserved
9-3 DEVADDR	Device address The application must perform the following: • Program this bit after every SetAddress request. • Reset this bit to 0 after USB reset.
2-0 DEVSPD	Device speed Indicates the speed at which the application requires the core to connect, or the maximum speed the application can support. However, the actual bus speed is determined only after the chirp sequence is completed, and is based on the speed of the USB host to which the core is connected. 000b - High-speed (USB 2.0 PHY clock is 30 MHz or 60 MHz) 001b - Full-speed (USB 2.0 PHY clock is 30 MHz or 60 MHz) 100b - SuperSpeed (USB 3.0 PHY clock is 125 MHz or 250 MHz)

33.2.48 Device control register (DCTL)

33.2.48.1 Offset

Register	Offset
DCTL	C704h

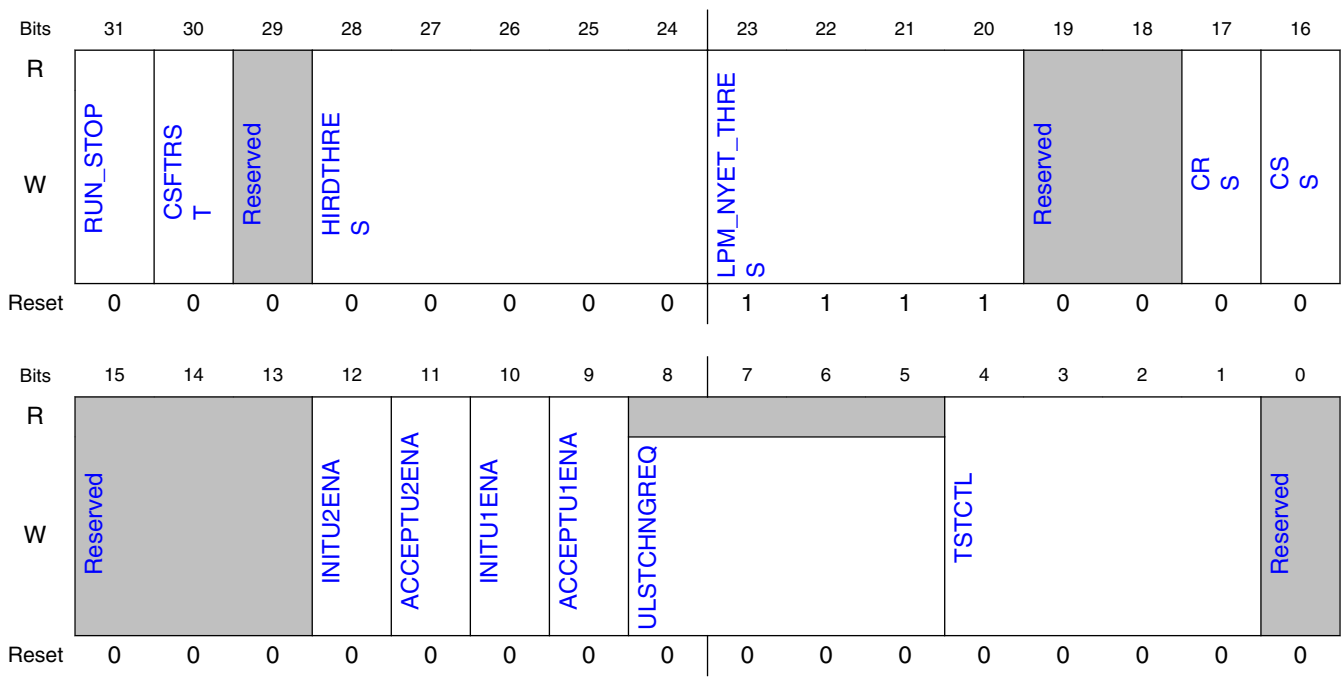
33.2.48.2 Function

The table below lists the minimum duration under various conditions for which the soft disconnect (SFTDISCON) bit must be set for the USB host to detect a device disconnect. To accommodate clock jitter, it is recommended that the application add extra delay to the specified minimum duration.

Table 33-6. Minimum duration for soft disconnect

Operating speed	Device state	Minimum duration
SuperSpeed	Suspended	30 ms
SuperSpeed	Idle	30 ms
SuperSpeed	Transmit or Receive	30 ms
High-speed	Suspended	10 ms
High-speed	Idle	10 ms
High-speed	Not Idle or Suspended (Performing transactions)	10 ms
Full-speed/Low-speed	Suspended	10 ms
Full-speed/Low-speed	Idle	10 ms
Full-speed/Low-speed	Not idle or suspended (performing transactions)	10 ms

33.2.48.3 Diagram



33.2.48.4 Fields

Field	Function
31	The software writes 1 to this bit to start the device controller operation.

Table continues on the next page...

Field	Function
RUN_STOP	To stop the device controller operation, the software must remove any active transfers and write 0 to this bit. When the controller is stopped, it sets the DSTS[DEVCTRLHLT] bit when the core is idle and the lower layer finishes the disconnect process. The RUN_STOP bit must be used in following cases as specified: 1. After power-on-reset and CSR initialization, the software must write 1 to this bit to start the device controller. The controller does not signal connect to the host until this bit is set. 2. The software uses this bit to control the device controller to perform a soft disconnect. When the software writes 0 to this bit, the host does not see that the device is connected. The device controller stays in the disconnected state until the software writes 1 to this bit. 3. If the software attempts a connect after the soft disconnect or detects a disconnect event, it must set DCTL[ULSTCHNGREQ] to 5 before reasserting the RUN_STOP bit. The minimum duration of keeping this bit cleared is specified in Table 33-6. 4. When the USB or Link is in a lower power state and the two power rails configuration is selected, software writes 0 to this bit to indicate that it is going to turn off the core power rail. After the software turns on the core power rail again and re-initializes the device controller, it must set this bit to start the device controller. For more details, see Low power operation
30 CSFTRST	Core soft reset Resets all clock domains as follows: • Clears the interrupts and all the CSRs except the following registers: GCTL, GUCTL, GSTS, GUID, GUSB2PHYCFG, GUSB3PIPECTL, DCFG, DCTL, DEVTEN, DSTS • All module state machines (except the SoC bus slave unit) are reset to the IDLE state, and all the TXFIFOs and the RXFIFO are flushed. • Any transactions on the SoC bus master are terminated as soon as possible, after gracefully completing the last data phase of a SoC bus transfer. Any transactions on the USB are terminated immediately. The application can write this bit at any time to reset the core. This is a self- clearing bit; the core clears this bit after all necessary logic is reset in the core, which may take several clocks depending on the core's current state. Once this bit is cleared, the software must wait at least 3 PHY clocks before accessing the PHY domain (synchronization delay). Typically, software reset is used during software development and also when the PHY selection bits are dynamically changed in the USB configuration registers listed above. When the PHY is changed, the corresponding clock for the PHY is selected and used in the PHY domain. Once a new clock is selected, the PHY domain must be reset for proper operation.
29 —	Reserved

28-24
HIRDTHRES
	HIRD threshold
	The core asserts output signals utmi_l1_suspend_n and utmi_sleep_n on the basis of this signal.
	• The core asserts utmi_l1_suspend_n to put the PHY into deep low-power mode in L1 when both of the following are true:
	  • HIRD value is greater than or equal to the value in DCTL[HIRD_THRES][3-0]
	  • HIRD_THRES[4] is set to 1'b1.
	• The core asserts utmi_sleep_n on L1 when one of the following is true:
	  • If the HIRD value is less than HIRD_Thres[3-0]
	  • HIRD_Thres[4] is set to 1'b0.
	NOTE: This bit must be set to 0 during SuperSpeed mode of operation.

23-20 LPM_NYET_THRES	When LPM errata is enabled: LPM NYET response threshold handshake response to LPM token specified by device application. Response depends on DCFG[LPMCAP].

Table continues on the next page...

Field	Function

	• DCFG[LPMCAP] is 1'b0 : The core always responds with timeout (that is, no response).
	• DCFG[LPMCAP] is 1'b1 : The core responds with an ACK on successful LPM transaction, which requires that all of the following are satisfied:
	  • There are no PID or CRC5 errors in both the EXT token and the LPM token (if not true, inactivity results in a timeout ERROR)
	  • A valid bLinkState = 0001B (L1) is received in the LPM transaction (else STALL) .
	  • No data is pending in the Transmit FIFO and OUT endpoints not in flow controlled state (else NYET).
	  • The BESL value in the LPM token is less than or equal to LPM_NYET_THRES[3-0]

19-18 —	Reserved **NOTE:** Software should program bit 19 to 0.
17 CRS	Controller restore state This command is similar to the USBCMD[CRS] bit in host mode and initiates the restore process. When software sets this bit to 1, the controller immediately sets DSTS[RSS] to 1. When the controller has finished the restore process, it sets DSTS[RSS] to 0. **NOTE:** When read, this bit always returns 0.
16 CSS	Controller save state This command is similar to the USBCMD[CSS] bit in host mode and initiates the save process. When software sets this bit to 1, the controller immediately sets DSTS[SSS] to 1. When the controller has finished the save process, it sets DSTS[SSS] to 0. **NOTE:** When read, this bit always returns 0.
15-13 —	Reserved
12 INITU2ENA	Initiate U2 enable On USB reset, hardware clears this bit to 0. Software sets this bit after receiving SetFeature(U2_ENABLE), and clears this bit when ClearFeature(U2_ENABLE) is received. If DCTL[ACCEPTU2ENA] is 0, the link immediately exits U2 state. 0b - May not initiate U2 (default) 1b - May initiate U2
11 ACCEPTU2ENA	Accept U2 enable On USB reset, hardware clears this bit to 0. Software sets this bit after receiving a SetConfiguration command. 0b - Reject U2 except when Force_LinkPM_Accept bit is set (default) 1b - Core accepts transition to U2 state if nothing is pending on the application side
10 INITU1ENA	Initiate U1 enable On USB reset, hardware clears this bit to 0. Software sets this bit after receiving SetFeature(U1_ENABLE), and clears this bit when ClearFeature(U1_ENABLE) is received. If DCTL[ACCEPTU1ENA] is 0, the link immediately exits U1 state. 0b - May not initiate U1 (default) 1b - May initiate U1
9 ACCEPTU1ENA	Accept U1 enable On USB reset, hardware clears this bit to 0. Software sets this bit after receiving a SetConfiguration command. 0b - Core rejects U1 except when Force_LinkPM_Accept bit is set (default) 1b - Core accepts transition to U1 state if nothing is pending on the application side
8-5	USB/Link state change request

Table continues on the next page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td>ULSTCHNGREQ</td><td>Software writes this bit to issue a USB/Link state change request. A change in this bit indicates a new request to the core. If software wants to issue the same request back-to-back, it must write a 0 to this bit between the two requests. The result of the state change request is reflected in the USB/Link state in DSTS. These bits are self-cleared on the MAC Layer exiting suspended state.

If software is updating other bits of the DCTL register and not intending to force any link state change, then it must write a 0 to this bit.

SS compliance mode is normally entered and controlled by the remote link partner. Refer to the USB3 specification. Alternatively, the local link can be directly forced into compliance mode, by resetting the SS link with the RUN_STOP bit set to 0. If you then write 10 to the USB/Link state change bit and 1 to RUN_STOP, the Link goes to compliance. Once you are in compliance, you may alternately write 0 and 10 to this bit to advance the compliance pattern.

In SS mode:

<table>
<tr><th>Value</th><th>Requested link state transition/action</th></tr>
<tr><td>0</td><td>No action</td></tr>
<tr><td>4</td><td>SS.Disabled</td></tr>
<tr><td>5</td><td>Rx.Detect</td></tr>
<tr><td>6</td><td>SS.Inactive</td></tr>
<tr><td>8</td><td>Recovery</td></tr>
<tr><td>Others</td><td>Reserved</td></tr>
</table>

In HS/FS/LS mode:

<table>
<tr><th>Value</th><th>Requested USB state transition</th></tr>
<tr><td>8</td><td>Remote wakeup request</td></tr>
<tr><td>Others</td><td>Reserved</td></tr>
</table>

The Remote wakeup request should be issued 2 µs after the device goes into suspend state (DSTS[USBLNKST] is 3 (refer to Device status register (DSTS)).</td></tr>
<tr><td>4-1

TSTCTL</td><td>Test control

The settings not defined are reserved.

0000b - Test mode disabled
0001b - Test_J mode
0010b - Test_K mode
0011b - Test_SE0_NAK mode
0100b - Test_Packet mode
0101b - Test_Force_Enable</td></tr>
<tr><td>0

—</td><td>Reserved</td></tr>
</table>

33.2.49 Device event enable register (DEVTEN)

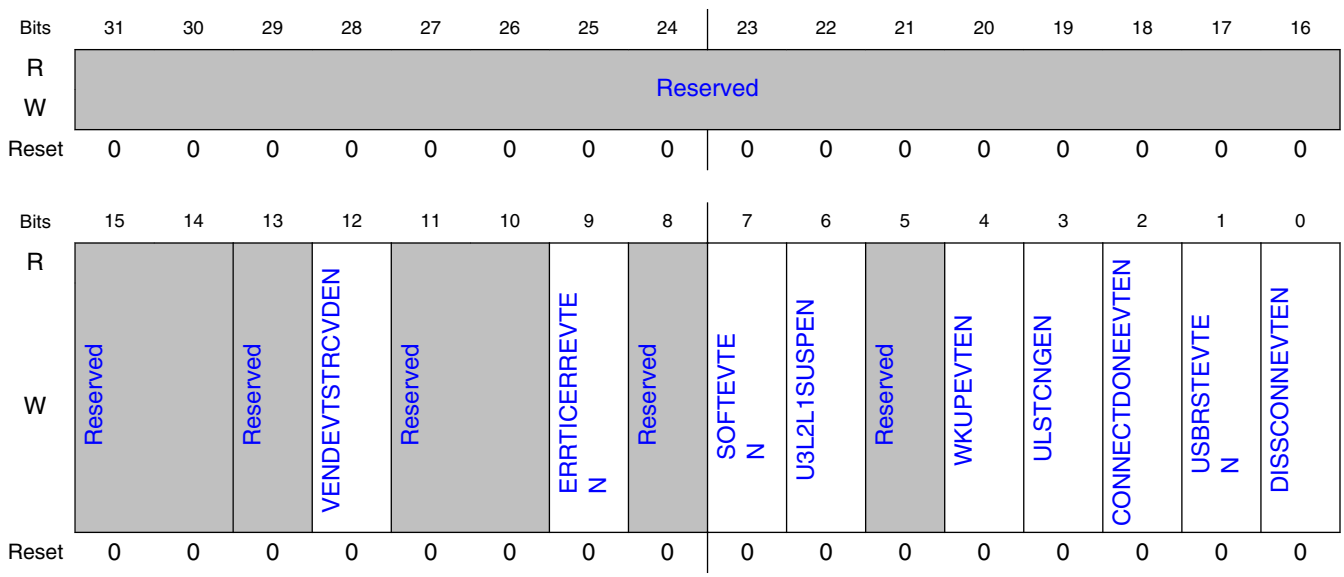
33.2.49.1 Offset

Register	Offset
DEVTEN	C708h

33.2.49.2 Function

This register controls the generation of device-specific events. If an enable bit is set to 0, the event is not generated.

33.2.49.3 Diagram



33.2.49.4 Fields

Field	Function
31-14 —	Reserved
13 —	Reserved
12	Vendor device test LMP received rvent

Table continues on the next page...

Field	Function
VENDEVTSTRCVDEN	
11-10 —	Reserved
9 ERRTICERREVTEN	Erratic error event enable
8 —	Reserved
7 SOFTEVTEN	Start of (micro)frame enable
6 U3L2L1SUSPEN	U3/L2-L1 suspend event enable
5 —	Reserved
4 WKUPEVTEN	Resume/Remote wakeup detected event enable
3 ULSTCNGEN	USB/Link state change event enable
2 CONNECTDONEEVTEN	Connection done enable
1 USBRSTEVTEN	USB reset enable
0 DISSCONNEVTEN	Disconnect detected event enable

33.2.50 Device status register (DSTS)

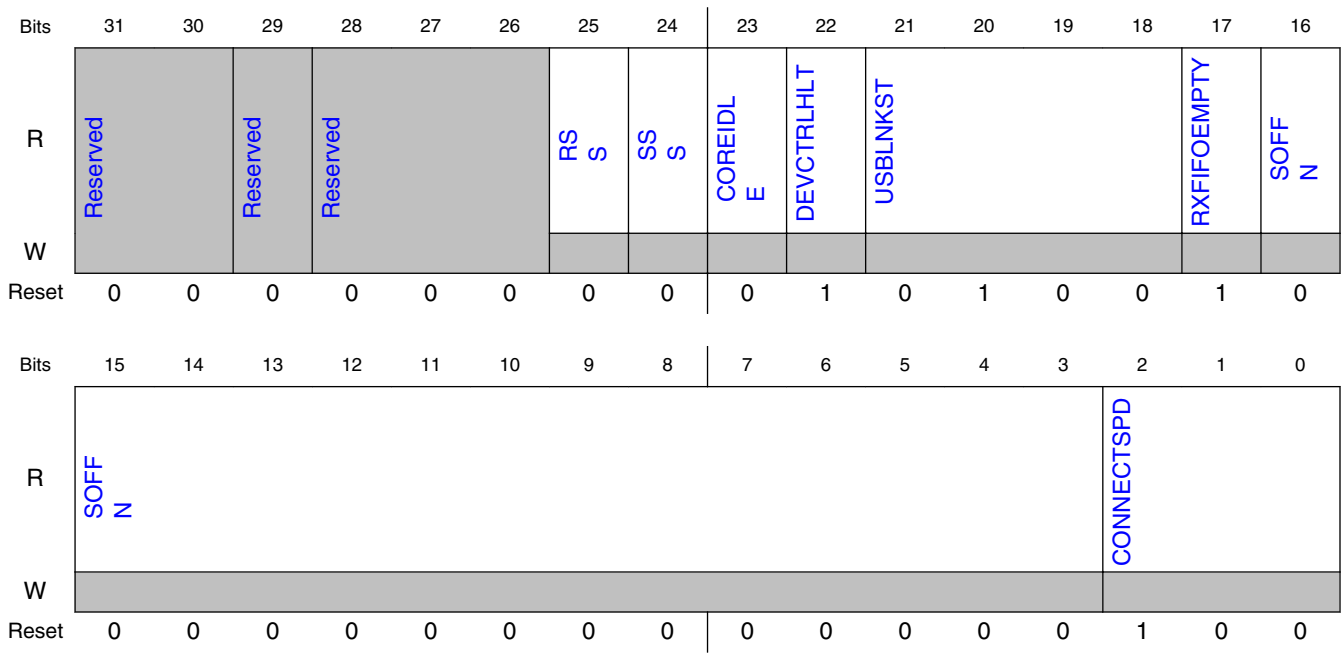
33.2.50.1 Offset

Register	Offset
DSTS	C70Ch

33.2.50.2 Function

This register indicates the status of the device controller with respect to USB-related events.

33.2.50.3 Diagram



33.2.50.4 Fields

Field	Function
31-30 —	Reserved
29 —	Reserved
28-26 —	Reserved
25 RSS	Restore state status This bit is similar to the USBSTS[RSS] in host mode. When the controller has finished the restore process, it completes the command by setting DSTS[RSS] to 0.
24	Save state status

Table continues on the next page...

Field	Function
SSS	This bit is similar to the USBSTS[SSS] in host mode. When the controller has finished the save process, it completes the command by setting DSTS[SSS] to 0.
23 COREIDLE	Core idle The bit indicates that the core finished transferring all RXFIFO data to system memory, writing out all completed descriptors, and all event counts are zero. **NOTE:** While testing for reset values, mask out the read value. This bit represents the changing state of the core and does not hold a static value.
22 DEVCTRLHLT	Device controller halted This bit is set to 0 when the DCTL[RUN_STOP] bit is set to 1. The core sets this bit to 1 when, after software sets RUN_STOP to 0, the core is idle and the lower layer finishes the disconnect process. When Halted =1, the core does not generate device events.
21-18 USBLNKST	USB/Link state In SS mode: LTSSM State 4'h0 U0 4'h1 U1 4'h2 U2 4'h3 U3 4'h4 SS_DIS 4'h5 RX_DET 4'h6 SS_INACT 4'h7 POLL 4'h8 RECOV 4'h9 HRESET 4'hA CMPLY 4'hB LPBK 4'hF Resume/Reset In HS/FS/LS mode. 4'h0 On state 4'h2 Sleep (L1) state 4'h3 Suspend (L2) state 4'h4 Disconnected state (default) Software must write 8 (Recovery) to the DCTL[ULSTCHNGREQ] bit to acknowledge the resume/reset request.
17 RXFIFOEMPTY	RXFIFO empty
16-3 SOFFN	Frame/microframe number of the received SOF When the core is operating at high-speed, [16:6] indicates the frame number [5:3] indicates the microframe number

Table continues on the next page...

Field	Function
	When the core is operating at high-speed, [16:14] is not used. Software can ignore these 3 bits [13:3] indicates the frame number
2-0 CONNECTSPD	Connected speed Indicates the speed at which the USB 3.0 core has come up after speed detection through a chirp sequence. Low-speed is not supported for devices using a UTMI+ PHY. 000b - High-speed (PHY clock is running at 30 MHz or 60 MHz) 001b - Full-speed (PHY clock is running at 30 MHz or 60 MHz) 010b - Low-speed (PHY clock is running at 6 MHz) 011b - Full-speed (PHY clock is running at 48 MHz) 100b - SuperSpeed (PHY clock is running at 125 MHz or 250 MHz)

33.2.51 Device generic command parameter register (DGCMDPAR)

33.2.51.1 Offset

Register	Offset
DGCMDPAR	C710h

33.2.51.2 Function

This register indicates the device command parameter. This must be programmed before or along with the device command. The available device commands are listed in Device generic command register (DGCMD).

33.2.51.3 Diagram

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

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

33.2.51.4 Fields

Field	Function
31-0 PARAMETER	Parameter for device command

33.2.52 Device generic command register (DGCMD)

33.2.52.1 Offset

Register	Offset
DGCMD	C714h

33.2.52.2 Function

This register enables software to program the core using a single generic command interface to send link management packets and notifications. This register contains command, control, and status bits relevant to the current generic command, while the DGCMDPAR register provides the command parameter.

Any commands that are not present in the following table are considered Reserved.

Table 33-7. Device generic command types

<table>
<tr><th>Command</th><th>Description</th></tr>
<tr><td>02h</td><td>Set periodic parameters

Parameter[9:0] (SystemExitLatency): Software should set this to the same value programmed by the host through the Set SEL device request, in microseconds.

The Set SEL control transfer has 6 bytes of data and contains 4 values.

<table>
<tr><th>Offset</th><th>Name</th><th>Meaning</th></tr>
<tr><td>0</td><td>U1SEL</td><td>Time in µs for U1 system exit latency</td></tr>
<tr><td>1</td><td>U1PEL</td><td>Time in µs for U1 device to host exit latency</td></tr>
<tr><td>2</td><td>U2SEL</td><td>Time in µs for U2 system exit latency</td></tr>
<tr><td>4</td><td>U2PEL</td><td>Time in µs for U2 device to host exit latency</td></tr>
</table>

If the device is enabled for U1 and U2, then the U2PEL should be programmed. If the device is enabled only for U1, then U1PEL should be programmed into this parameter.

If the value is greater than 125 µs, then the software must program a value of 0 into this register.</td></tr>
<tr><td>04h</td><td>Set scratchpad buffer array address low

This command sets bits [31:0] of the external address of the scratchpad buffer array used for save/restore.

If either this command or command 05h is issued while the controller is stopped (RUN_STOP = 0), the CMDIOC bit must be set to 0.

The device scratchpad buffer array has the same format as the xHCI scratchpad buffer array; it contains an array of 64-bit pointers to data buffers that is used to save the controller's state.</td></tr>
<tr><td>05h</td><td>Set scratchpad buffer array address high

This command sets bits [63:32] of the external address of the scratchpad array buffer used for save/restore.

If either this command or command 04h is issued while the controller is stopped (RUN_STOP = 0), the CMDIOC bit must be set to 0.

The device scratchpad buffer array has the same format as the xHCI scratchpad buffer array; it contains an array of 64-bit pointers to data buffers that is used to save the controller's state.</td></tr>
<tr><td>07h</td><td>Transmit device notification

This command allows any device notification to be transmitted, using the notification type and notification parameters specified in the DGCMDPAR register.

DGCMDPAR[3:0] = Notification type

DGCMDPAR[31:4] = Notification parameters, depends on the notification type

For example, to transmit a function wake, software sets DGCMDPAR[3:0] to 1, and DGCMDPAR[10:4] to the interface number.

This bit relates to the "Notification Type Specific" bit in a device notification transaction packet as described in Section 8.5.6 of the USB3 Specification. The following bits of the DGCMDPAR register have been put into the corresponding DWORD described in Section 8.5.6 of the USB3 Specification:

DGCMDPAR[3:0] into DWORD1[7:4] (Notification Type)

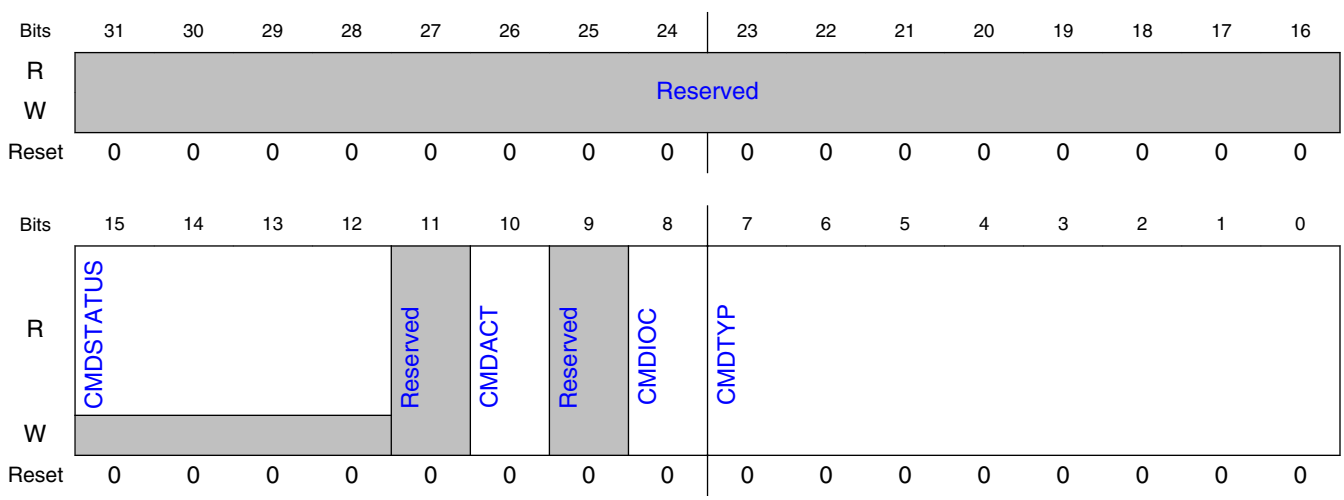
DGCMDPAR[27:4] into DWORD1[31:8] (Notification Type Specific)</td></tr>
</table>

Table continues on the next page...

Table 33-7. Device generic command types (continued)

Command	Description
	DGCMDPAR[31:28] into DWORD2[3:0] (Notification Type Specific) There is one exception for the bus interval adjustment device notification: DGCMDPAR[19:4] represents the bus interval adjustment bit; however, in the USB3 specification, the bus interval adjustment bit is actually at 31:16 of DWORD1.
09h	Selected FIFO flush Parameter[4:0] = FIFO number Parameter[5] = 1 for TXFIFO or 0 for RXFIFO
0Ah	All FIFO flush No parameter
0Ch	Set endpoint NRDY Issuing this command makes the core think that the given endpoint is in an NRDY state. If there are buffers available in that endpoint, the core immediately transmits an ERDY. Parameter[4:0] = Physical endpoint number

33.2.52.3 Diagram



33.2.52.4 Fields

Field	Function
31-16 —	Reserved
15-12 CMDSTATUS	Command status 0000b - Indicates command success

Table continues on the next page...

Field	Function
	0001b - CmdErr, indicates that the device controller encountered an error while processing the command
11 —	Reserved.
10 CMDACT	Command active The software sets this bit to 1 to enable the device controller to execute the generic command. The device controller sets this bit to 0 after executing the command.
9 —	Reserved.
8 CMDIOC	Command interrupt on complete When this bit is set, the device controller issues a generic command completion event after executing the command. Note that this interrupt is mapped to DCFG[INTRNUM]. **NOTE:** This bit must not set to 1 if the DCTL[RUN_STOP] bit is 0.
7-0 CMDTYP	Command type Specifies the type of command the software driver is requesting the core to perform. 8'h0 Reserved 8'h1 Set endpoint configuration - 64-bit or 96-bit parameter 8'h2 Set endpoint transfer resource configuration - 32-bit parameter 8'h3 Get endpoint state - no parameter needed 8'h5 Clear stall (see set stall) - no parameter needed 8'h6 Start transfer - 64-bit parameter 8'h7 Update transfer - no parameter needed 8'h8 End transfer - no parameter needed 8'h9 Start new configuration - no parameter needed

33.2.53 Device active USB endpoint enable register (DALEPENA)

33.2.53.1 Offset

Register	Offset
DALEPENA	C720h

33.2.53.2 Function

This register indicates whether a USB endpoint is active in a given configuration or interface.

33.2.53.3 Diagram

Bits	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

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								USBACTEP							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

33.2.53.4 Fields

Field	Function
31-8 —	Reserved
7-0 USBACTEP	USB active endpoints This bit indicates if a USB endpoint is active in the current configuration and interface. Bit[0]: USB EP0-OUT Bit[1]: USB EP0-IN Bit[2]: USB EP1-OUT Bit[3]: USB EP1-IN Bit[4]: USB EP2-OUT Bit[5]: USB EP2-IN Bit[6]: USB EP3-OUT Bit[7]: USB EP3-IN The entity programming this register must set bits 0 and 1 because they enable control endpoints that map to physical endpoints (resources) after USB reset. Hardware clears these bits for all endpoints (other than EP0-OUT and EP0-IN) after detecting a USB reset event. After receiving SetConfiguration and SetInterface requests, the application must program endpoint registers accordingly and set these bits.

33.2.54 Device physical endpoint-n command parameter 2 register (DEPCMDPAR2_0 - DEPCMDPAR2_7)

33.2.54.1 Offset

Register	Offset
DEPCMDPAR2_0	C800h
DEPCMDPAR2_1	C810h
DEPCMDPAR2_2	C820h
DEPCMDPAR2_3	C830h
DEPCMDPAR2_4	C840h
DEPCMDPAR2_5	C850h
DEPCMDPAR2_6	C860h
DEPCMDPAR2_7	C870h

33.2.54.2 Function

This register indicates the physical endpoint command parameter 2. It must be programmed before issuing the command.

33.2.54.3 Diagram

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

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

33.2.54.4 Fields

Field	Function
31-0 PARAMETER	Parameter 2

33.2.55 Device physical endpoint-n command parameter 1 register (DEPCMDPAR1_0 - DEPCMDPAR1_7)

33.2.55.1 Offset

Register	Offset
DEPCMDPAR1_0	C804h
DEPCMDPAR1_1	C814h
DEPCMDPAR1_2	C824h
DEPCMDPAR1_3	C834h
DEPCMDPAR1_4	C844h
DEPCMDPAR1_5	C854h
DEPCMDPAR1_6	C864h
DEPCMDPAR1_7	C874h

33.2.55.2 Function

This register indicates the physical endpoint command parameter 1. It must be programmed before issuing the command.

33.2.55.3 Diagram

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

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

33.2.55.4 Fields

Field	Function
31-0 PARAMETER	Parameter 1

33.2.56 Device physical endpoint-n command parameter 0 register (DEPCMDPAR0_0 - DEPCMDPAR0_7)

33.2.56.1 Offset

Register	Offset
DEPCMDPAR0_0	C808h
DEPCMDPAR0_1	C818h
DEPCMDPAR0_2	C828h
DEPCMDPAR0_3	C838h
DEPCMDPAR0_4	C848h
DEPCMDPAR0_5	C858h
DEPCMDPAR0_6	C868h
DEPCMDPAR0_7	C878h

33.2.56.2 Function

This register indicates the physical endpoint command parameter 0. This must be programmed before or along with the command. For commands needing only one 32-bit parameter, this register must be programmed with the command register.

33.2.56.3 Diagram

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

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

33.2.56.4 Fields

Field	Function
31-0 PARAMETER	Parameter 0

33.2.57 Device physical endpoint-n command register (DEPCMD_0 - DEPCMD_7)

33.2.57.1 Offset

Register	Offset
DEPCMD_0	C80Ch
DEPCMD_1	C81Ch
DEPCMD_2	C82Ch
DEPCMD_3	C83Ch

Table continues on the next page...

Register	Offset
DEPCMD_4	C84Ch
DEPCMD_5	C85Ch
DEPCMD_6	C86Ch
DEPCMD_7	C87Ch

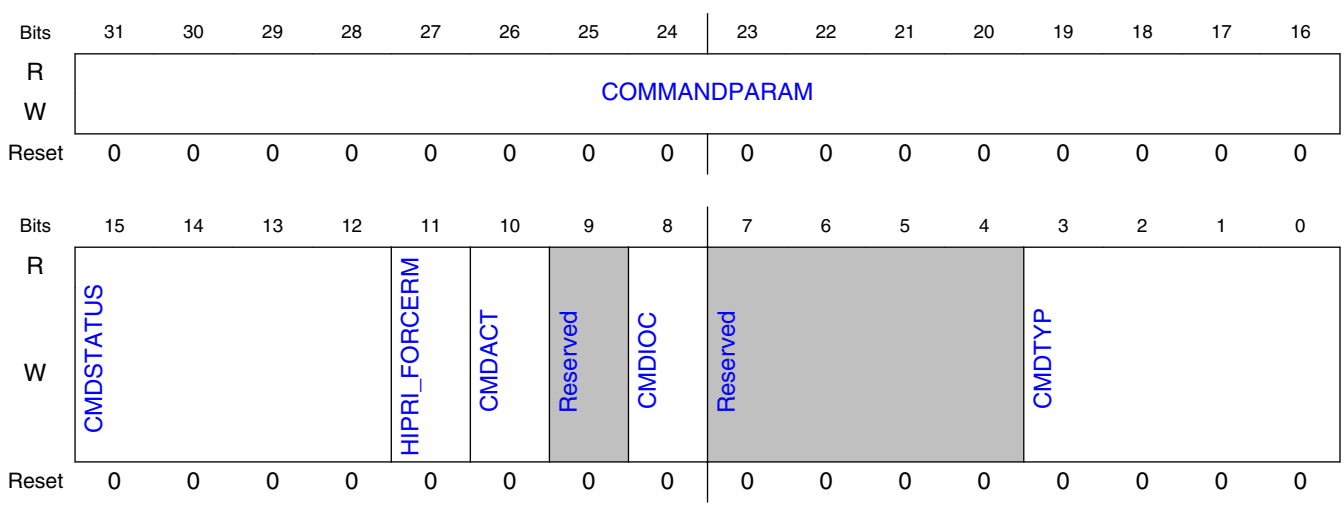
33.2.57.2 Function

This register enables software to issue physical endpoint-specific commands. This register contains command, control, and status bits relevant to the current generic command, while the DEPCMDPAR[2:0]n registers provide command parameters and return status information.

Several bits (including command type) are write-only, so their read values are undefined. After power-on, prior to issuing the first endpoint command, the read value of this register is undefined. In particular, the DGCMD[CMDACT] bit may be set after power-on. In this case, it is safe to issue an endpoint command.

For more details about the commands, refer Device physical endpoint-specific commands.

33.2.57.3 Diagram



33.2.57.4 Fields

Field	Function

31-16
COMMANDPARAM
	Command parameters. When this register is written for:
	• Start transfer command:
	  • [31-16] StreamID. The USB StreamID assigned to this transfer
	• Start transfer command applied to an isochronous endpoint:
	  • [31-16] StartMicroFramNum Indicates the (micro) frame number to which the first TRB applies
	• For update transfer, end transfer, and start new configuration commands:
	  • [22-16] Transfer resource index (XferRscIdx). The hardware- assigned transfer resource index for the transfer, which was returned in response to the start transfer command. The application software-assigned transfer resource index for a start new configuration command.
	
	Event parameters (EventParam). When this register is read, refer to bits 31:16 in Table 33-16

15-12 CMDSTATUS	Command completion status Additional information about the completion of this command is available in this bit. The information is in the same format as bits 15-12 of the endpoint command complete event. Refer Table 33-16.
11 HIPRI_FORCERM	HighPriority/ForceRM HighPriority: Only valid for start transfer command ForceRM: Only valid for end transfer command ClearPendIN: Only valid for clear stall command; software sets this bit to clear any pending IN transaction (on that endpoint) stuck at the lower layers when a clear stall command is issued.
10 CMDACT	Command active Software sets this bit to 1 to enable the device endpoint controller to execute the generic command. The device controller sets this bit to 0 when the CMDSTATUS bit is valid and the endpoint is ready to accept another command. This does not imply that all the effects of the previously-issued command have taken place.
9 —	Reserved
8 CMDIOC	Command interrupt on complete When this bit is set, the device controller issues a generic endpoint command complete event after executing the command. Note that this interrupt is mapped to DEPCFG[INTRNUM]. When the DEPCFG command is executed, the command interrupt on completion goes to the interrupt pointed by the DEPCFG[INTRNUM] in the current command. **NOTE:** This bit must not set to 1 if the DCTL[RUN_STOP] bit is 0.
7-4 —	Reserved
3-0 CMDTYP	Command type Specifies the type of command the software driver is requesting the core to perform. 0000b - Reserved 0001b - Set endpoint configuration: 64-bit or 96-bit parameter 0010b - Set endpoint transfer resource configuration: 32-bit parameter 0011b - Get endpoint state: no parameter needed 0100b - Set stall: no parameter needed 0101b - Clear stall (see set stall): no parameter needed 0110b - Start transfer: 64-bit parameter

Field	Function
	0111b - Update transfer: no parameter needed 1000b - End transfer: no parameter needed 1001b - Start new configuration: no parameter needed

33.2.58 OTG configuration register (OCFG)

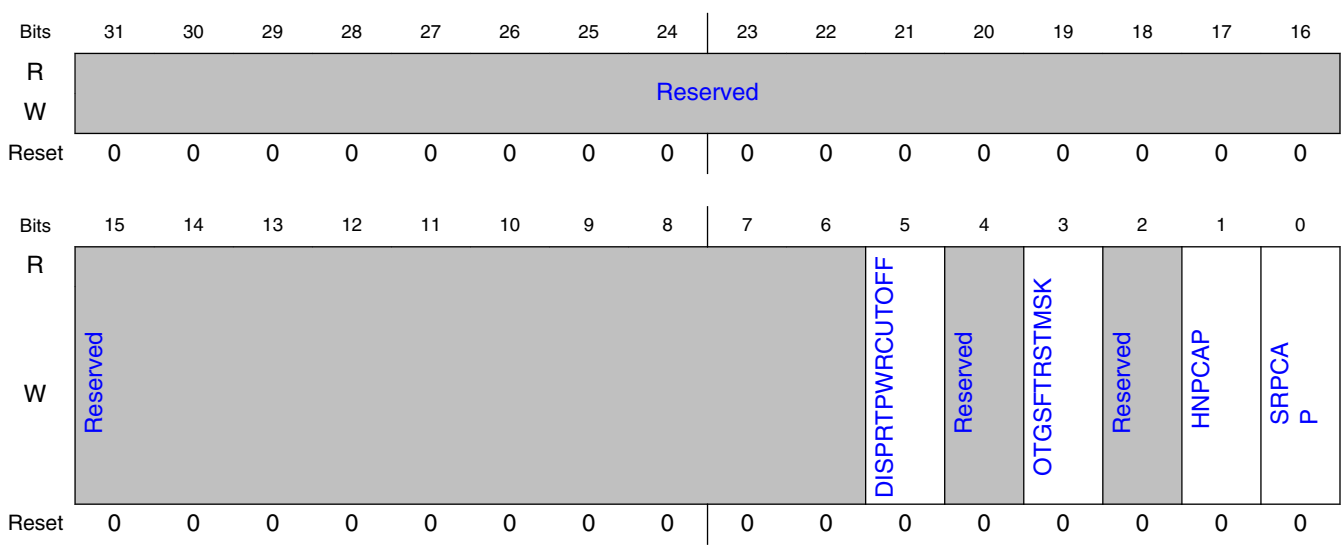
33.2.58.1 Offset

Register	Offset
OCFG	CC00h

33.2.58.2 Function

This register specifies the HNP and SRP capability of the USB 3.0 core.

33.2.58.3 Diagram



33.2.58.4 Fields

Field	Function
31-6 —	Reserved
5 DISPRTPWRCUTOFF	OTG disable port power cut off 0b - The core automatically switches-off the VBUS by clearing the OCTL[PRTPWRCTL] after A_WAIT_BCON timeout whenever the port is disconnected in disconnected state. 1b - The core maintains VBUS ON even after A_WAIT_BCON timeout when port is in disconnected state. The core remains in A_WAIT_BCON state continuously waiting for a Connect.
4 —	Reserved
3 OTGSFTRSTMSK	OTG soft reset mask This bit is used to mask specific soft resets from affecting the OTG functionality of the core. When set, the xHCI-based USBCMD[HCRST] in host mode and DCTL[CSFTRST] in device mode are masked from affecting reset signal outputs sent to the PHY, the OTG FSM logic of the core and also the resets to the VBUS filters inside the core. This bit can be programmed to allow existing xHCI flows (with USBCMD[HCRST] programming) to function in OTG scenarios without any software changes. This bit should be programmed only when GCTL[PRTCAPDIR] = 2'b11. Otherwise it should be set at 1'b0. **NOTE:** When using the core for OTG2 applications, it is not recommended to program USBCMD[HCRST] during role switch. 0b - The xHCI-based USBCMD[HCRST] and DCTL[CSFTRST] resets the OTG logic of the core 1b - The xHCI-based USBCMD[HCRST] and DCTL[CSFTRST] are masked from the OTG logic of the core
2 —	Reserved
1 HNPCAP	HNP capability The application uses this bit to control the USB 3.0 core's HNP capabilities. 0b - HNP capability is not enabled 1b - HNP capability is enabled
0 SRPCAP	SRP capability The application uses this bit to control the USB 3.0 core's SRP capabilities. If this bit is not set for B-device, it cannot request the connected A-device (host) to activate VBUS and start a session. If this bit is not set for A-device, it cannot detect the SRP from B-device (device) to activate VBUS and start a session. 0b - SRP capability is not enabled 1b - SRP capability is enabled

33.2.59 OTG control register (OCTL)

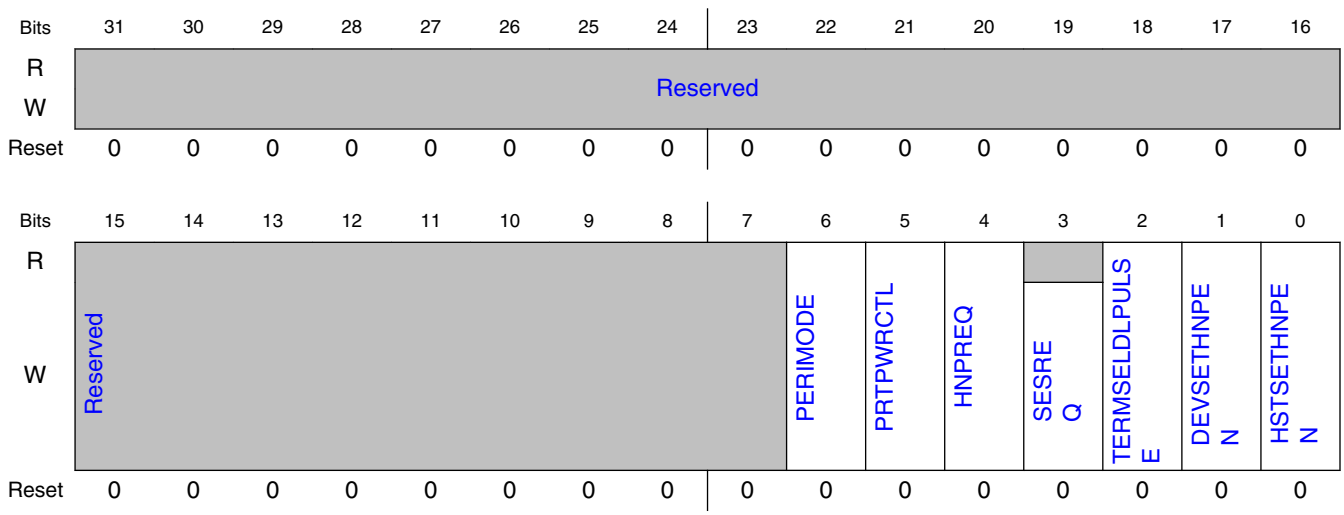
33.2.59.1 Offset

Register	Offset
OCTL	CC04h

33.2.59.2 Function

This register controls the behavior of the OTG function of the core.

33.2.59.3 Diagram



33.2.59.4 Fields

Field	Function
31-7 —	Reserved
6 PERIMODE	Peripheral mode Application uses this bit to program the core to work as a peripheral or as a host. 0b - The OTG device acts as a host 1b - The OTG device acts as a peripheral
5 PRTPWRCTL	Port power control Application sets this bit to initiate VBUS drive when it is an A- device. The application should clear this bit only if it wants to switch off the VBUS to B-device. The core clears this bit in the following conditions:

Table continues on the next page...

Field	Function
	• Transition from any state to A-IDLE state defined in OTG 2.0 state machine. • When AIDL_BDIS_TOUT occurs in A_SUSPEND • When A_WAIT_BCON_TOUT occurs in A_WAIT_BCON • Transition to any B-state defined in OTG 2.0 state machine
4 HNPREQ	HNP request The application sets this bit to initiate a HNP request to the connected USB host. The application clears this bit by writing a 1'b0 when either of the following is detected: OEVT[OTGBDEVBHOSTENDEVNT] OEVT[OTGBDEVVBUSCHNGEVNT] 0b - No HNP request 1b - HNP request
3 SESREQ	Session request The application sets this bit to initiate a session request on the USB. Writing 1'b1 to this bit triggers the core to send SRP (data line pulsing) on PHY interface. In the absence of OEVT[OTGBDevSessVldDetEvnt] after a session request, the application must wait for atleast TB_SRP_FAIL time (6 secs) before initiating another session request. This bit returns 1'b0 when read. 0b - No session request 1b - Session request
2 TERMSELDLPULSE	TermSel DLine pulsing selection This bit selects utmi_termselect to drive data line pulse during SRP. 0b - Data line pulsing using utmi_txvalid (default) 1b - Data line pulsing using utmi_termsel
1 DEVSETHNPEN	Device set HNP enable 0: HNP is not enabled in the application 1: HNP is enabled in the application The application sets this bit in HS/FS mode, when it successfully receives a SetFeature.SetHNPEnable command from the connected USB host.
0 HSTSETHNPEN	Host set HNP enable The application sets this bit in HS/FS mode, when it has successfully enabled HNP (using the SetFeature.SetHNPEnable command) from the connected device. 0b - Host set HNP is not enabled 1b - Host set HNP is enabled

33.2.60 OTG events register (OEVT)

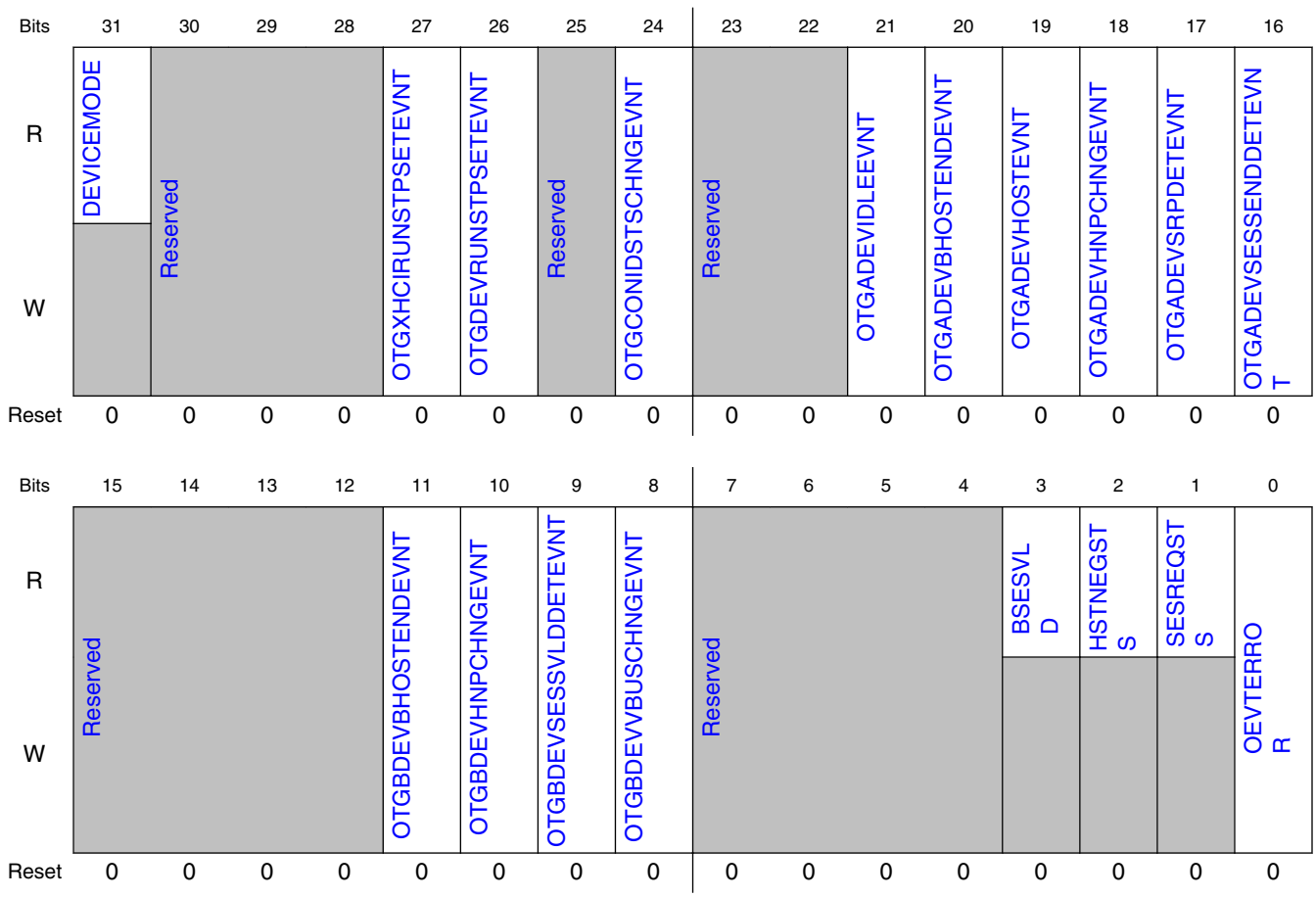
33.2.60.1 Offset

Register	Offset
OEVT	CC08h

33.2.60.2 Function

Any event set in this register causes otg_interrupt signal to go high. Writing 1'b1 to the event information bit in this register clears the register bit and the associated interrupt. The otg_interrupt signal goes low when there are no more pending OTG events.

33.2.60.3 Diagram



33.2.60.4 Fields

Field	Function
31 DEVICEMODE	Device mode Indicates whether the device is in A-device or B-device mode based on utmiotg_iddig The rest of the OTG event information bits (OTGxxxxEVTINFO) in OEVT register are based on the contents of this bit.

Table continues on the next page...

Field	Function
	0b - A-device mode 1b - B-device mode
30-28 —	Reserved
27 OTGXHCIRUNSTPSETEVNT	OTG host run stop set event This event is set when the host driver programs the USBCMD[R/S] bit to 1.
26 OTGDEVRUNSTPSETEVNT	OTG device run stop set event This event is set when the device driver programs the DCTL[RUN_STOP] to 1.
25 —	Reserved
24 OTGCONIDSTSCHNGEVNT	Connector ID status change event Set in both A-device/B-device mode. This event is generated when there is a change in connector ID status.
23-22 —	Reserved
21 OTGADEVIDLEEVNT	A-device A-IDLE event Set in A-device mode only. The event is generated when A-device enters A-IDLE state. This event is set when the OTG 2.0 FSM of the core enters A-IDLE state from any other OTG state.
20 OTGADEVBHOSTENDEVNT	A-device B-Host end event Set in A-device mode only. The event is generated when B-device has completed its host role. **NOTE:** This bit is applicable for OTG 2.0 mode of operation.
19 OTGADEVHOSTEVNT	A-device host event Set in A-device mode only. This event is generated when A-device enters host role. In HS/FS mode, it occurs after the initial connect to a B-device as A-host as well as when there is a role change from A-peripheral to A-host. **NOTE:** This bit is applicable only for OTG 2.0 mode of operation.
18 OTGADEVHNPCHNGEVNT	A-Dev HNP change event Set in A-device mode only. The event is generated when there is an HNP attempt. **NOTE:** This bit is applicable only for OTG 2.0 mode of operation.
17 OTGADEVSRPDETEVNT	SRP detect event Set in A-device mode only. This event is asserted when a session request from the B-device is detected through SRP. **NOTE:** This bit is applicable for OTG 2.0 mode of operation.
16 OTGADEVSESSENDDETEVNT	Session end detected event Set in A-device mode only. This event is asserted when the utmiotg_vbusvalid signal goes low indicating the end of a session. **NOTE:** This bit is applicable for OTG 2.0 mode of operation.
15-12 —	Reserved
11	B-Device B-Host end event

Table continues on the next page...

Field	Function
OTGBDEVBHOSTENDEVNT	Set in B-device mode only. This event is generated when B-device has completed its host role. **NOTE:** This bit is applicable for OTG 2.0 mode of operation.
10 OTGBDEVHNPCHNGEVNT	B-device HNP change event Set in B-device mode only. This event is generated when there is a success or failure of an HNP attempt. **NOTE:** This bit is applicable for OTG 2.0 mode of operation.
9 OTGBDEVSESSVLDDETEVNT	Session valid detected event Set in B-device mode only. This event is asserted when there is a valid VBUS from A-device and B-device succeeds in starting a session. **NOTE:** This bit is applicable for OTG 2.0 mode of operation.
8 OTGBDEVVBUSCHNGEVNT	VBUS change event Set in B-device mode only. This event is asserted when the utmisrp_bvalid signal goes low (indicating the end of a session), or goes high. **NOTE:** This bit is applicable for OTG 2.0 mode of operation.
7-4 —	Reserved
3 BSESVLD	B-Session valid Indicates the device mode transceiver status. The core updates this bit when OEVTEN[OTGBDEVVBUSCHNGEVNT] is set. **NOTE:** This bit is applicable for OTG 2.0 mode of operation. 0b - B-session is not valid 1b - B-session is valid
2 HSTNEGSTS	Host negotiation status The core updates this bit when any of the following bits is set: OEVTEN[OTGADEVHNPCHNGEVNT] OEVTEN[OTGBDEVHNPCHNGEVNT] This bit indicates host negotiation success or failure. **NOTE:** This bit is applicable only for OTG 2.0 mode of operation. 0b - Host negotiation failure. In A-device, for HS/FS, this indicates an imminent end of session indication from the core. In B-device, for HS/LS, it indicates that the timer used to wait for an A-device to signal a connection (b_ase0_brst_tmout in OTG 2.0) timed out resulting in B-device staying as B-peripheral. 1b - Host negotiation success. This indicates that the host negotiation was successful.
1 SESREQSTS	Session request status Ignore this bit.
0 OEVTERROR	OTG event error There are no errors currently defined.

33.2.61 OTG events enable register (OEVTEN)

33.2.61.1 Offset

Register	Offset
OEVTEN	CC0Ch

33.2.61.2 Function

Setting a bit in this register enables the generation of corresponding events in OEVT and assertion of otg_interrupt due to this event. When the enable bit is 1'b0, the event is not set in OEVT and otg_interrupt is not asserted due to this event.

33.2.61.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved				OTGXHCIRUNSTPSETEVNTEN	OTGDEVRUNSTPSETEVNTEN	Reserved	OTGCONIDSTSCHNGEVNTEN	HRRCONFNOTIFEVNTEN	HRRINITNOTIFEVNTEN	OTGADEVIDLEEVNTEN	OTGADEVBHOSTENDEVNTEN	OTGADEVHOSTEVNTEN	OTGADEVHNPCHNGEVNTEN	OTGADEVSRPDETEVNTEN	OTGADEVSESSENDDETEVNTEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved				OTGBDEVBHOSTENDEVNTEN	OTGBDEVHNPCHNGEVNTEN	OTGBDEVSESSVLDDETEVNTEN	OTGBDEVVBUSCHNGEVNTEN	Reserved							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

33.2.61.4 Fields

Field	Function
31-28 —	Reserved
27 OTGXHCIRUNSTPSETEVNTEN	OTG host run stop set event enable When this bit is set, OEVT[XHCIRUNSTPSET] event is enabled. If not, that event is disabled.
26 OTGDEVRUNSTPSETEVNTEN	OTG device run stop set event enable When this bit is set, OEVT[DEVRUNSTPSET] event is enabled. If not, that event is disabled.
25 —	Reserved
24 OTGCONIDSTSCHNGEVNTEN	OTGCOMMONEVTINFOEN[0] Connector ID status change event enable (OTGCONIDSTSCHNGEVNTEN). When this bit is set, OEVT[OTGCONIDSTSCHNGEVNT] is enabled. If not, the event is disabled.
23 HRRCONFNOTIFEVNTEN	OTGCOMMONEVTINFOEN[2] HRRCONFNOTIF event enable (HRRCONFNOTIFEVNTEN). When this bit is set, OEVT[HRRCONFNOTIFEVNT] is enabled. If not, the event is disabled.
22 HRRINITNOTIFEVNTEN	OTGCOMMONEVTINFOEN[1] HRRINITNOTIF event enable (HRRINITNOTIFEVNTEN). When this bit is set, OEVT[HRRINITNOTIFEVNT] is enabled. If not, the event is disabled.
21 OTGADEVIDLEEVNTEN	OTGADEVEVTINFOEN[5] A-device A-IDLE event (OTGADEVIDLEEVNTEN) When this bit is set, OEVT[OTGADEVIDLEEVNT] is enabled. If not, the event is disabled.
20 OTGADEVBHOSTENDEVNTEN	OTGADEVEVTINFOEN[4] A-device B-host end event enable (OTGADEVBHOSTENDEVNTEN) When this bit is set, OEVT[OTGADEVBHOSTENDEVNT] is enabled. If not, the event is disabled.
19 OTGADEVHOSTEVNTEN	OTGADEVEVTINFOEN[3] A-device host event (OTGADEVHOSTEVNTEN) When this bit is set, OEVT[OTGADEVHOSTEVNT] is enabled. If not, the event is disabled
18 OTGADEVHNPCHNGEVNTEN	OTGADEVEVTINFOEN[2] A-Device HNP change event enable (OTGADEVHNPCHNGEVNTEN) When this bit is set, OEVT[OTGADEVHNPCHNGEVNT] is enabled. If not, the event is disabled
17 OTGADEVSRPDETEVNTEN	OTGADEVEVTINFOEN[1] SRP detect event enable (OTGADEVSRPDETEVNTEN). When this bit is set, OEVT[OTGADEVSRPDETEVNT] is enabled. If not, the event is disabled.
16 OTGADEVSESSENDDETEVNTEN	OTGADEVEVTINFOEN[0] Session end detected event enable (OTGADEVSESSENDDETEVNTEN) When this bit is set, OEVT[OTGADEVSESSENDEVNT] is enabled. If not, the event is disabled
15-12	Reserved

Table continues on the next page...

Field	Function
—	
11 OTGBDEVBHOSTENDEVNTEN	OTGBDEVEVTINFOEN[3] B-device B-host end event enable (OTGBDEVBHOSTENDEVNTEN) When this bit is set, OEVT[OTGBDEVHOSTENDEVNT] is enabled. If not, the event is disabled
10 OTGBDEVHNPCHNGEVNTEN	OTGBDEVEVTINFOEN[2] B-device HNP change event enable (OTGBDEVHNPCHNGEVNTEN) When this bit is set, OEVT[OTGBDEVHNPCHNGEVNT] is enabled. If not, the event is disabled
9 OTGBDEVSESSVLDDETEVNTEN	OTGBDEVEVTINFOEN[1] Session valid detected event enable (OTGBDEVSESSVLDDETEVNTEN) Set in B-device mode only. This event is asserted when there is a valid VBUS from A- device and B-device succeeds in starting a session.
8 OTGBDEVVBUSCHNGEVNTEN	OTGBDEVEVTINFOEN[0] VBUS change event enable (OTGBDEVVBUSCHNGEVNTEN) When this bit is set, OEVT[OTGBDEVVBUSCHNGEVNT] is enabled. If not, the event is disabled
7-0 —	Reserved

33.2.62 OTG status register (OSTS)

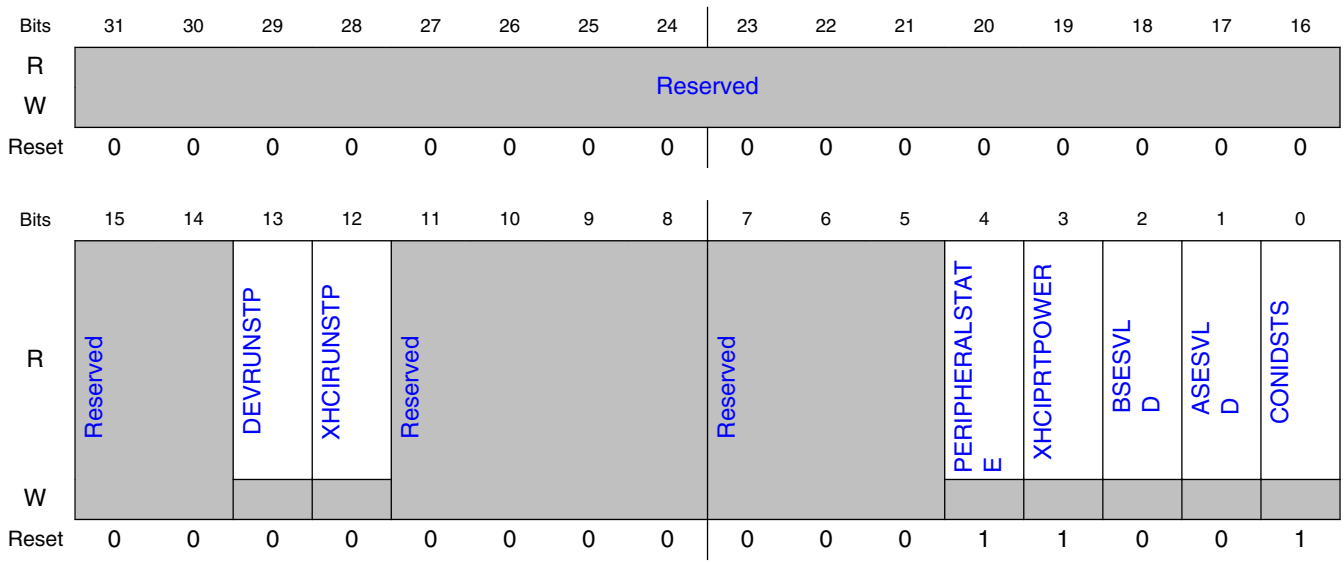
33.2.62.1 Offset

Register	Offset
OSTS	CC10h

33.2.62.2 Function

The OTG status register reflects the status of the OTG function of the core.

33.2.62.3 Diagram



33.2.62.4 Fields

Field	Function
31-14 —	Reserved
13 DEVRUNSTP	This bit reflects the status of the DCTL[RUN_STOP] bit. 0b - Device Run/Stop is set to 0 1b - Device Run/Stop is set to 1
12 XHCIRUNSTP	OTG host run stop set event This event is set when the host driver programs the [USBCMD[R/S]] to 1'b1.
11-8 —	Reserved
7-5 —	Reserved
4 PERIPHERALSTATE	Indicates whether the core is acting as a peripheral or host. 0b - Host 1b - Peripheral
3 XHCIPRTPOWER	This bit reflects the PORTSC[PP] bit in the xHCI register.
2 BSESVLD	B-session valid Indicates the device mode transceiver status. In OTG mode, applications use this bit to determine if the device is connected.

Table continues on the next page...

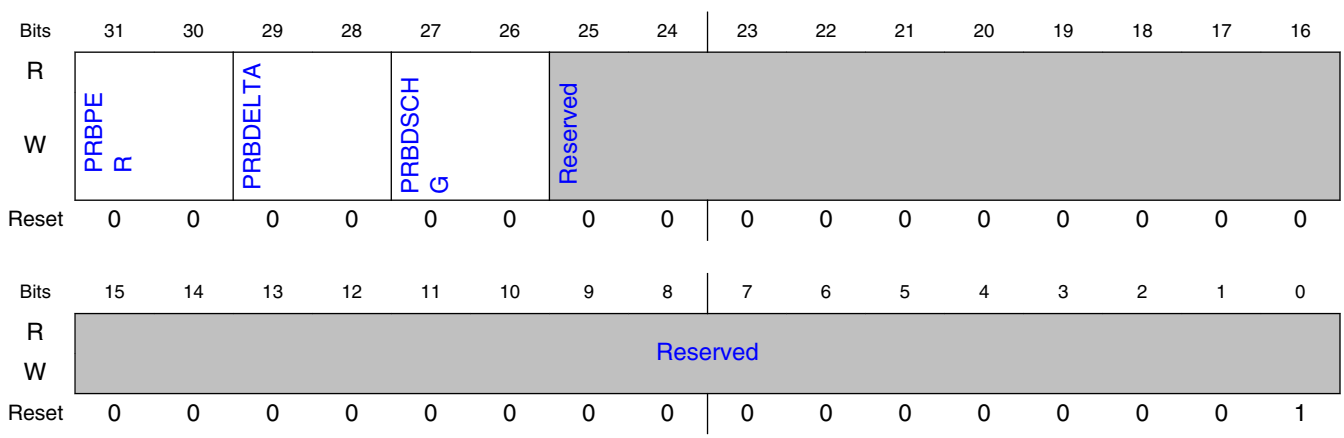
Field	Function
	0b - B-session is not valid 1b - B-session is valid
1 ASESVLD	VBUS valid Indicates the host mode transceiver status. 0b - A-session is not valid 1b - A-session is valid
0 CONIDSTS	Connector ID status **NOTE:** The reset value of this bit depends on the power-on value of the IDDIG signal from the PHY. 0b - The USB 3.0 core is in A-device mode 1b - The USB 3.0 core is in B-device mode

33.2.63 ADP configuration register (ADPCFG)

33.2.63.1 Offset

Register	Offset
ADPCFG	CC20h

33.2.63.2 Diagram



33.2.63.3 Fields

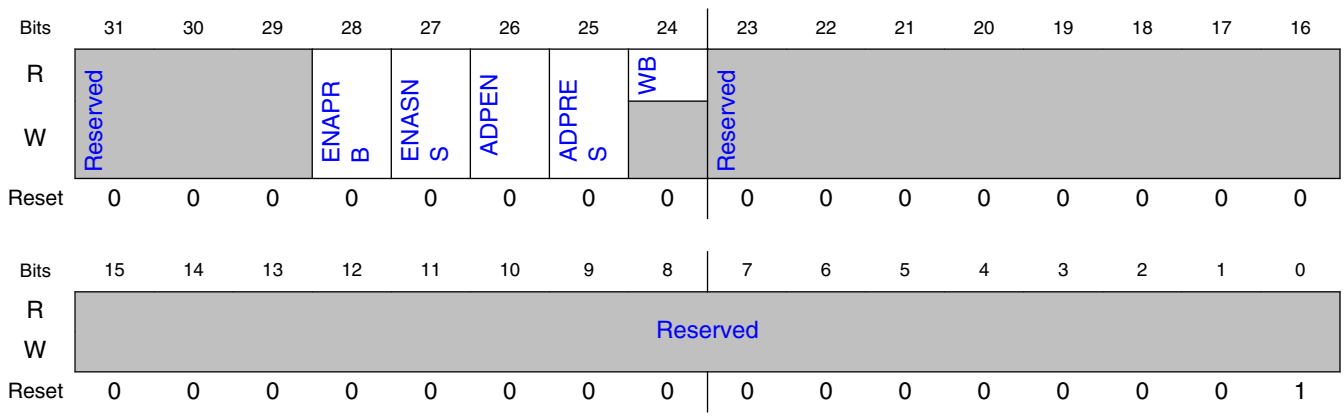
Field	Function
31-30 PRBPER	Probe period This bit sets the value of T_ADP_PRB. The scaledown values for PRBPER are: 00 12.5 ms 01 18.75 ms 10 25 ms 11 31.25 ms 00b - 0.775 sec 01b - 1.55 sec 10b - 2.275 sec 11b - Reserved
29-28 PRBDELTA	Probe delta These bits set the resolution for RTIM value. They are defined in units of 32 KHz clock cycles. For example, if this value is chosen to be 2'b01, it means that RTIM increments for every two 32 KHz clock cycles. 00b - 1 cycle 01b - 2 cycles 10b - 3 cycles 11b - 4 cycles
27-26 PRBDSCHG	Probe discharge These bits set the time for TADP_DSCHG. The scaledown values for the PrbDschg are as follows: 00 62.5 µs 01 125 µs 10 250 µs 11 500 µs 00b - 4 ms 01b - 8 ms 10b - 16 ms 11b - 32 ms
25-0 —	Reserved

33.2.64 ADP control register (ADPCTL)

33.2.64.1 Offset

Register	Offset
ADPCTL	CC24h

33.2.64.2 Diagram



33.2.64.3 Fields

Field	Function
31-29 —	Reserved
28 ENAPRB	Enable probe When set to 1 along with ADPEN, the core performs a probe operation.
27 ENASNS	Enable sense When set to 1 along with ADPEN, the core performs a sense operation.
26 ADPEN	ADP enable When set to 1, the core performs either ADP probing or sensing based on ENAPRB and ENASNS. Setting ADPEN to 0 gates the suspend clock for major portion of ADP related logic.
25 ADPRES	ADP reset When set to 1, the ADP controller is reset. This bit is auto-cleared after the reset procedure is complete in the ADP controller.
24 WB	Write busy The application can read or write ADPCFG and ADPCTL registers only if this bit is cleared. Hardware sets this bit when the write is in progress in the suspend clock domain. 0b - Write completed

Table continues on the next page...

Field	Function
	1b - Write in progress
23-0 —	Reserved

33.2.65 ADP event register (ADPEVT)

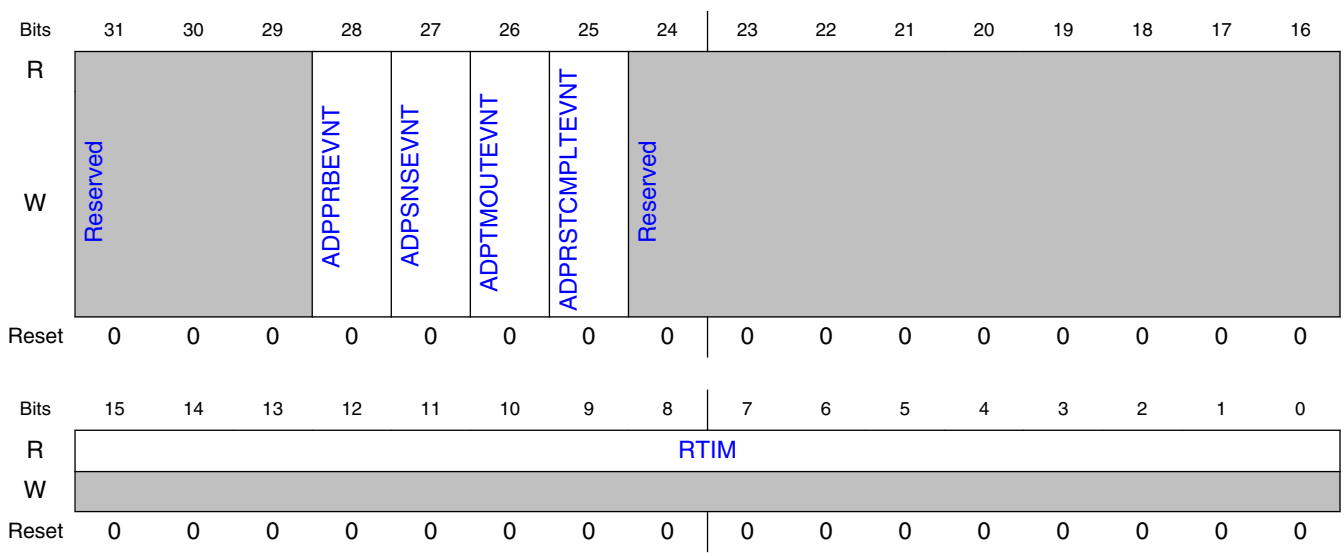
33.2.65.1 Offset

Register	Offset
ADPEVT	CC28h

33.2.65.2 Function

Writing 1 to the information bit in this register clears the register bit and associated interrupt.

33.2.65.3 Diagram



33.2.65.4 Fields

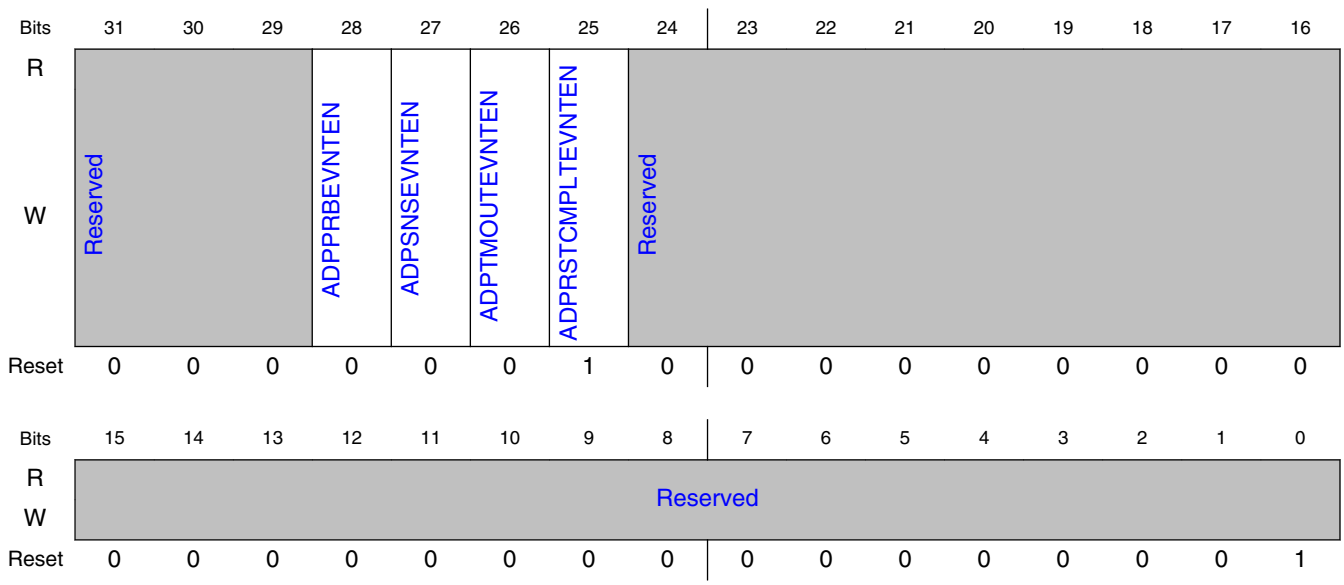
Field	Function
31-29 —	Reserved
28 ADPPRBEVNT	ADPEVTINFO[4] ADP probe event (ADPPRBEVNT) When this event is set, it means that the VBUS voltage is greater than VADPPRB or VADPPRB is reached.
27 ADPSNSEVNT	ADPEVTINFO[3] ADP sense event (ADPSNSEVNT) When this event is set, it means that the VBUS voltage is greater than VADPSNS or VADPSNS is reached.
26 ADPTMOUTEVNT	ADP timeout event This event is relevant when ADP probe command is executed. When this event is set, it means that the ramp time is completed (GADPCTL[RTIM] has reached its terminal value of 0x7FF). This is a debug feature that allows software to read the ramp time after each cycle.
25 ADPRSTCMPLTEVNT	ADP reset complete event This event when set, indicates that the ADP reset command is successful.
24-16 —	Reserved
15-0 RTIM	Ramp time These bits capture the latest time it took for VBUS to ramp from VADP_SINK to VADP_PRB. The bits are defined in units of 32 KHz clock cycles. The maximum time of 65536 cycles corresponds to a time of 2.04 seconds. Note for scaledown ramp_timeout PRBDELTA = 2'b00 => 6250 µs PRBDELTA = 2'b01 => 3125 µs PRBDELTA = 2'b10 => 1562.5 µs PRBDELTA = 2'b11 => 781.25 µs 0000000000000000b - 1 cycle 0000000000000001b - 2 cycles 0000000000000010b - 3 cycles .... 1111111111111111b - 65536 cycles

33.2.66 ADP event enable register (ADPEVTEN)

33.2.66.1 Offset

Register	Offset
ADPEVTEN	CC2Ch

33.2.66.2 Diagram



33.2.66.3 Fields

Field	Function
31-29 —	Reserved
28 ADPPRBEVNTEN	ADP probe event enable When this bit is set, the ADPEVT[ADPPRBEVNT] bit is enabled.
27 ADPSNSEVNTEN	ADP sense event enable When this bit is set, the ADPEVT[ADPSNSEVNT] bit is enabled.
26 ADPTMOUTEVNTEN	ADP timeout event enable When this bit is set, the ADPEVT[ADPTMOUTEVNT] bit is enabled.
25	ADP reset complete event enable

Table continues on the next page...

Field	Function
ADPRSTCMPLTEVNTEN	When this bit is set, the ADPEVT[ADPRSTCMPLTEVNT] bit is enabled.
24-0 —	Reserved

33.3 USB PHY SuperSpeed register descriptions

The USBPHY register space must not be accessed if it is powered down.

33.3.1 USB_PHY_SS Memory map

USB3_PHY_REG_SS base address: 84F_0000h

Offset	Register	Width (In bits)	Access	Reset value
0h	SUP_IDCODE_LO (IP_IDCODE_LO)	16	RO	64CDh
2h	SUP_IDCODE_HI (SUP_IDCODE_HI)	16	RO	A21Ch
60h	MPLL_LOOP_CTL (MPLL_LOOP_CTL)	16	RW	0040h
200Ch	LANE0_RX_OVRD_IN_HI (LANE0_RX_OVRD_IN_HI)	16	RW	0000h

33.3.2 SUP_IDCODE_LO (IP_IDCODE_LO)

33.3.2.1 Offset

Register	Offset
IP_IDCODE_LO	0h

33.3.2.2 Diagram

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

33.3.2.3 Fields

Field	Function
15-0 DATA	DATA These bits indicate the IP version.

33.3.3 SUP_IDCODE_HI (SUP_IDCODE_HI)

33.3.3.1 Offset

Register	Offset
SUP_IDCODE_HI	2h

33.3.3.2 Diagram

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

33.3.3.3 Fields

Field	Function
15-0	DATA

Field	Function
DATA	These bits indicate the IP version.

33.3.4 MPLL_LOOP_CTL (MPLL_LOOP_CTL)

33.3.4.1 Offset

Register	Offset
MPLL_LOOP_CTL	60h

33.3.4.2 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								PROP_CNTRL				Reserved			
Reset	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

33.3.4.3 Fields

Field	Function
15-8 —	-
7-4 PROP_CNTRL	PROP_CNTRL Charge pump proportional current setting
3-0 —	-

33.3.5 LANE0_RX_OVRD_IN_HI (LANE0_RX_OVRD_IN_HI)

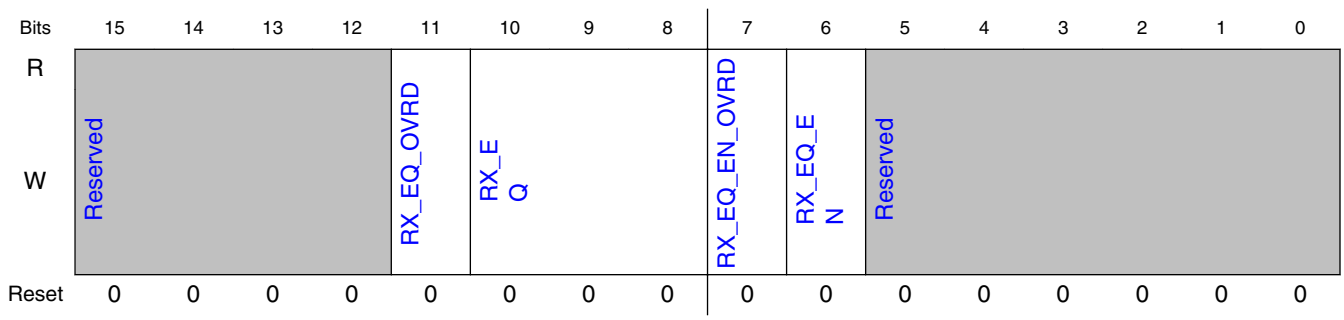
33.3.5.1 Offset

Register	Offset
LANE0_RX_OVRD_IN_HI	200Ch

33.3.5.2 Function

For initialization details, refer Initialization/application information.

33.3.5.3 Diagram



33.3.5.4 Fields

Field	Function
15-12 —	-
11 RX_EQ_OVRD	RX_EQ_OVRD Override value for rx_eq
10-8 RX_EQ	RX_EQ Override value for rx_eq
7 RX_EQ_EN_OVRD	RX_EQ_EN_OVRD Override enable for rx_eq_en
6 RX_EQ_EN	RX_EQ_EN Override value for rx_eq_en
5-0 —	Reserved.

33.4 Functional Description

33.4.1 System memory descriptor and data buffers

The software creates transfer request blocks (TRBs), and four DWORDs each, that point to the data buffers. Normally, the TRBs are allocated consecutively in system memory; only the data buffers can be scattered. In the case of a circular buffer, the link-TRB points to the next TRB. Once the TRBs and data buffers are set up in the system memory, the software driver issues a start transfer command that points to the location of the first TRB in the system memory to start the DMA operation. TRBs, though small (only four DWORDs), provide a rich set of features for the software to schedule transfers, isochronous, control, interrupt moderation, and so on.

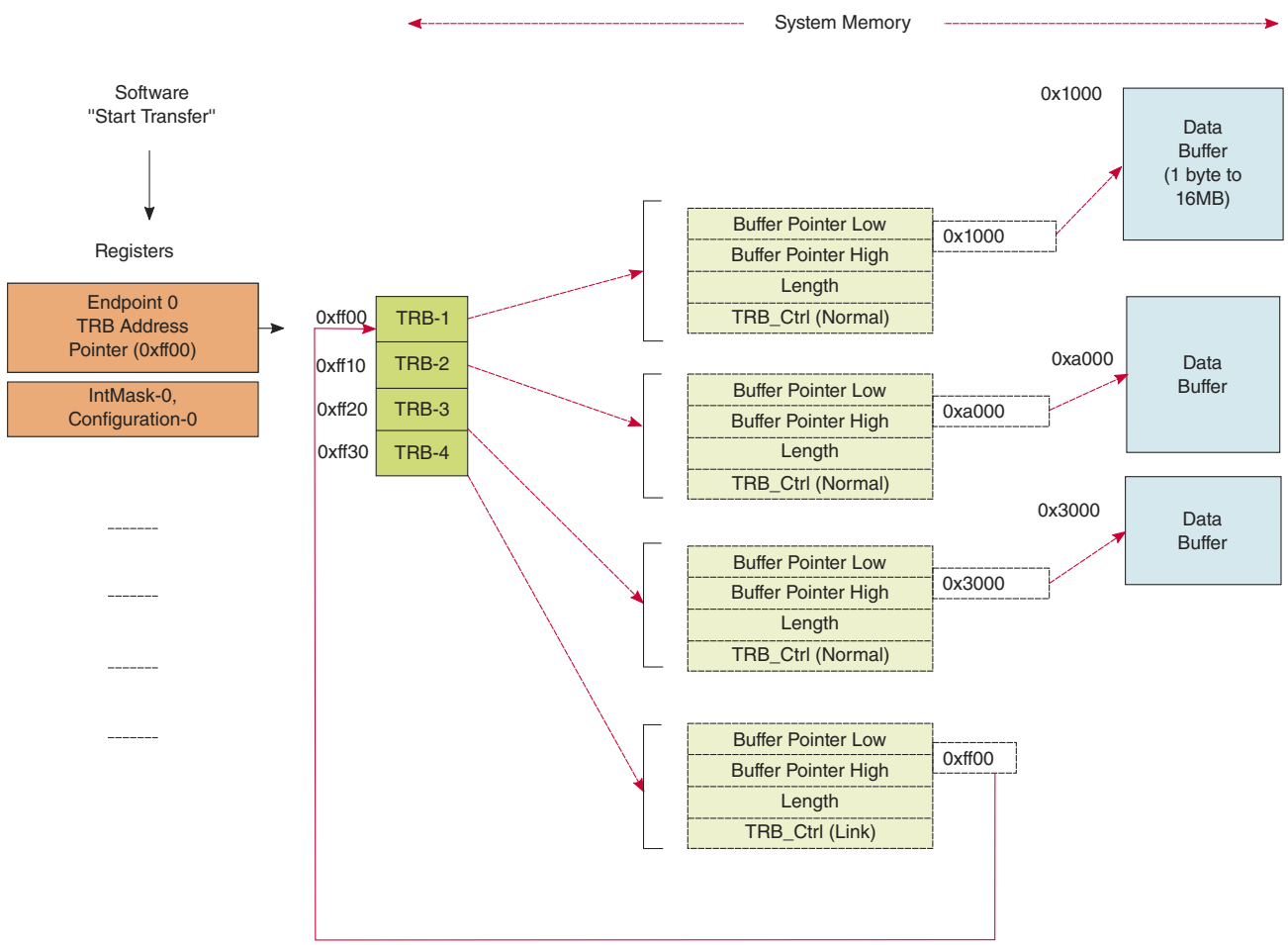


Figure 33-4. System Memory Descriptor and Data Buffers

33.4.2 Device descriptor structures

Device mode transfer request blocks (TRBs) are small 4 DWORDs and at the same time provide a rich set of features so that the software can efficiently manage the USB core, memory buffers, and MIPS requirements.

The following is a list of device mode TRB characteristics:

- TRBs provide support for scattered data structures. The scattered data buffers can be zero bytes to 16 MB in length.
- TRBs must be placed in system memory aligned to a 16-byte boundary.
- TRBs are kept in linear memory to enhance descriptor caching performance. If the data buffers are small (as in an Ethernet application) the core can efficiently collect the scattered data to build USB packets without wasting bus efficiency that collecting scattered descriptors requires.
- Supports TRB linking. This allows software to set up a circular ring of TRBs with the last TRB linking to the first one.
- Supports system memory interface with 40-bit addressing capability
- Supports byte-aligned buffers for each TRB of a transfer. This feature prevents the need for buffer copying in cases where the USB device driver receives unaligned data buffers from other applications (for example, Ethernet). Whenever the application controls buffer allocation, it must allocate SoC bus width and burst-aligned buffers to facilitate efficient bus and memory utilization. For transmitting, the core supports byte-aligned buffers on all TRBs.

  For example: To use 16 bursts in a 64-bit system, you must try to allocate buffers that are 16 * 8 = 128 bytes aligned. This is the normal buffer structure because Linux-like OSs allocate 4 KB buffers. SDR/DDR memory controllers also provide better performance when requests are burst aligned.

- Supports software queueing of multiple USB transfers (LST bit for IN/OUT transfers and CSP bit for OUT transfers control this function).
- Provides interrupt moderation capability, allowing software to selectively enable events on TRB completion, as controlled by the IOC (Interrupt on completion) bit. The IOC event is also used by software to reallocate the released buffers, allowing software to re-use just a few buffers in a circular fashion. This helps when the memory is limited but have enough MIPS to process interrupts. Larger buffers can be allocated to reduce the number of interrupts.
  - In USB 3.0 larger transfers are recommended because the raw transfer rate is almost 10 times faster, unlike USB 2.0, where drivers set up only 64 KB or 128 KB transfers. For example, a 64 KB transfer that takes 1.3 ms in USB 2.0

requires only 164 s in USB 3.0. If you set up buffers of 64 KB and enable the transfer completion event, then you receive an interrupt every 164 sec.

- Supports streaming (Stream ID field used for this purpose).

33.4.2.1 Structures

This figure shows the control and status field of a transfer request block.

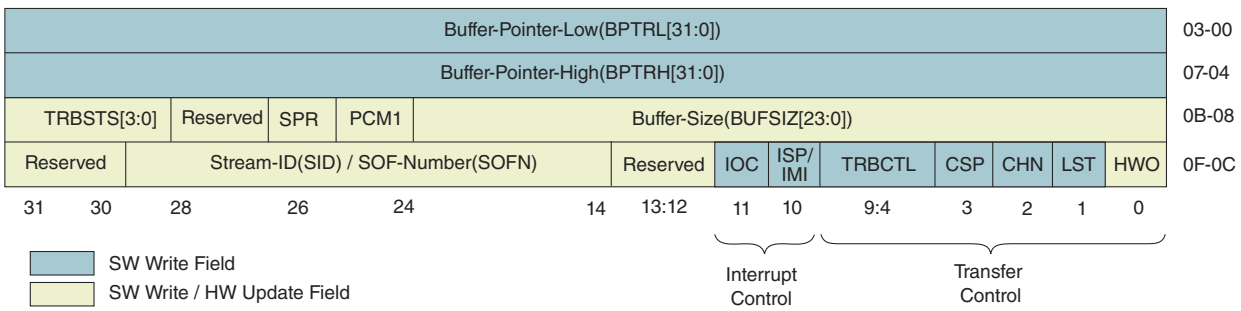


Figure 33-5. TRB control and status fields

Table 33-8. Device descriptor structure field definitions

Field	Description	Hardware access
DW 03-00		
31:0	Buffer pointer low (BPTRL) Data buffer pointer to low 32 bit address (BPTR[31:0]). Hardware may also update this field (implementation specific).	R/W
DW 07-04		
31:0	Buffer pointer high (BPTRH) Data buffer pointer to high 32-bit address (BPTR[63:32]).	R/W
DW 0B-08		
31:28	TRB status (TRBSTS) Hardware updates this field with transfer status information before releasing the TRB. • 4'h0: OK • 4'h1: MissedIsoc: Isochronous interval missed or incomplete • 4'h2: SetupPending - During the current control transfer data/status phase, another SETUP was received. • 4'h4: TransferInProgress - During the current transfer, an end transfer command was received.	R/W
27	Reserved. Hardware may update these bits for internal usage.	R/W
26	SPR • If GUCTL1[DEV_TRB_OUT_SPR_IND] is 1'b1, this field indicates the short packet received for the last processed TRB for the transfer. If a short packet has been	R/W

Table continues on the next page...

Table 33-8. Device descriptor structure field definitions (continued)

Field	Description	Hardware access
	received, this TRB field is set during an OUT transfer TRB write back if this is the last TRB used for that transfer descriptor. • If GUCTL1[DEV_TRB_OUT_SPR_IND] is 1'b0, this field is reserved. Hardware may update these bits for internal usage.	
25:24	Packet count M1 (PCM1) For high-speed, High bandwidth isochronous IN endpoints, this field in an Isoc-First TRB represents the total number of packets in the Buffer Descriptor minus 1.	R/W
23:0	Buffer size (BUFSIZ) If CHN=0 and HWO=0 for the TRB, this field represents the total remaining Buffer Descriptor buffer size in bytes. Valid Range: 0 bytes to (16 MB - 1 byte). The hardware decrements this field to represent the remaining buffer size after data is transferred. For a Link TRB, the buffer size should be "0".	R/W
DW 0F-0C		
31:30	Reserved.	R/W
29:14	Stream ID / SOF Number For stream-based bulk endpoints: The Stream ID of the transfer this TRB is associated with. Stream ID must be the same in all TRBs (R/W). For isochronous endpoints: The (micro)frame number in which the last packet of this TRB's buffer was transmitted or received (debug purposes only) (RO).	R/W
13:12	Reserved.	R/W
11	Interrupt on Complete (IOC) When IOC is set in a TRB, and once the transfer for this buffer is completed, the core will issue XferInProgress event with IOC bit set in the event's status. This indicates the buffer is available for software to reuse or release.	R
10	Interrupt on Short Packet / Interrupt on Missed ISOC (ISP/IMI) Applicable to OUT endpoints when a short packet is received, and CSP=1 and LST=0. If this bit is 1, the core generates an XferInProgress event. For Isochronous endpoints: If this bit is 1, the core generates an XferInProgress event when the interval represented by the Buffer Descriptor completes with a "Missed Isoc" status.	R
9:4	TRB Control (TRBCTL) Indicates the type of TRB: 1: Normal (Control-Data-2+ / Bulk / Interrupt) - Set TRBCTL to 1 for all TRBs used in data stage except the first TRB 2: Control-Setup 3: Control-Status-2 - Set TRBCTL to 3 for a SETUP request without data stage 4: Control-Status-3 - Set TRBCTL to 4 for a SETUP request with data stage 5: Control-Data - Set TRBCTL to 5 for the first TRB of a data stage 6: Isochronous-First - Set TRBCTL to 6 for the first TRB of a Service Interval 7: Isochronous 8: Link TRB Others: Reserved	R

Table continues on the next page...

Table 33-8. Device descriptor structure field definitions (continued)

Field	Description	Hardware access
3	Continue on Short Packet (CSP) Applicable to OUT endpoints only when a short packet is received. If this bit is 1, the core will continue to the next Buffer Descriptor. This setting is required for isochronous endpoints. If this bit is 0, the core will generate an XferComplete event and remove the stream.	R
2	Chain Buffers (CHN) Applicable to IN and OUT endpoints. Set to 1 by software to associate this TRB with the next TRB. A Buffer Descriptor is defined as one or more TRBs. The CHN bit is used to identify the TRBs that comprise a Buffer Descriptor. The CHN bit is always 0 in the last TRB of a Buffer Descriptor and when the LST field is set to 1.	R
1	Last TRB (LST) Indicates this is the last TRB in a list. After completing the transfer for the associated buffer the core will stop the transfer for the endpoint / bulk-stream and issues an XferComplete event. The stream is automatically removed by the hardware.	R
0	Hardware Owner of Descriptor (HWO) Indicates that hardware owns the TRB. Software sets this bit to 1 when it creates the TRB, and cannot modify it until hardware resets this bit to 0. However, there are exceptions for short packets on OUT endpoints and Link TRBs. Because the hardware autonomously checks this bit to determine if the entire TRB is valid, software must set this field to '1' after preparing the other three DWORDs of the TRB with valid information.	R/W

33.4.2.1.1 Normal (Control-Data/Bulk/Interrupt), Isochronous, and Status Transfer Request Block Structure

The Normal TRB is used for Bulk/Control-Data/Interrupt endpoint transfers. The data buffers can be scattered anywhere and each may have different sizes. The TRB Buffer Pointer and the Buffer Size fields point to buffer address and size respectively. The Stream ID for bulk endpoints will be programmed by the application.

For isochronous endpoints, the first TRB in a service interval must have the Isoc-First type, the last TRB in a service interval must have CHN=0, and any other TRBs have CHN=1. The starting (micro)frame time is communicated via the Start Transfer command, and the core tracks the (micro)frame times of the subsequent Buffer Descriptors.

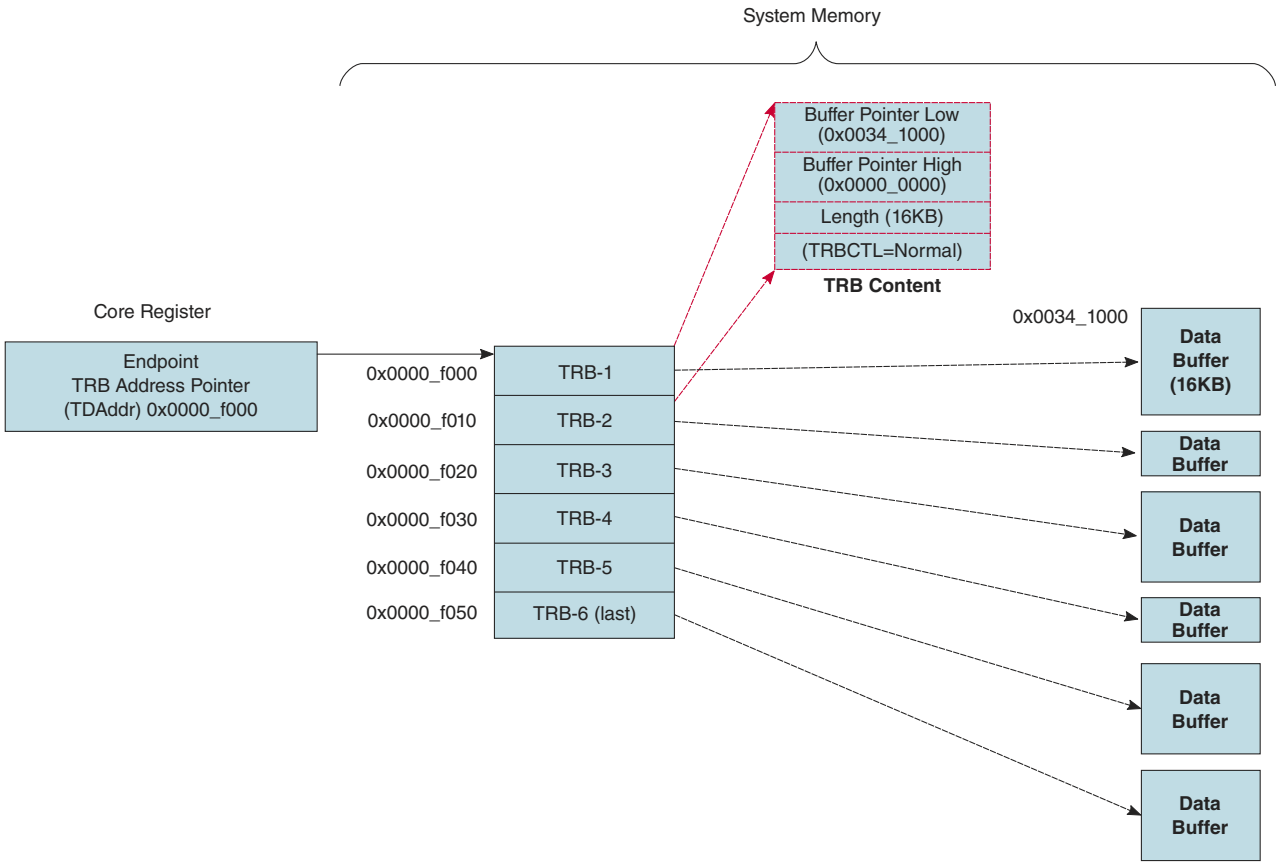


Figure 33-6. Normal (Control-Data/Bulk/Interrupt) Descriptor Structure

33.4.2.1.2 Setup and Status TRB Structure

To receive a SETUP packet, the driver queues up a single Setup TRB, whose buffer pointer value may be set to any address, including the address of the TRB. The buffer size must be set to 8. The core will write the 8 bytes of the received SETUP to the address requested. If the address of the TRB is used, there is no need for a separate data buffer to receive a SETUP packet. After completing Setup stage, driver will schedule Data stage and Status stage transfers. For more information, see Control transfer programming model

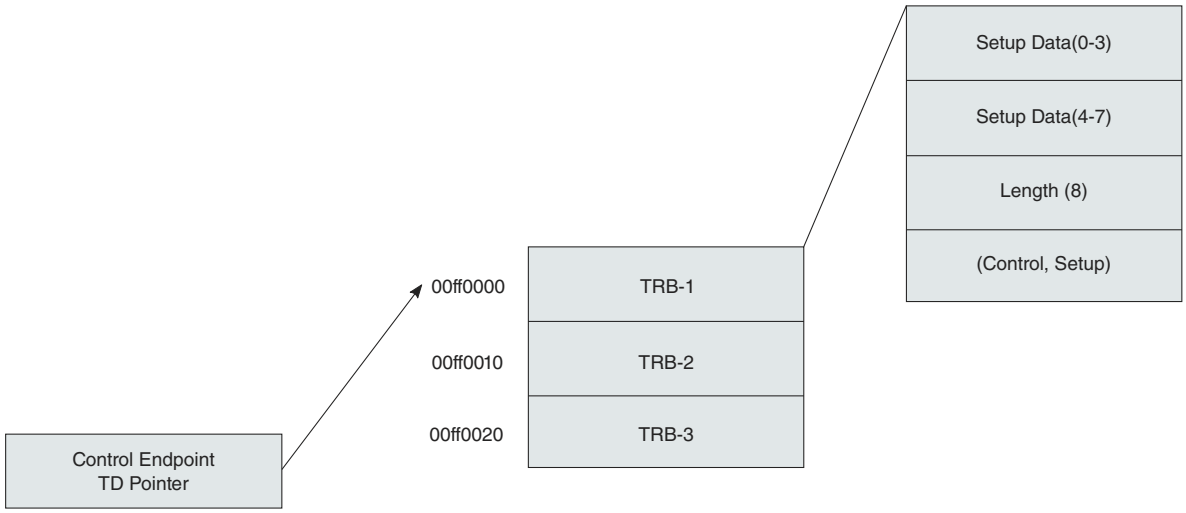


Figure 33-7. SETUP Descriptor Structure with Buffer Pointing to Setup TRB

After interpreting the SETUP bytes, software will determine if the next stage of the control transfer is a data stage or status stage.

If the SETUP bytes require a 3-stage control transfer, the TRB type used in the Data stage must be Control- Data for the first TRB of the Buffer Descriptor. When the host moves on to the Status stage, the TRB Type must be Control-Status-3.

If the SETUP bytes require a 2-stage control transfer, the TRB Type must be Control-Status-2.

The Status TRB is a zero-length TRB, with an unspecified Buffer Pointer value and a Buffer Size of zero. There is no data buffer associated with a Status TRB.

33.4.2.1.3 Link TRB Structure

The Link TRB is used to link back to the starting TRB for reusing TRBs in a circular fashion.

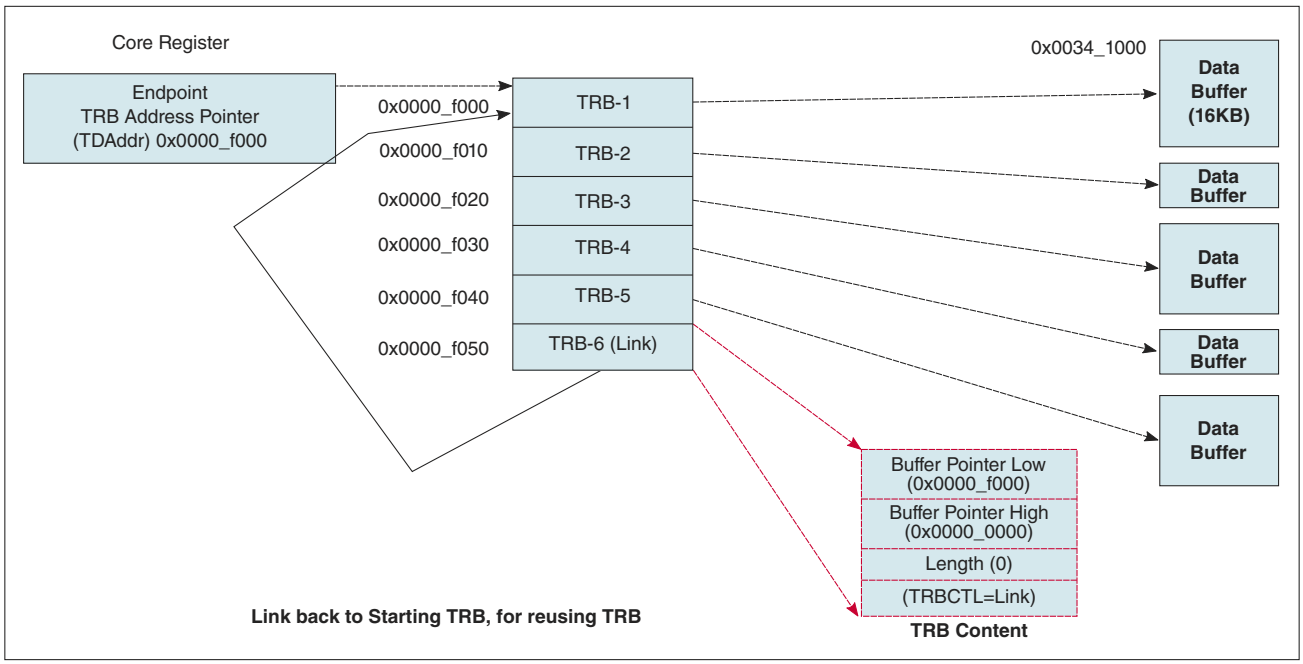


Figure 33-8. Link TRB Structure

If software prepares a circular TRB list which is shorter than the number of cached TRBs (four), the core automatically detects the loop and will not re-fetch a TRB that has already been fetched. For example, for four cached TRBs the software setup three TRBs in the following way:

- TRB-1: Normal, HWO=1
- TRB-2: Normal, HWO=1
- TRB-3: Link to TRB 1

The core will fetch TRBs 1, 2, and 3. Then the core will follow the link to TRB-1, note that its address is the same as the TRB-1 that has already been fetched into the cache, and will temporarily stop fetching TRBs. When TRB-1 completes due to traffic on the USB, the core will write TRB-1 back to memory with the HWO field set to '0', generate a XferInProgress event if necessary, and will automatically attempt to fetch TRB-1 again. In most cases, the core will see the HWO field set to '0' and will stop fetching until software updates the TRB, sets the HWO field back to '1', and issues an Update Transfer command.

33.4.2.1.4 Chaining buffers (CHN) and interrupt on completion (IOC) usage

This figure shows a chaining buffer example for an isochronous IN and a bulk IN transaction.

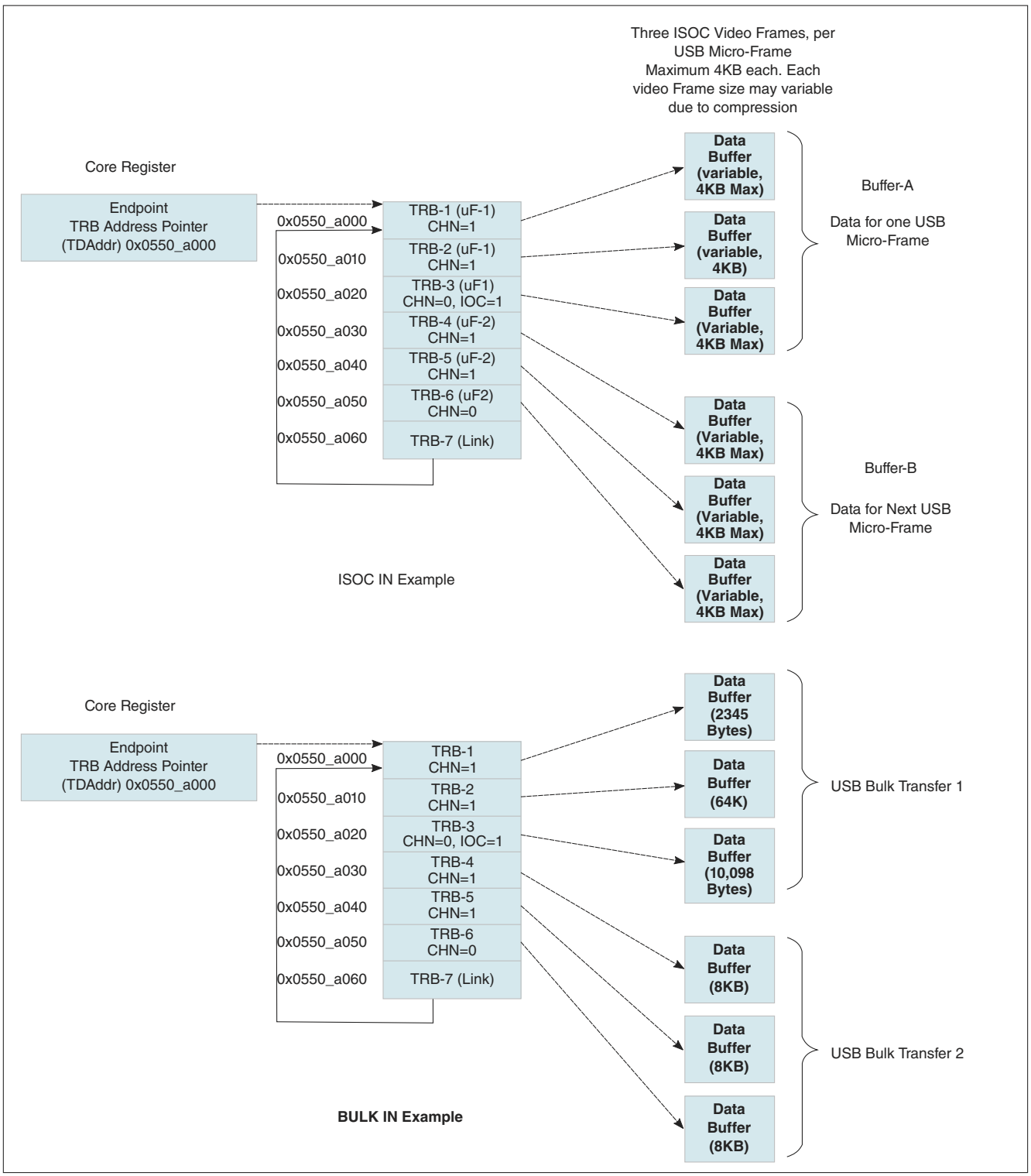


Figure 33-9. Chaining buffers for isochronous and bulk IN

In the isochronous IN application shown in the above figure, there are three video frames to be sent to the host in each microframe. Each video buffer size is maximum of 4 KB and the size may vary for each microframe depending on the compression. The CHN bit

in the TRB-3 is 0, which indicates this is the last buffer of a transfer for the given microframe. The device schedules TRB-1, 2, and 3 in the first bInterval. Similarly, TRB-6 indicates (CHN=0) microframe boundary, and TRB-4, 5, and 6 will be sent during the next bInterval.

For the last packet of last transfer in a microframe, the device internally sets last packet flag when responding at SuperSpeed for isochronous IN transfers.

The application can also use the IOC bit to receive an interrupt when the buffer-A transfer is completed, so it can use buffer-A to send data on the bInterval following the next bInterval. As long as the driver can service the interrupt and set up data before another bInterval, the USB transfers can continue without interruption. Depending upon the system interrupt latency, the buffer size can be adjusted.

If interrupt latency is high occasionally, the software may not have time to set up the linked TRB before the next bInterval. In this case, the core, on seeing HWO bit set to 0 in the TRB, stops processing further TRBs of this endpoint. When hardware receives the Update Transfer command from software, it will re-fetch the TRB. In the case of bulk application, the core will issue the NRDY signal. In isochronous transfers, zero- length packets are sent to the host until software enables the transfer again.

In the bulk IN example, the CHN bit indicates USB Transfer boundary. During a TRB transfer, CHN indicates whether to send the bytes from next TRB buffer as part of the current transfer. For example, even though TRB-1 has 2,345 bytes, the last 297 (2345 - 2 * 1024 = 297) bytes will be combined with the data in TRB-2 and sent as a 1,024 byte packet on the USB since CHN=1. On the other hand, when there is a short packet left in TRB-3, the short packet will be sent separately, since CHN=0.

The device writes data into TRB-1 and TRB-2 for the first transfer. The CHN bit (CHN=0) in TRB-2 indicates that this is the last buffer of the transfer, then TRB-3 and TRB-4 are written for the second transfer. Similarly, the CHN bit (CHN=0) in TRB-4 indicates the second transfer boundary.

If an interrupt latency is high, software may not have time to set up the TRB. In this case, the core, on seeing the HWO bit set to 0 in the TRB, will stop processing further TRBs for the endpoint. Hardware will re-fetch the TRB when software issues the update transfer command. In the case of bulk, control, or interrupt endpoints, the core will issue NRDY. In the case of isochronous endpoints, packets will be dropped until software enables the transfer again.

In the bulk OUT example below, when a short packet is received for TRB-1, TRB-3 will be used for the next OUT transfer and TRB-2 will be closed.

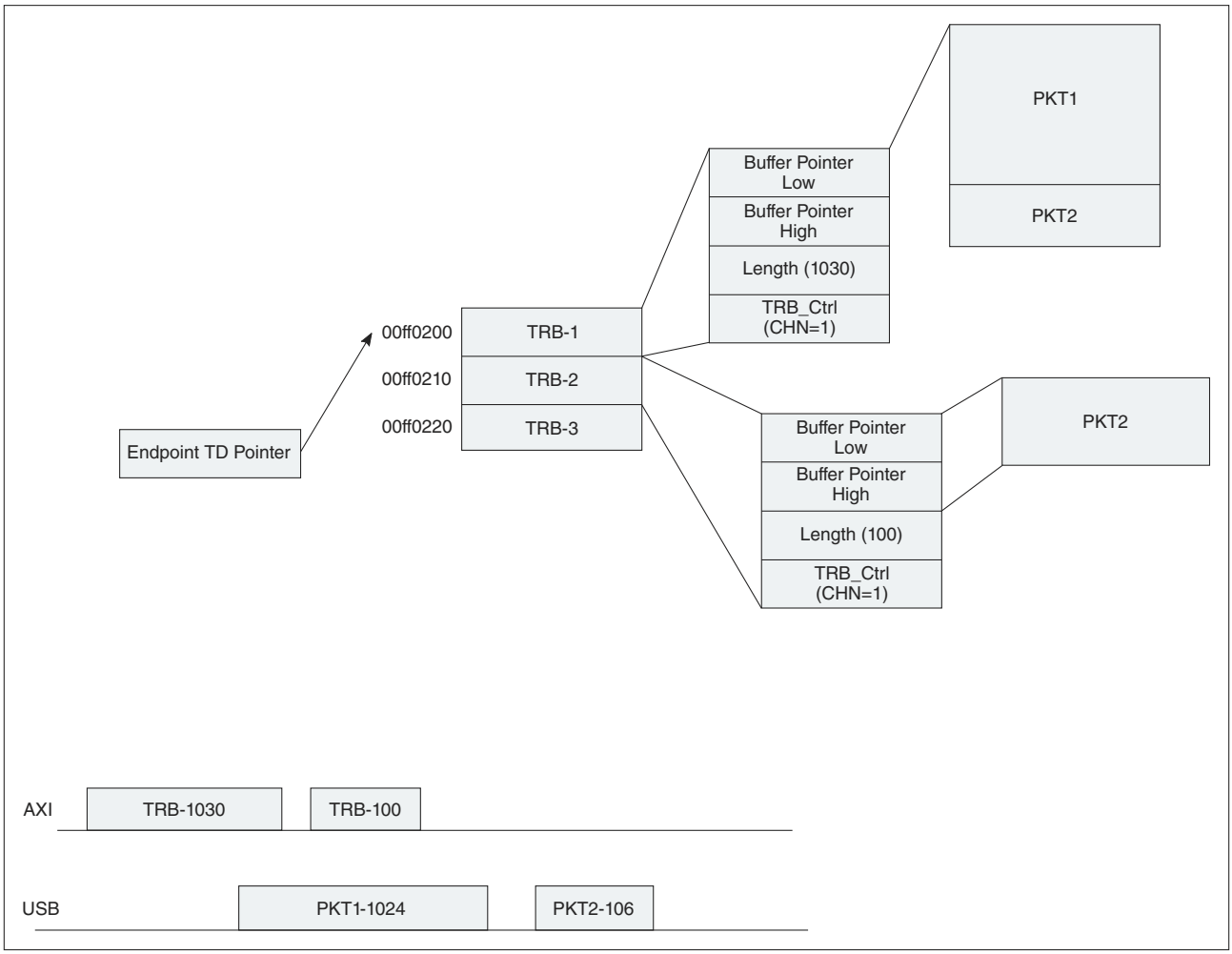


Figure 33-10. Chaining buffers, example 1

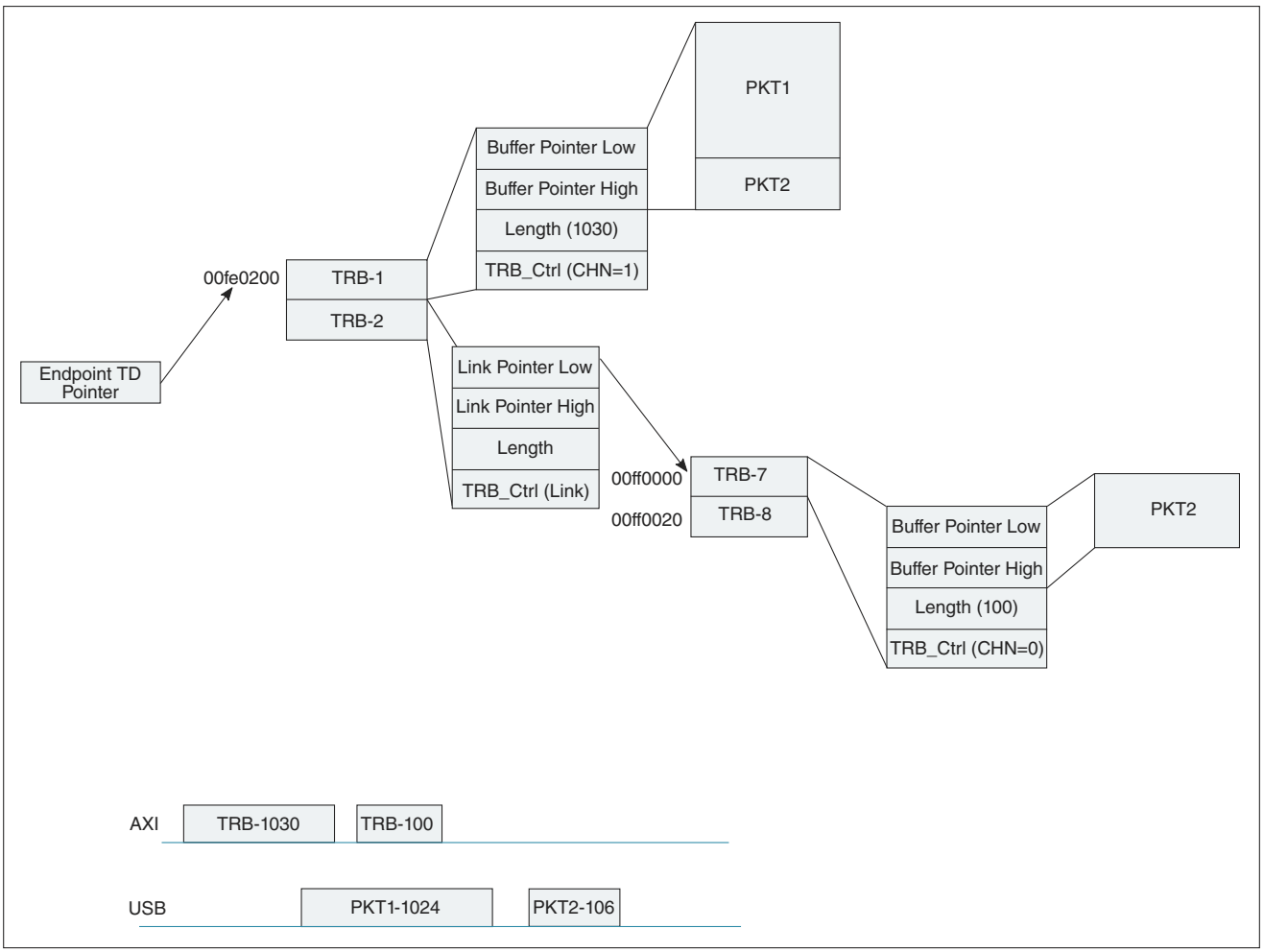


Figure 33-11. Chaining buffers, example 2

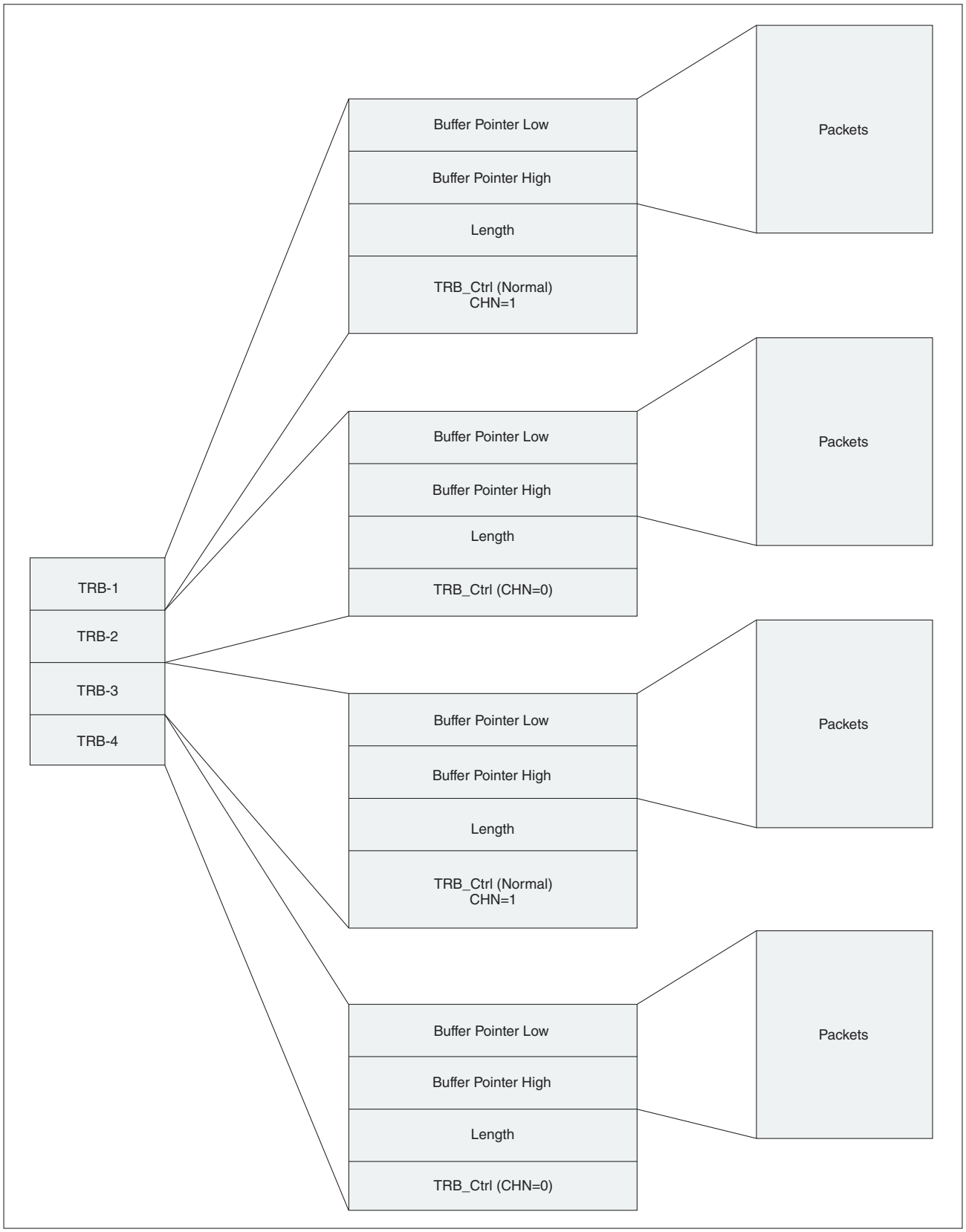


Figure 33-12. Bulk OUT with two transfers

33.4.2.1.5 Interrupt on Short Packet (ISP) and Continue on Short Packet (CSP) Usage

These two bits are used to schedule single or multiple OUT transfer. In most applications where only one OUT transfer per endpoint is scheduled, the software always sets ISP=1 and CSP=0. If the device receives a short packet, then it indicates a USB transfer completion through XferComplete events.

In an Ethernet-over-USB application, where software knows it is going to receive short packets, it can set up multiple transfer size buffers in one step by setting CSP=1 in all the TRBs. On a short packet, the device will update the byte count and move to next TRB. Software can also set ISP once in n number of TRBs to receive an XferInProgress event so it can process the previous short packets.

The following example shows software setting multiple 1500-byte Ethernet transfers and enabling the XferInProgress event once for four Ethernet packets to reduce the number of interrupts.

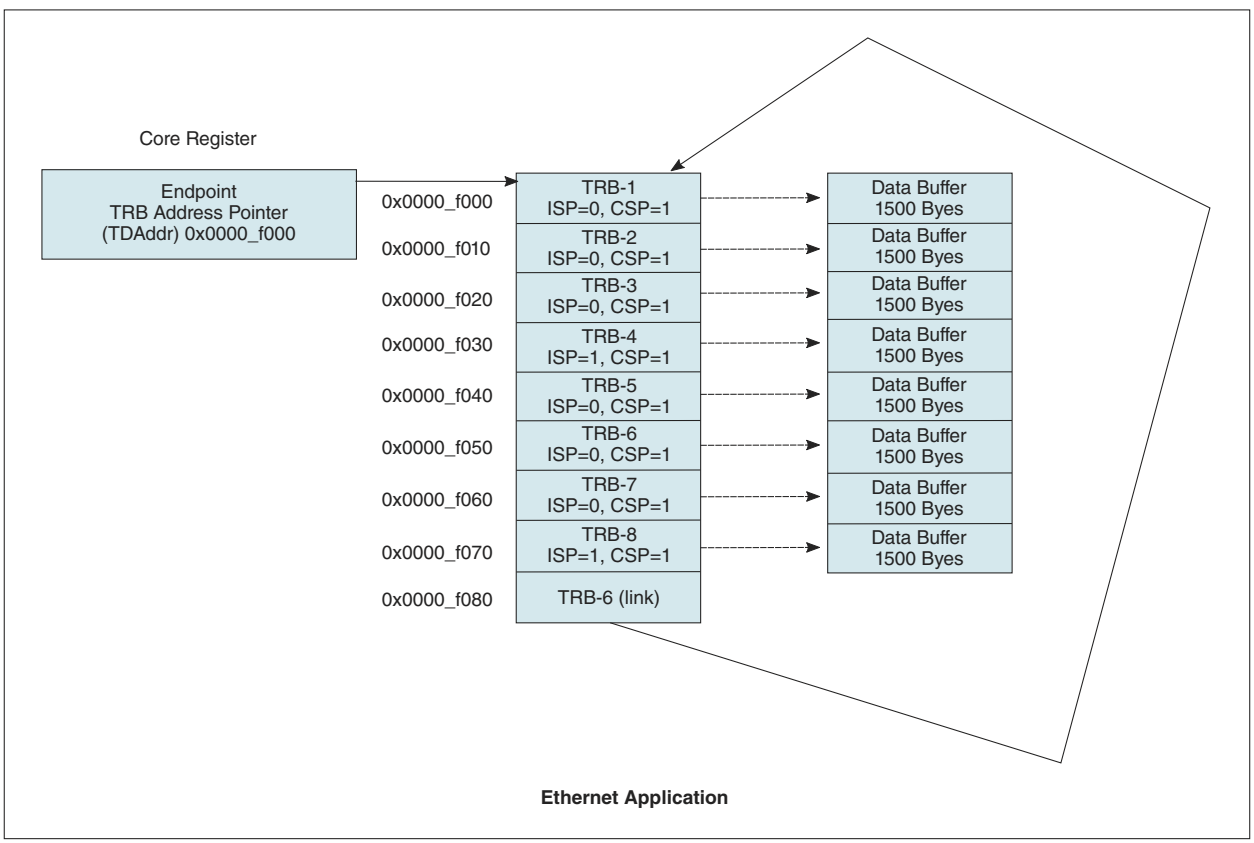


Figure 33-13. ISP and CSP Usage

33.4.2.1.6 Example of Setting Up TRBs

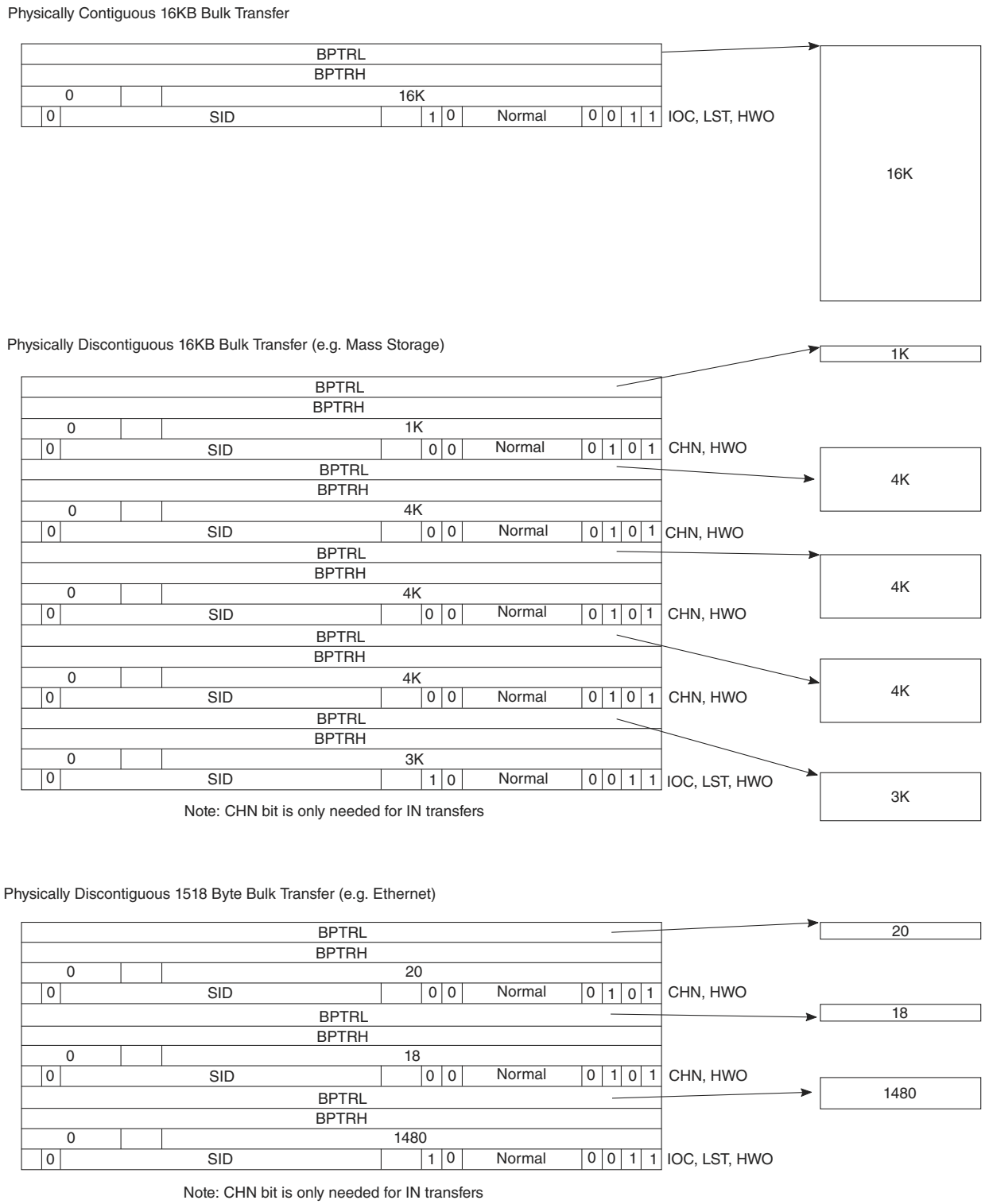


Figure 33-14. Bulk IN TRB Examples

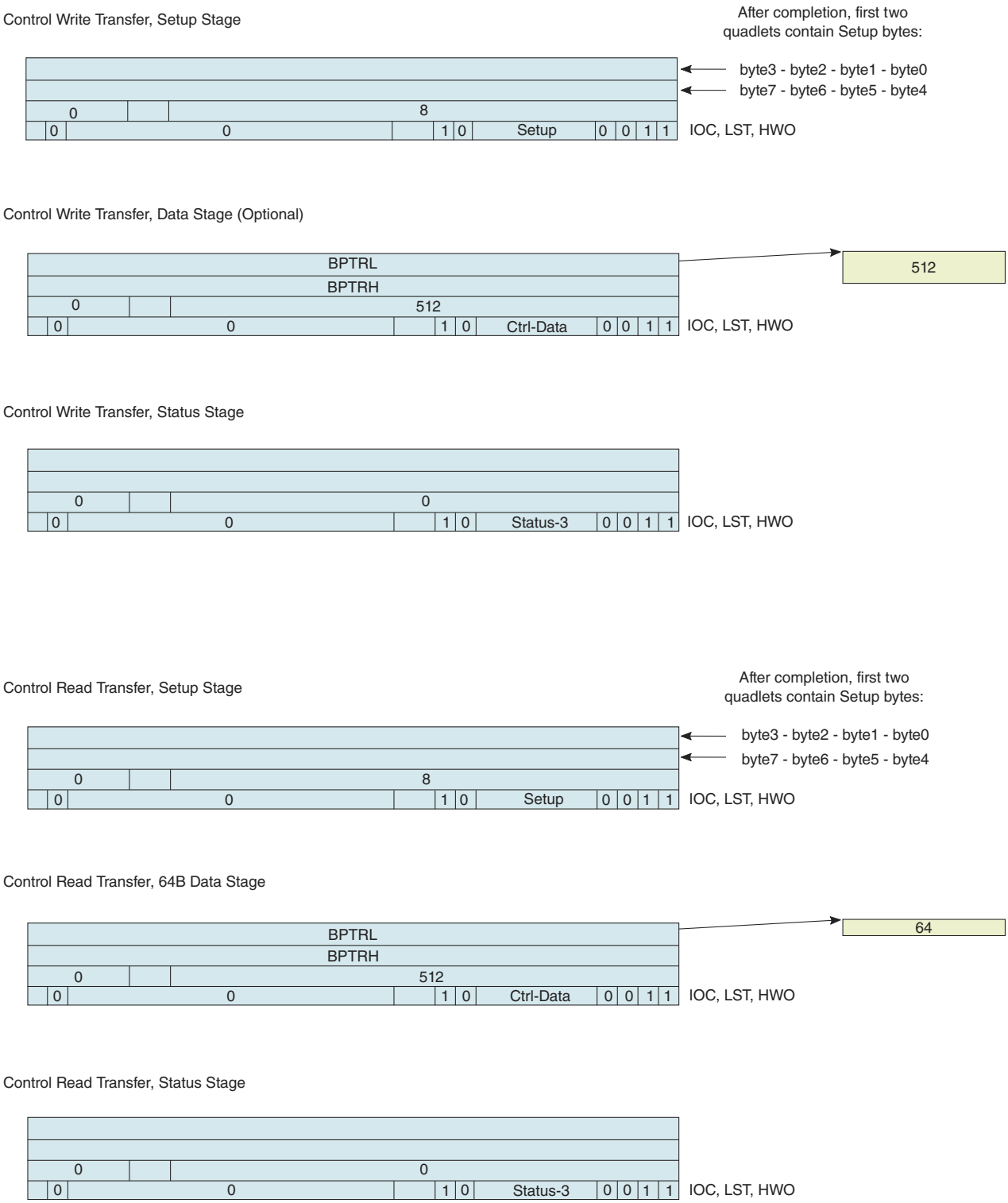


Figure 33-15. Setup/Control/Status TRB Examples

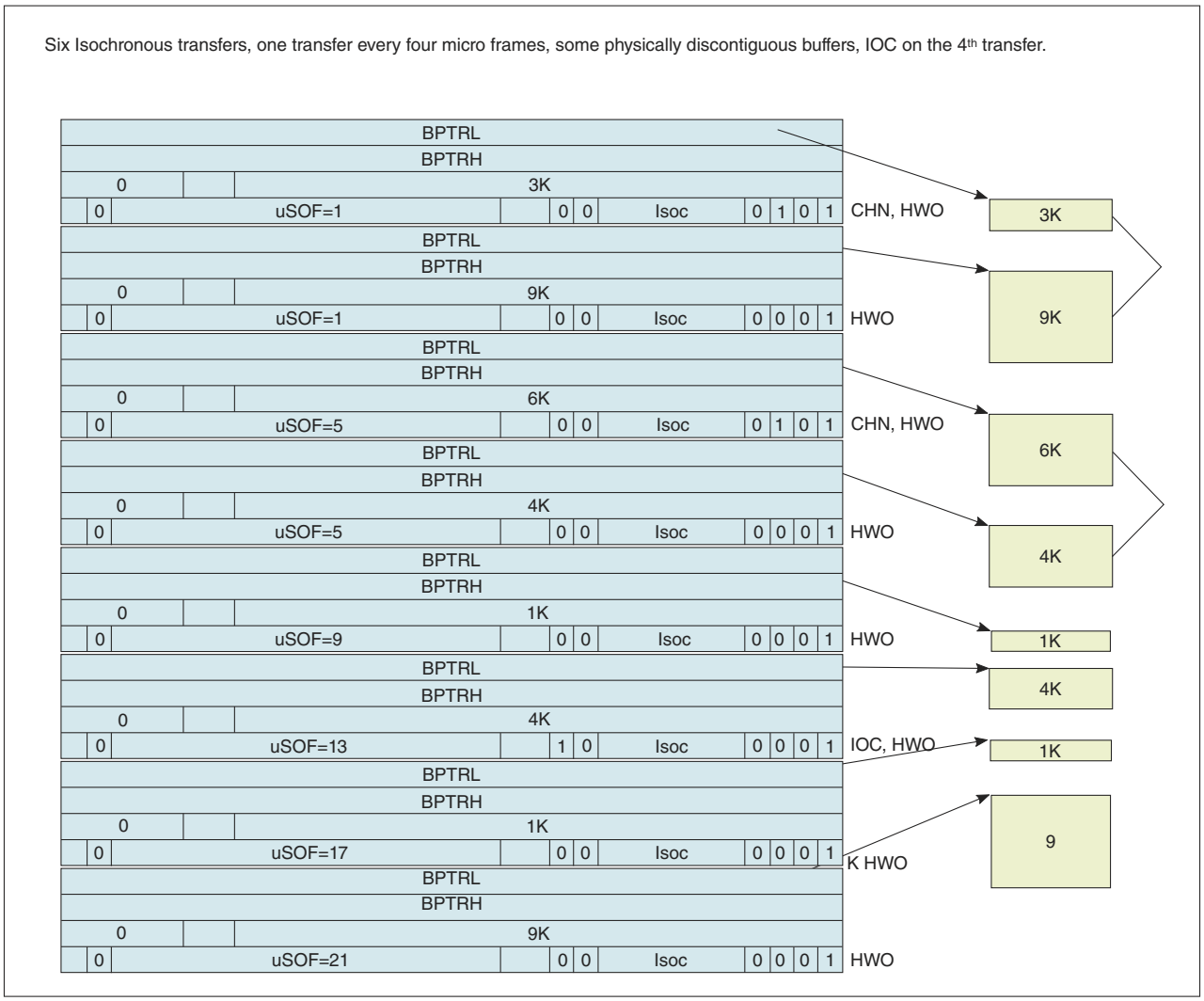


Figure 33-16. Isochronous IN TRB, Example 2

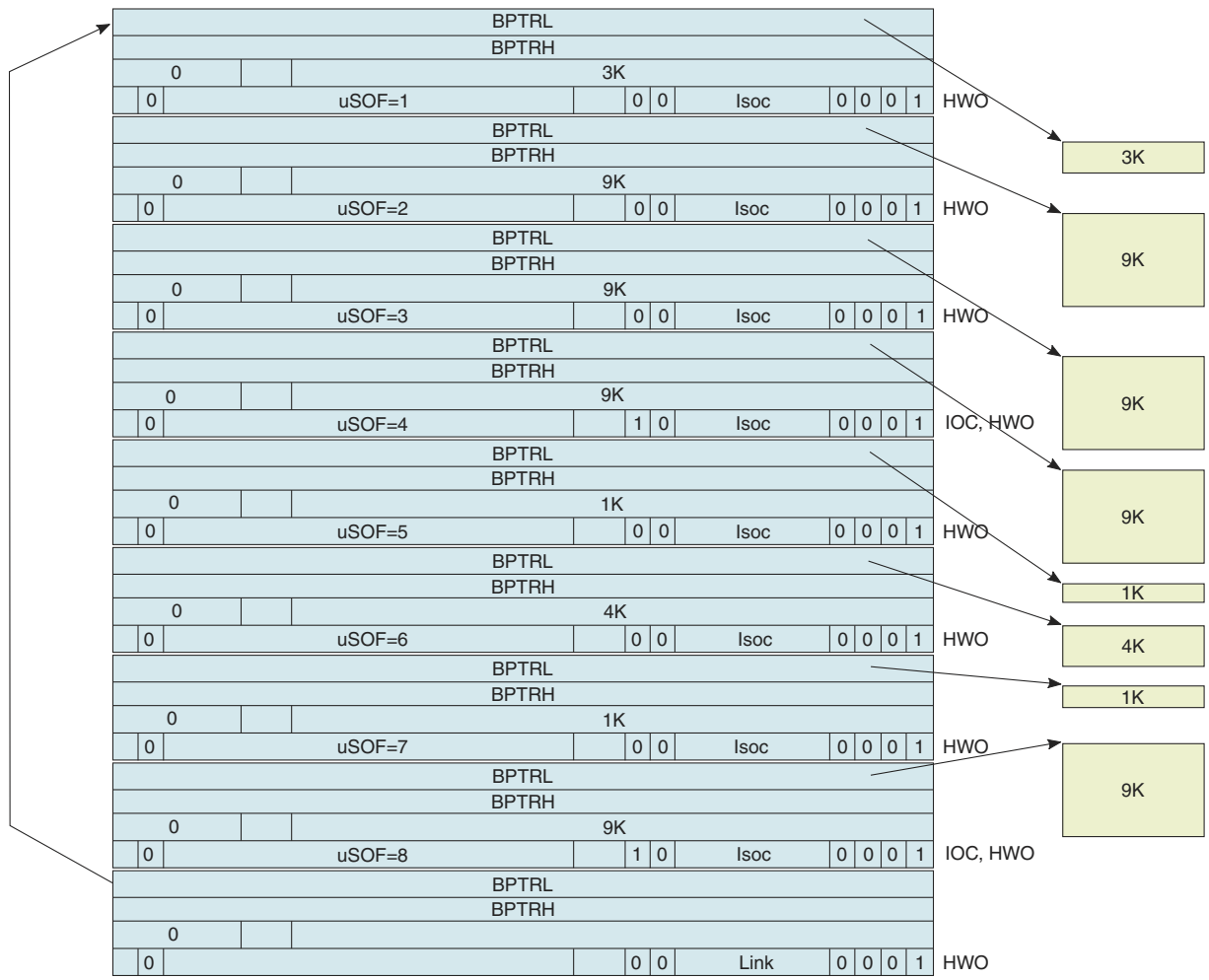


Figure 33-17. Isochronous IN TRB, Example 1

33.4.3 Device Programming Model

33.4.3.1 Register Initialization

The USB3 core contains global registers (prefixed by G*) and device registers (prefixed by D*) that are programmed to start operation and handle certain events. This section describes which registers must be accessed depending on the event that software is attempting to handle:

- Power-On or Soft Reset
- USB Reset Event

- Connect Done Event
- SetAddress Device Request
- SetConfiguration Device Request
- Disconnect Event
- Device-Initiated Disconnect and Reconnect

33.4.3.1.1 Device Power-On or Soft Reset

This section explains device core initialization after power-on or soft reset. The application must follow this initialization sequence for device mode operation.

When the core is first powered on, software initializes the following registers. The order of operations is not important, except for the first and last steps (DCTL[CSFTRST]=1 and DCTL[RUN_STOP]=1).

Table 33-9. Power-On or Soft Reset Register Initialization

Register	Description
DCTL	Set the CSftRst field to '1' and wait for a read to return '0'. This resets the device core.
GSBUSCFG0/1	Leave the default values, refer Global SoC bus configuration register 0 (GSBUSCFG0) and Global SoC bus configuration register 1 (GSBUSCFG1)
GTXTHRCFG/ GRXTHRCFG	This is required only if threshold is enabled.
GUSB2PHYCFG	Program PHY as per Global USB2 PHY configuration register (GUSB2PHYCFG) or leave the default values if the correct power-on values were selected. Note: The PHY must not be enabled for auto-resume in device mode.
GUSB3PIPECTL	Program the following PHY [DatWidth] or leave the default values if the correct power-on values were selected.
GTXFIFOSIZn	Write these registers to allocate prefetch buffers for each Tx endpoint. Unless the packet sizes of the endpoints are application-specific, it is recommended to use the default value.
GRXFIFOSIZ0	Write this register to allocate the receive buffer for all endpoints. Unless the packet sizes of the endpoints are application-specific, it is recommended to use the default value.
GEVNTADR/ GEVNTSIZ/ GEVNTCOUNT	Depending on the number of interrupts allocated, program the Event Buffer Address and Size registers to point to the Event Buffer locations in system memory, the sizes of the buffers, and unmask the interrupt. Note: USB operation stops if the Event Buffer memory is insufficient, because the core stops receiving/ transmitting packets.
GCTL	Program this register to override scaledown, RAM clock select, and clock gating parameters.
DCFG	Program device speed and periodic frame interval
DEVTEN	At a minimum, enable USB Reset, Connection Done, and USB/Link State Change events
DEPCMD0	Issue a DEPSTARTCFG command with DEPEVT.XferRscIdx set to 0 and CmdIOC set to 0 to initialize the transfer resource allocation. Poll CmdAct for completion.
DEPCMD0/ DEPCMD1	Issue a DEPCFG command for physical endpoints 0 & 1 with the following characteristics, and poll CmdAct for completions: USB Endpoint Number = 0 or 1 (for physical endpoint 0 or 1) FIFONum= 0

Table continues on the next page...

Table 33-9. Power-On or Soft Reset Register Initialization (continued)

Register	Description
	XferNRdyEn and XferCmplEn = 1 Maximum Packet Size = 512 Burst Size = 0 EPType = 2'b00 (Control)
DEPCMD0/ DEPCMD1	Issue a DEPXFERCFG command for physical endpoints 0 & 1 with DEPCMDPAR0_0/1 set to 1, and poll CmdAct for completions
DEPCMD0	Prepare a buffer for a setup packet, initialize a setup TRB, and issue a DEPSTRTXFER command for physical endpoint 0, pointing to the setup TRB. Poll CmdAct for completion. Note: The core attempts to fetch the setup TRB via the master interface after this command completes.
DALEPENA	Enable physical endpoints 0 & 1 by writing 0x3 to this register
DCTL	Set DCTL[RunStop] to '1' to allow the device to attach to the host. At this point, the device is ready to receive SOF packets, respond to control transfers on control endpoint 0, and generate events.

After the controller has been started, wait for the following events:

- Wait for a DEVT.USBReset event. This indicates that a reset is detected on the USB.
- Wait for a DEVT.ConnectionDone event. This event indicates the end of the reset on the USB. On this event, read the DSTS register to get the connection speed.

33.4.3.1.2 Initialization on USB reset

To initialize the core as a device, during USB Reset, the application must perform the following steps:

Table 33-10. Initialization on USB reset

Register	Description
DEPCMD0	If a control transfer is still in progress, complete it and get the core into the "Setup a Control-Setup TRB / Start Transfer" state
DEPCMDn	Issue a DEPENDXFER command for any active transfers (except for the default control endpoint 0)
DEPCMDn	Issue a DEPCSTALL (ClearStall) command for any endpoint in STALL mode prior to the USB Reset (excluding control endpoints)
DCFG	Set DevAddr to '0'

NOTE

- Special Reset Considerations for Default Control Endpoint 0: The default control endpoint is not affected by a USB Reset, so software must continue following the software flow for control transfers explained in "Control Transfer Programming Model" even across a USB Reset event.

Resources can only be assigned to the default control endpoint once after a power on or soft reset.

- The USB PHY frequency (SCFG_USB_REFCLK_SELCRn) should be programmed in the PBI phase such that the USB PHY comes out of reset before SYSTEM_READY.

33.4.3.1.3 Initialization on connect done

When this event is received, software must perform the following steps:

Table 33-11. Initialization on connect done

Register	Description
DSTS	Read this register to obtain the connection speed
GCTL	Program the RAMClkSel field to select the correct clock for the RAM clock domain. This field is reset to 0 after USB reset, so it must be reprogrammed each time on Connect Done.
DEPCMD0/ DEPCMD1	Issue a DEPCFG command (with Config Action set to "Modify") for physical endpoints 0 & 1 using the same endpoint characteristics from Power-On Reset, but set MaxPacketSize to 512 (SuperSpeed), 64 (High-Speed), 8/16/32/64 (Full-Speed), or 8 (Low-Speed).
GUSB2CFG/ GUSB3PIPECTL	Depending on the connected speed, write to the other PHY's control register to suspend it
GTXFIFOSIZn	(optional) Based on the new MaxPacketSize of IN endpoint 0, software may choose to re-allocate the TXFIFO sizes by writing to these registers.

33.4.3.1.4 Initialization on SetAddress request

When the application receives a SetAddress request in a SETUP packet, it performs the following steps:

Table 33-12. Initialization on SetAddress request

Register	Description
DCFG	Program the DCFG register with the device address received as part of the SetAddress request when SETUP packet is decoded.
DEPCMD1	After receiving the XferNotReady(Status) event, acknowledge the status stage by issuing a DEPSTRTXFER command pointing to a Status TRB. This step must be done after the DCFG register is programmed with the new device address.

At this point, the device is ready to receive micro-SOF/ITP and is configured to receive control transfers on control endpoint 0 with a new address assigned.

33.4.3.1.5 Initialization on SetConfiguration or SetInterface Request

When the application receives a SetConfiguration or SetInterface request in a SETUP packet, it performs the following steps:

Table 33-13. Initialization on SetConfiguration or SetInterface Request

Register	Description
DALEPENA	Set this register to 0x3 to disable all endpoints other than the default control endpoint 0.
DEPCMDn	Issue a DEPENDXFER command for any active transfers (except for the default control endpoint 0).
DEPCMD1	Issue a DEPCFG command (with Config Action field set to "Modify") for physical endpoint 1 using the current endpoint characteristics to re-initialize the TXFIFO allocation.
DEPCMD0	Issue a DEPSTARTCFG command with DEPCMD0.XferRscIdx set to 2 to re-initialize the transfer resource allocation.
DEPCMDn	Issue a DEPCFG command (with the Config Action field set to "Initialize") for each endpoint that is present in the new configuration (except for the default control endpoint). Note: Control endpoints are bi-directional and must use consecutive even/odd physical endpoint numbers for the OUT/IN direction (such as 2/3, or 4/5), and the FIFONum must be configured to the same value in both directions.
DEPCMDn	Issue a DEPXFERCFG command for each endpoint that is present in the new configuration (except for the default control endpoint 0).
GTXFIFOSIZn	(optional) Based on the new configuration of IN endpoints, software may choose to re-allocate the TXFIFO sizes by writing to these registers.
DALEPENA	Enable the logical endpoints that are active in the new configuration.
DEPCMD1	After receiving the XferNotReady(Status) event, acknowledge the status stage by issuing a DEPSTRTXFER command pointing to a Status TRB. This step must be done after the previous steps to ensure the host does not access any other endpoints that are being set up.

At this point, the core is prepared to accept Start Transfer commands for the newly-configured endpoints. For information on how to set up transfers, see the Operational Model.

33.4.3.1.6 Alternate initialization on SetInterface request

When handling a SetInterface device request, another possibility is that software may reconfigure existing endpoints instead of starting the configuration from the beginning. This is only possible if no TXFIFOs need to be reassigned. This flow also assumes that the endpoints that are being removed or reconfigured in the new interface have halted their traffic prior to the host issuing the SetInterface request.

Table 33-14. Alternate initialization on SetInterface request

Register	Description
DEPCMDn	Make sure there are no transfers still active on the endpoints that are changing in the new interface. This is normally ensured by the host prior to issuing the SetInterface, but if not, issue an End Transfer command for the transfer on each endpoint that is changing.

Table continues on the next page...

Table 33-14. Alternate initialization on SetInterface request (continued)

DEPCMDn	Issue a DEPCFG command (with the Config Action field set to "Initialize") for each endpoint that is changing in the new configuration. The side effect of the DEPCFG command is that the endpoint's sequence number is automatically set to 0.
DALEPENA	Write this register with all the endpoints that are enabled (including the ones that are not changing).
DEPCMDn	Using the StartXfer command, acknowledge the status stage response for the SetInterface request.

33.4.3.1.7 Initialization on Disconnect Event

When the application receives a Disconnect event, it must set DCTL[ULSTCHNGREQ] to 5. Other than this, the core does not require any initialization. Because the DCTL[RUN_STOP] bit is still '1', the device attempts to reconnect to the host, at which time a USB Reset and Connect Done event occurs. However, if the application does not want to attempt to reconnect to the host, it should perform the steps in the next section.

NOTE

When DCFG[DEVSPD] is programmed for 2.0 only mode (such as, High-Speed or Full-Speed), and if the application wants to issue any commands to clear any pending transfers during a Disconnect interrupt, then it has to disable USB_GUSB2PHYCFG[SUSPENDUSB20] before issuing any commands and re-enable it after the commands have completed.

33.4.3.1.8 Device-initiated disconnect

If the application wants to disconnect from the host, it should perform the following actions:

Table 33-15. Initialization on device-initiated disconnect

Register	Description
DEPCMD0	If a control transfer is still in progress, complete it and get the core into the "Setup a Control-Setup TRB / Start Transfer" state
DEPCMDn	Issue a DEPENDXFER command for any active transfers (except for the default control endpoint 0)
DCTL	Set DCTL[RUN_STOP] to '0' to disconnect from the host
DSTS	Poll [DEVCTRLHLT] until it is '1'

At this point, the device is disconnected from the host and will not attempt to reconnect.

33.4.3.1.9 Reconnect after Device-Initiated Disconnect

If the application decides to reconnect to the host, it must follow the steps in Device Power-On or Soft Reset.

33.4.3.2 Operational Model

The following sections describes the processes and data structures that the core uses to implement the USB 3.0 specification.

33.4.3.2.1 USB and Physical Endpoints

Endpoints are referred to in two ways:

- As a USB endpoint number: USB endpoints are defined in the USB specification.
- As a physical endpoint resource number: The hardware has a fixed number of physical endpoint resources. Each resource is unidirectional and can be configured to refer to either direction of any USB endpoint.

The software always works on physical endpoints and it knows how physical endpoint corresponds to USB endpoints. DALEPENA is the only register that has one enable bit per USB endpoint.

During SetConfiguration, software maps physical endpoint resources to the required USB endpoints. When a USB request comes in, the USB endpoint number gets converted to a physical endpoint number in the core. Similarly, when a USB packet is sent out, the physical endpoint number is converted to the USB endpoint and sent out.

A USB control endpoint requires two physical endpoints. One physical endpoint is mapped to the OUT direction of the Control endpoint, and the other one is mapped to the IN direction of the Control endpoint. Specifically the two physical endpoints are used as the following:

- The Setup stage of any control transfer uses the OUT direction physical endpoint.
- For a control write or 2-stage transfer, the Data stage (if present) uses the OUT direction physical endpoint and the Status stage uses the IN direction physical endpoint.
- For a control read transfer, the Data stage uses the IN direction physical endpoint and the Status stage uses the OUT direction physical endpoint.

33.4.3.2.2 Event Buffers

Hardware passes command completion, transfer progress, and asynchronous events to software through one or more Event Buffers.

To configure an Event Buffer, the software performs the following steps:

1. Sets up an empty buffer in system memory.
2. Writes the address of the beginning of the buffer into GEVNTADR. This address must be aligned to the Event Buffer size.
3. Writes the size of the buffer and interrupt mask into GEVNTSIZ. Depending on your system interrupt latency, enough Event Buffer space must be allocated to avoid lost interrupts or reduced performance.
4. Write a 0 into the GEVNTCOUNT register. This must be the last step, as it enables the Event Buffer. After the Event Buffer has been configured, software must not change the size or address.

There is one interrupt line per Event Buffer that indicates there are one or more events present. Software reads one or more events out of the buffer and indicates to hardware how many events it processed by writing the byte count to the GEVNTCOUNT register.

Clock crossing delays may result in the interrupt's continual assertion after software acknowledges the last event. Therefore, when the interrupt line is asserted, software must read the GEVNTCOUNT register and only process events if the GEVNTCOUNT is greater than 0.

The first event produced by the core after the Event Buffer is configured will be written to the address specified in GEVNTADR. Most events are 32 bits, and subsequent events will be written to the address (PreviousEventAddress + 4). When that address exceeds the sum of GEVNTADR and GEVNTSIZ, the core wraps around to the first GEVNTADR value. In this way, the Event Buffer operates like a circular buffer with hardware writing to the "tail" of the buffer and software reading from the "head."

Most events are exactly 4 bytes in size, but there is one exception: The Vendor Device Test LMP Received Event (VndrDevTstRcved) is a 12 byte event that includes a header in the first 4 bytes and the contents of the LMP in the following 8 bytes.

When an event occurs within the core, hardware checks the enable bit that corresponds to the event to decide whether the event will be written to the Event Buffer or not. The Event Buffer contains one of the following types of information, depending on the value of the lower bits of the event:

- Device Endpoint-Specific Event (DEPEVT) (Event[0] = 0x0)
  - The DEPCFG endpoint-specific command specifies the bits that are enabled and which Event Buffer to use for these events.
- Device-Specific Event (DEVT) (Event[0] = 0x1, Event[7:1] = 0x00)

- The Generic Command Complete event is enabled through the DGCMD[CmdIOC] field when the command is issued.
- The Event Buffer Overflow Event cannot be disabled and is written to the Event Buffer that encounters the overflow.
- The rest of the Device-Specific events are enabled through the DEVTEN register.
- Except for the Event Buffer Overflow Event, these events are written to the Event Buffer specified in the DCFG[IntrNum] field.

The core always leaves one entry free in each Event Buffer. When the Event Buffer is almost full, hardware writes the Event Buffer Overflow event and the USB will eventually get stalled when endpoints start responding NRDY or the link layer will stop returning credits (in SuperSpeed). This event is an indication to software that it is not processing events quickly enough. During this time, events will be queued up internally. When software frees up Event Buffer space, the queued up events will be written out and the USB will return to normal operation.

Event Buffer Content for Device Endpoint-Specific Events (DEPEVT)

Table 33-16. Device Endpoint-n Events: DEPEVT

Field	Description
31–16	Event Parameters (EventParam) For XferNotReady, XferComplete, and Stream events on Bulk Endpoints: [31-16]StreamID. Applies only to bulk endpoints that support streams. This indicates the Stream ID of the transfer for which the event is generated For XferInProgress: [31-16] Isochronous Microframe Number (IsocMicroFrameNum). Indicates the microframe number of the beginning of the interval that generated the XferInProgress event (debug purposes only) For XferNotReady events on Isochronous Endpoints: [31-16] Isochronous Microframe Number (IsocMicroFrameNum). Indicates the microframe number during which the endpoint was not ready **NOTE:** USB 3.0 core represents USB bus time as a 14-bit value on the bus and also in the DSTS register (DSTS[SOFFN]), but as a 16-bit value in the XferNotReady event. Use the 16-bit value to interact with Isochronous endpoints via the StartXfer command. The extra two bits that the USB 3.0 core produces will be necessary for handling wrap-around conditions in the interaction between software and hardware. EPCmdCmplt events For all EPCmdCmplt events [27-24]- Command Type. The command type that completed (Valid only in a DEPEVT event. Undefined when read from the DEPCMD.EventParam field). For EPCmdCmplt event in response to Start Transfer command. [22-16] Transfer Resource Index (XferRscIdx). The internal hardware transfer resource index assigned to this transfer. This index must be used in all Update Transfer and End Transfer commands.
15–12	Event Status (EventStatus)

Table continues on the next page...

Table 33-16. Device Endpoint-n Events: DEPEVT (continued)

Field	Description
	Within an XferNotReady event- [15]: Indicates the reason why the XferNotReady event is generated: 0- XferNotActive- Host initiated a transfer, but the requested transfer is not present in the hardware 1- XferActive- Host initiated a transfer, the transfer is present, but no valid TRBs are available [14]: Not Used [13:12]: For control endpoints, indicates what stage was requested when the transfer was not ready: 2'b01- Control Data Request 2'b10- Control Status Request Within an XferComplete or XferInProgress event- [15]: LST bit of the completed TRB (XferComplete only) [15]: MissedIsoc: Indicates the interval did not complete successfully (XferInProgress only) [14]: IOC bit of the TRB that completed [13]: Indicates the TRB completed with a short packet reception or the last packet of an isochronous interval [12]: Reserved If the host aborts the data stage of a control transfer, software may receive a XferComplete event with the EventStatus field equal to '0'. This is a valid event that must be processed as a part of the Control transfer programming model. Within a Stream Event- [15:12]: - 4'h2- StreamNotFound- This stream event is issued when the stream-capable endpoint performed a search in its transfer resource cache, but could not find an active and ready stream. - 4'h1- StreamFound- This stream event is issued when the stream-capable endpoint found an active and ready stream in its transfer resource cache, and initiated traffic for that stream to the host. The ID of the selected Stream is in the EventParam field. In response to a Start Transfer command- 4'h2- Indicates expiry of the bus time reflected in the Start Transfer command. 4'h1- Indicates there is no transfer resource available on the endpoint. In response to a Set Transfer Resource (DEPXFERCFG) command- 4'h1- Indicates an error has occurred because software is requesting more transfer resources to be assigned than have been configured in the hardware. In response to a End Transfer command- 4'h1- Indicates an invalid transfer resource was specified.
11–10	This field is reserved.
9–6	4'h7- Endpoint Command Complete (EPCmdCmplt) Indicates software may issue another Device Endpoint command to the endpoint. When issued in response to an End Transfer command, indicates that DMA stopped for the endpoint. For all other commands, this event does not imply that all effects of the commandtook place. The DEPCMD register contains the same status information present in the Event Status Bits field. 4'h6- Stream Event (StreamEvt)

Table continues on the next page...

Table 33-16. Device Endpoint-n Events: DEPEVT (continued)

Field	Description
	Indicates that a stream-capable endpoint initiated a search within its transfer resource cache. The result of the search is in the EventStatus field (Found or NotFound). 4'h5- Reserved 4'h4 - Reserved 4'h3- XferNotReady Event (XferNotReady) Indicates receipt of a transaction when no TRBs are available for the endpoint. For isochronous IN endpoints, a zero-length packet is automatically sent by hardware and NRDY for non-isochronous endpoints. The application must enable this event if it plans to issue Start Transfer on demand. This event can happen when software issues a Start Transfer or Update Transfer. In this case, software must ignore this event because it has already issued the Start Transfer or Update Transfer. XferNotReady is generated when the core responds NRDY to the host on the USB. It is useful in the beginning of a transfer when software wants to wait for the host to start polling the endpoint before setting up TRBs, but it is not efficient to use this event to determine when the core has run out of TRBs. For determining when the core has processed TRBs and needs software to setup more, use the IOC field in a TRB along with the XferInProgress event that is generated when the core completes a TRB with IOC=1. For non-stream-capable endpoints, the hardware filters multiple events if the host continues transactions, even after the core responds with NRDY. This internal filter is reset after software issues any endpoint command to this endpoint. For stream-capable endpoints, this event is generated each time the host attempts a transaction, even if the core responds with NRDY. For isochronous endpoints, this event is generated only once prior to the Start Transfer command to communicate the current bus time. For additional information, see the Event Status field. 4'h2- XferInProgress Event (XferInProgress) Applies to IN and OUT endpoints. Indicates an EP/Stream specific event happened and it is continuing the transfer. For additional information, see the Device Programming Model. 4'h1- XferComplete Event (XferComplete) Applies to IN and OUT endpoints. Indicates that a EP/Stream transfer completed and the core stopped the transfer. For additional information, see the Device Programming Model. 4'h0- Reserved
5–1	Physical Endpoint Number (0-31)
0	1'b0- Indicates that this is an endpoint-specific event.

Event Buffer Content for Device-Specific Events (DEVT)

Table 33-17. Device-specific events: DEVT

Field	Description
31-25	Reserved
24-16	Event Information Bits (EvtInfo)

Table continues on the next page...

Table 33-17. Device-specific events: DEVT (continued)

Field	Description
	For a USB/Link State change Event, this field indicates the state of the link- EvtInfo[4] - SuperSpeed event. Set to 1 for SS; Set to 0 for non-SS. EvtInfo[3:0] - Link State. Indicates link state at the time of the event. Follows same encoding as DSTS link state bits.
15-12	Reserved
11-8	11- Event Buffer Overflow Event (EvntOverflow) The core writes this event to indicate there is no space in the event buffer, and one or more Device-specific events may have been dropped after this event. Endpoint-Specific events will not be dropped, but they will be delayed which can cause a drop in performance on the USB. Software uses this event as a warning that the Event Buffer is too small and the core should be reconfigured with a larger Event Buffer or software needs to process events more quickly. In order to avoid repeated Event Buffer Overflow Events, software must free up space in the Event Buffer by acknowledging more than 1 event (writing a value greater than 4 to the GEVNTCOUNT register). This event cannot be disabled, and is always written to the Event Buffer that encounters the overflow. 10- Generic Command Complete Event (CmdCmplt) The core writes this event to indicate the generic command is complete. 9- Erratic Error Event (ErrticErr) The core writes this event to report erratic errors (phy_rxvalid_i/phy_rxvldh_i or phy_rxactive_i is asserted for at least 2 ms, due to PHY error) seen on the UTMI+. Due to erratic errors, the USB 3.0 core goes into Suspended state and a USB/Link state change event (ULStChng) is generated to the application. If the early suspend is asserted due to an erratic error, the application can only perform a soft disconnect recover. For SuperSpeed operation, the PHY erratic error event indicates that the PIPE is not responding to the PHY command (for example, pipe3_PowerDown change or receiver detection). After the core requested any PHY command, if it did not receive pipe3_PhyStatus within 100ms, the core will timeout and generate the erratic error event. After generating the erratic error, the core assumes that the PHY command is completed successfully and proceeds with the next pending PHY command. Software must reset the controller on receiving the erratic error. 8- Reserved 7- Start of (micro)Frame (Sof) 6- USB Suspend Entry Event This event provides a notification that the link has gone to a suspend state (L2, U3, or L1). The existing Link State Change event (3) provides the same information, but is generated for every link state change. 5- Reserved 4- Resume/Remote Wakeup Detected Event (WkUpEvt) This event is generated when the host initiates a resume condition on the USB bus. This event is not generated by the device initiating a remote wakeup to the host. 3- USB/Link State Change (ULStChng) The core writes this event to indicate a change of USB or Link state. Bits 23-16 (EvtInfo) of this event indicate the Link Status. The event is generated in SuperSpeed when the link LTSSM state changes, except when LTSSM exits from HOT_RESET, POLL or LTSSM enters into RECOVERY. 2- Connection Done (ConnectDone)

Table continues on the next page...

Table 33-17. Device-specific events: DEVT (continued)

Field	Description
	1- USB Reset (USBRst) 0- Disconnect Detected Event (DisconnEvt) Note: This event is generated only if VBUS is off.
7-1	7'h00 - Device Specific Event
0	1'b1 - Non-Endpoint-Specific Event

33.4.3.2.3 Transfer and Buffer Rules

Buffers that are used to transfer data to and from an endpoint are defined using a Buffer Descriptor, which consists of one or more Transfer Request Blocks (TRBs). A CHN flag in the TRB is used to identify the TRBs that comprise a Buffer Descriptor. Therefore, a Buffer Descriptor refers to a consecutive set of TRB data structures where the CHN flag is set in all TRBs, except the last. Note that a Buffer Descriptor may consist of a single TRB, whose CHN flag will not be set.

Software communicates the location of the first TRB by using the Start Transfer command on an endpoint. Even endpoints that are not stream-capable use this command.The core fetches TRBs from external memory starting at the address provided in the Start Transfer command parameters, continuing in a linear fashion and following the Link TRB until a TRB is encountered that has its LST bit set to '1' or HWO bit set to '0'. Therefore, software must ensure that it has valid TRBs prepared before issuing the Start Transfer command to prevent the core from reading uninitialized memory.

Descriptor fetch requests are buffered within the hardware and handled separately from the progress of the current transfer. Therefore, it is possible that the core completes a transfer with XferComplete but still continues reading TRBs that it has not cached yet. If software is immediately de-allocating the memory for TRBs based on the XferComplete event, it is recommended that software issue an End Transfer command for the endpoint/ transfer resource prior to de-allocating the memory. The completion of the End Transfer command flushes out any pipelined descriptor fetches and avoids a potential bus error.

While processing TRBs, two conditions may cause the core to write out an event and raise an interrupt line:

- TRB Complete:
  - For OUT endpoints, a packet is received which reduces the remaining byte count in the TRB buffer to zero.
  - For IN endpoints, an acknowledgement is received for a transmitted packet which reduces the remaining byte count in the TRB buffer to zero.
- Short Packet Received:

- For OUT endpoints only. While writing to a TRB buffer, the endpoint receives a packet that is smaller than the endpoint's MaxPacketSize.

This table shows the USB 3.0 Core Actions Based on TRB Control Bits.

Table 33-18. USB 3.0 Core Actions Based on TRB Control Bits

Direction	TRB Complete Packet	ISP	IOC	CHN	LST	CSP	Action
IN	Yes	-	X	0	1	-	XferComplete event
IN	Yes	-	0	X	0	-	No event
IN	Yes	-	1	X	0	-	XferInProgress event
OUT	Yes No	-	X	0	1	-	XferComplete event
OUT	Yes No	-	0	X	0	-	No event
OUT	Yes No	-	1	X	0	-	XferInProgress event
OUT	X Yes	X	X	X	X	0	XferComplete event [1]
OUT	X Yes	X	X	0	1	1	XferComplete event
OUT	X Yes	X	X	1	0	1	Search for CHN=0, accumulate IOC and ISP, then follow CHN=0 rules [2]
OUT	X Yes	X	1	0	0	1	XferInProgress event
OUT	X Yes	0	0	0	0	1	No event
OUT	X Yes	1	0	0	0	1	XferInProgress event

1. When a TRB receives whose CSP bit is 0 and CHN bit is 1 receives a short packet, the chained TRBs that follow it are not written back (for example, the BUFSIZ and HWO fields remain the same as the software-prepared value)
2. In the case of an OUT endpoint, if the CHN bit is set (and CSP is also set), and a short packet is received, the core retires the TRB in progress and skip past the TRB where CHN=0, accumulating the ISP and IOC bits from each TRB. If ISP or IOC is set in any TRB, the core generates an XferInProgress event. Hardware does not set the HWO bit to 0 in skipped TRBs. If the endpoint type is isochronous, the CHN=0 TRB will also be retired and its buffer size field updated with the total number of bytes remaining in the BD.

On XferComplete and XferInProgress events, status bits in the event indicate LST, IOC, Short Packet Received, or Bus Error status. In the "fast-forward" case, the IOC status is accumulated from the skipped TRBs, not just the TRB that received the short packet.

When the hardware writes back the TRBs, it updates the BUFSIZ field to represent the remaining unused buffer.

33.4.3.2.3.1 Number of TRBs Rule

- Software must set up only one TRB for a control setup or status stage.
- If software is preparing multiple transfers for an IN endpoint, it may be necessary to place a 0-length TRB between transfers that are MaxPacketSize aligned to indicate

transfer boundaries to the host. This is not necessary, if the class driver and the host can handle bursting between transfers.
- If software is preparing multiple transfers for an OUT endpoint, it needs to place a MaxPacketSize TRB between transfers, if it expects the host to transmit a 0-length packet between transfers.

33.4.3.2.3.2 TRB Control Bit Rules

Transfer control bits must conform to the following restrictions:

For OUT endpoints, the CSP bit must be the same in every TRB within a Buffer Descriptor (either set or clear).

- The core autonomously checks the HWO field of a TRB to determine if the entire TRB is valid.
- Therefore, software must ensure that the rest of the TRB is valid before setting the HWO field to '1'. In most systems, this means that software must update the fourth DWORD of a TRB last. Every time software validates a TRB by setting HWO=1, it must also issue an Update Transfer command to the core.
- Software sets the HWO bit to 1 when it creates the TRB, and cannot modify it until hardware resets it to 0. However,
  - Software must detect when a "fast-forward" occurs on an OUT endpoint that receives a short packet, since some TRBs in a chain may still have their HWO bit set to 1 while belonging to software.
  - Hardware will not clear the HWO bit of a Link TRB. Therefore, software can only modify a Link TRB if the TRB prior to the Link TRB has its HWO bit set to 0.
- The LST bit must not be set to 1 for isochronous endpoints.
- For a Setup or Status TRB, set CHN=0, LST=1, and CSP=0.
- For the data stage of a control transfer, set CSP=0 in all TRBs and LST=1 in the last TRB of the data stage.
- For Link TRBs, the LST, CHN, IOC, ISP, CSP, and Stream ID fields are ignored and must be set to 0.
- The Link TRB's chain bit is implicitly equal to the chain bit of the TRB before it. If the TRB before it has CHN=1, then the Link acts as if it's CHN=1. If the TRB before it has CHN=0, then the Link acts as if it's CHN=0. A Link TRB cannot be the last TRB in a Buffer Descriptor.
- When CSP=1 and the core receives a short packet, it searches for the end of the Buffer Descriptor by finding the Normal TRB that has CHN=0. Therefore, it is illegal to setup a circular buffer with all Normal TRBs with CHN=1.

33.4.3.2.3.3 Buffer size rules and zero-length packets

The hardware contains a cache that holds four TRBs per transfer.

For IN endpoints, the following rules apply:

- The number of chained TRBs necessary to construct a single packet must never exceed 3 . A maximum of one link TRB can be present in the chain.
- If software wants to indicate a transfer completion to the host by sending a zero-length packet after a multiple of MAXPACKETSIZE, it must set up a zero-length TRB following the last TRB in the transfer.

For OUT endpoints, the following rules apply:

- If the first TRB has CHN=1, its buffer size must be ≥ 1 byte
- The total size of a buffer descriptor must be a multiple of MAXPACKETSIZE
- A received zero-length packet still requires a MAXPACKETSIZE buffer. Therefore, if the expected amount of data to be received is a multiple of MAXPACKETSIZE, software should add MAXPACKETSIZE bytes to the buffer to sink a possible zero-length packet at the end of the transfer.

For IN and OUT endpoints, the following rule applies:

- The BUFSIZ field in a link TRB must be set to 0.

33.4.3.2.3.4 Transfer setup recommendations

Software can either set up transfers before the host attempts to move data on an endpoint ("preset" transfers) or can set up transfers on demand ("on-demand" transfers). When using preset transfers, software can safely disable the XferNotReady event in the endpoint configuration. However, when using on-demand transfers, the XferNotReady event must be enabled and software may not use the "No Response" variant of the Update Transfer command. The XferNotReady event is issued when the host attempts to move data on an endpoint when one of the following conditions is present:

- No previous transfer was started with Start Transfer.
- Not enough hardware-owned (HWO=1) TRBs are available to handle the requested data movement. The XferNotReady event must not be disabled for control endpoints because the event is an integral part ofcontrol transfer handling.

Although there are many valid ways to set up transfers, it is recommended that you choose one of three general mechanisms:

- When software wants to set up one transfer at a time and has the entire buffer available for transfer, it must set up TRBs that point to the data buffers and in the last

TRB it must set the LST bit and issue Start Transfer, which points to the first TRB location.
    - When the USB transfer completes, the core will notify the software through the XferComplete event. The LST bit will be set in the status field of the event. The XferComplete will also release the Transfer Resource.
    - A premature XferComplete event can happen before all the data buffers are exhausted, if there is a Bus Error or, in an OUT transfer a short packet has been received and CSP=0. During these conditions the TRB will have an updated BUFSIZ field which represents the amount of buffer remaining after the successful part of data transfer.
    - Software can also set up an IOC bit TRB, so that it gets notified when data buffers are used up and can free them up sooner than waiting for the entire transfer to complete. On completing a TRB with IOC set, and not the last one, the core will issue an XferInProgress event with IOC set in the status field of the event.
- When software wants to set up one transfer at a time, but it has fewer data buffers available than the full transfer size, it must set up circular TRBs (using a Link TRB), and also set up IOC bits in the TRBs. Depending on buffer allocation and interrupt frequency it can set IOC for once in "x" number of TRBs.
    - As soon as the USB transfer for a TRB is completed, and if IOC is set, the core will generate an XferInProgress event with IOC set in the status field.
    - On seeing the event, software reuses the TRB and updates it with the next data buffer. It also issues an Update Transfer command, indicating it has updated a TRB.
    - If software is slow and hardware finds a TRB with HWO reset to 0, it waits for an Update Transfer command. Upon seeing the Update Transfer, it prefetches the TRB and continues the transfer.
    - When software reaches the end of the transfer, it sets the TRB LST bit. When hardware completes the TRB, it issues an XferComplete event and releases the Transfer Resource.
- When the Device software has multiple transfers to set up, it must set up circular TRBs and also set up CHN bits in all TRBs, except the last of each transfer. For OUT endpoints, multiple transfers can be supported when the CSP field is set to '0' if each transfer has a multiple of MaxPacketSize bytes, since a short packet will end the transfer. If multiple transfers may contain short packets, the CSP field must be set to '1' to enable the next transfer to continue even if a short packet is received. Depending on buffer allocation and interrupt frequency, it can either set the IOC for once in "x" number of TRBs or in the last TRB of each transfer.
    - For OUT endpoints, the transfer can finish prematurely due to a short packet from the host. In this case, the core processes the remaining TRBs, skipping the updates to these TRBs until it reaches TRB of the next transfer. While skipping

these TRBs, if any of the TRB it encounters the interrupt setting of ISP or IOC, it will generate the corresponding event once and ignore the interrupt settings of the remaining TRBs until a TRB of next transfer is reached.
- Device software can reclaim these skipped TRBs even though the HWO still indicates 1 (hardware-owned).
- For IN endpoints, software cannot stop providing transfers while it is ending transfers with CHN=0 and LST=0, otherwise it is possible that the endpoint is left in a flow-controlled state on the USB.

33.4.3.2.4 Transfer Resource Usage and Transfer State

Software allocates one Transfer Resource for an endpoint during initialization. When software issues a Start Transfer command, this Transfer Resource is used. When an XferComplete event happens, the Transfer Resource is released back to software. Similarly, when End Transfer completes, the Transfer Resource is released.

Following is a typical transfer usage summary (not complete usage model):

- Software sets up data buffer(s) and TRB(s) in external memory with the TRB(s) pointing to the buffer(s).
- Software issues Start Transfer with the pointer to the first TRB in the command parameters.
- If software sets up all the buffers and TRBs at the same time (Host and Device Software negotiate the transfer size), then just one Start Transfer is enough, and software waits for a XferComplete event.
- If software did not set up all the TRBs, then every time software adds a new TRB (by setting HWO=1) it must issue an Update Transfer command. The hardware will fetch the TRB again and continue the transfer.
- The End Transfer command is used only during error conditions and not used during normal transfers. For example, if software has set up multiple transfers and if a USB Reset event happens, it must remove all the transfers from the queue using End Transfer with the ForceRM bit set 1.
- A transfer is completed when all the data has been transferred or a short packet has been received. If software is queuing in only single transfers at a time (normal method for mass storage - for ethernet over USB, software can set up multiple transfers using the Chain bit and Continue on Short Packet bit), once XferComplete interrupt has happened, the transfer is completed. When starting a new transfer, software now needs to issue the Start Transfer command.
- An OUT transfer's transfer size (Total TRB buffer allocation) must be a multiple of MaxPacketSize even if software is expecting a fixed non-multiple of MaxPacketSize transfer from the Host.

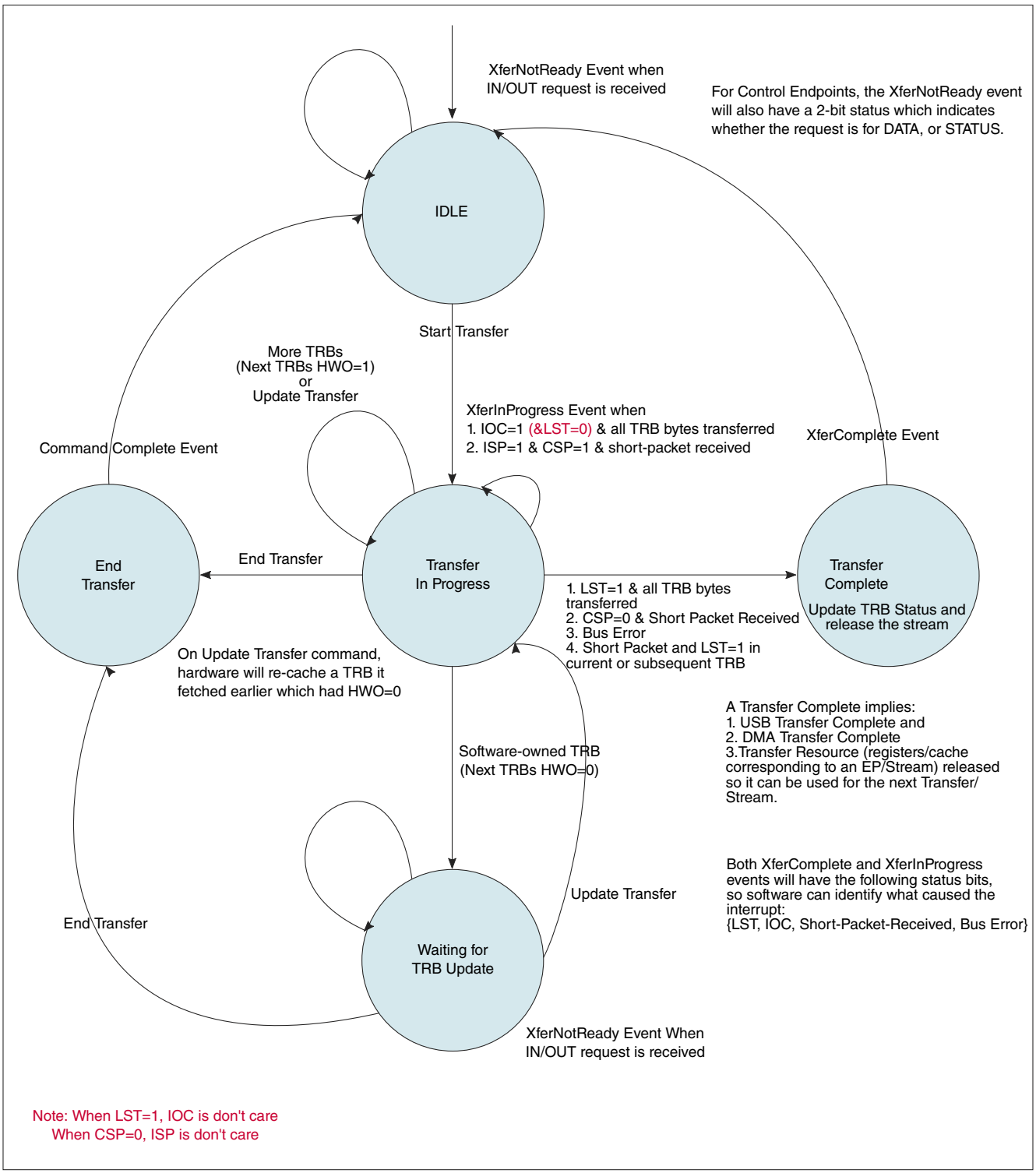


Figure 33-18. Transfer Usage State Machine.

33.4.3.2.5 Transfer Descriptions

The following sections describe typical transfer processes.

33.4.3.2.5.1 Non-Isochronous OUT Transfers

The sequence below describes a regular, non-isochronous OUT transfer (control-data/ bulk/interrupt).

Table 33-19. Non-Isochronous OUT Transfer Sequence

Steps	Host	Device Core	Device Software
Software sets up data buffers before a host request			
Step 1	-	-	Software sets up Normal TRBs and enables DMA by issuing Start Transfer.
Step 2	-	Device fetches the TRB and caches it.	-
Step 3	Host sends data packets	-	-
Step 4	-	Core receives the packets in RXFIFO and stores them in to TRB data buffers through DMAs. While the RXFIFO has space and the BUS DMA keeps up with the host data transfers, the device receives all the packets. Once all the data is received, the core updates the TRB status and then generates a XferComplete event. In SS mode, if GRXTHRCFG.USBRxPktCntSel is set to, • 0: the NumP sent depends on the DCFG.NumP and bMaxBurstSize • 1: the NumP sent depends on the RXFIFO space available when the packet is received When DMA is slow or CRC fails: • If the RXFIFO overruns or CRC error is detected, device times out in Non-SS mode or asks for the same data packet (through ACK/SeqNumber/Retry Bit) in SS mode. • In HS mode, on NAK the host will issue PING and device will ACK it (if there is enough space in the RXFIFO) and the host resumes with data again. In SS mode, the host will send the requested data. If the device receives any packet with other than the requested data sequence, it will ignore it.	-
Step 5	-	-	Software processes the XferComplete event.
Software sets up data buffers after the host request.			
Step 6	Host sends data packets.	-	-
Step 7	-	Since DMA is not set up, the device will send NAK in Non-SS mode and will send NRDY in SS mode to the host. It also generates an XferNotReady event.	-
Step 8	-	-	On seeing the XferNotReady event, Software sets up TRBs and enables DMA by issuing Start Transfer.
Step 9	-	Steps 2-5 are followed.	-

Table continues on the next page...

Table 33-19. Non-Isochronous OUT Transfer Sequence (continued)

Steps	Host	Device Core	Device Software
Step 10	-	If the core encounters a short packet before the transfer size data programmed in the TRB is received, it skips the remaining TRBs of the transfer. If the core encounters control bits LST, IOC, or ISP it will generate events and, in the case of an LST TRB, the stream will be completed. Software has to reclaim any TRBs skipped with HWO=1.	On detecting a skipped TRB condition, software reclaims the TRBs with HWO=1.

33.4.3.2.5.2 Isochronous OUT Transfers

This section describes the difference between isochronous and non-isochronous OUT transfers referring to the sequence in Non-Isochronous OUT Transfers

- In Step 2, if the TRB is not fetched in time, the core discards the packet. In addition subsequent packets in the service interval also will be dropped.
- In Step 4, if DMA is enabled and if RXFIFO overrun happens or CRC fails, the core discards the packet. In addition subsequent packets in the service interval also will be dropped.
- In Step 7, if DMA is not enabled, the core drops the data and generates XferNotReady event.

In general, the TRB chain of a service interval will be released if the core receives DATA0 PID in HS USB 2.0 mode or receives a packet with LPF set in SuperSpeed mode.

Special scenarios are:

- If the core misses this last packet indication, it releases TRB at the service interval boundary.
- If the core encounters a short packet before the transfer size data programmed in TRB is received, it skips the remaining TRBs of the transfer.
- If the core encounters control bits 'LST' or 'IOC' or "ISP" it will generate event and, in the case of LST TRB, the stream will be completed, but software has to reclaim TRBs if any TRBs skipped with HWO=1.

33.4.3.2.5.3 Non-isochronous IN transfers

The sequence below describes a regular, non-isochronous IN data transfer (control-data/ bulk/interrupt).

Table 33-20. Non-isochrnous IN transfers

Steps	Host	Device Core	Device Software

Table continues on the next page...

Table 33-20. Non-isochrnous IN transfers (continued)

	Software sets up data buffers before a host request		
Step 1	-	-	Software sets up TRBs and enables DMA by issuing Start Transfer.
Step 2	-	Device fetches the TRB, caches it, and prefetches the data into TXFIFO. Only in SS mode, the core sends ERDY (Async ERDY, even before host requesting data provided endpoint is in flow controlled state. LSP flow is same regardless of if the host asked for it.)	-
Step 3	Non-SS host issues a token request, SS host issues ACK TP to request data.	-	-
Step 4	-	If DMA fetches keep up with host requests, the device sends all the data, updates the TRB status, and then generates an XferComplete event when ACKs for all the packets are received. When DMA is slow: • If the TXFIFO becomes empty on a host data request, the device sends a NAK in Non-SS mode. In SS mode, the device sends NRDY, and when data is available in TXFIFO it sends ERDY and waits for the host request again. • If the TXFIFO becomes empty in the middle of a packet transfer (happens only in Threshold mode), in Non-SS mode the device will invert the CRC. In SS mode, it terminates with EDB after appending the corrupted CRC. The core waits for the host to retry the packet before fetching the data again. If the host retries a packet: • In Non-SS mode, a transmitted packet is kept in the TXFIFO until the ACK arrives (this is immediate in Non-SS mode). If the host retries a packet, it is immediately sent out from the TXFIFO. • In SS mode, before ACK can be received, the core could have sent additional packets in a burst. When a retry happens, NRDY is sent to the host and the TXFIFO is flushed. A DMA command is issued on the master bus to prefetch data from where the host is asking. Once data is fetched into the TXFIFO, the device sends ERDY and waits for host request again. • In Threshold mode, when the TX threshold amount of data is fetched, the device sends ERDY to the host.	-
Step 5	-	-	Software processes XferComplete event.
	Software sets up data buffers after a host request.		

Table continues on the next page...

Table 33-20. Non-isochrnous IN transfers (continued)

Step 6	Non-SS host issues a token request. SS host issues ACK TP to request data.	-	-
Step 7	-	Since DMA is not set up, the device sends s NAK in Non- SS mode, or NRDY in SS mode, to the host. The core also generates an XferNotReady event	-
Step 8	-	-	On seeing the XferNotReady event, software sets up Normal TRBs and enables DMA by issuing Start Transfer.
Step 9	-	Steps 2-5 are followed.	-
Step 10	-	When software knows it is talking to a host that always ends transfers with short packets, and therefore always requires a zero-byte packet to end a transfer, it includes a zero-byte TRB (CHN=0) after the last TRB which is an even multiple of MaxPacketSize. When software does not know if the host is going to do another IN token after receiving the exact number of bytes it expected (and it was a multiple of MPS), it does not include a zero-byte TRB, and sets LST in the last TRB. If the host returns with another IN for a zero-byte packet, it uses the XferNotReady/Start Transfer mechanism to add one zero-byte TRB. This is called "On- Demand."	-

33.4.3.2.5.4 Isochronous IN Transfers

This section describes the difference between isochronous and non-isochronous IN transfers referring to the sequence in Non-isochronous IN transfers.

- In Step 4, if the TXFIFO is empty when host IN request arrives, the core returns a zero length packet in both Non-SS and SS modes.
- In Step 7, the device returns zero length packet and generates an XferNotReady event.

Special scenarios:

- When data is fetched and there is no request from the host (possibly due to a missing request or corrupted request) the data will be flushed at the end of the service interval (corresponding Micro- Frame) boundary and the core moves onto fetching next service interval data. If there are some TRBs that are not serviced in the service interval, the core will skip these TRBs.
- If the core encounters control bits LST or IOC, it will generate an event, and in the case of LST TRB, the stream will be completed, but has to reclaim TRBs if any TRBs skipped with HWO=1.

33.4.3.2.6 Handling ENDPOINT_HALT

On receiving a SetFeature (ENDPOINT_HALT) control transfer, software issues a Set Stall command on the endpoint. Because of the delay from when software issues the command and when it is recognized by the core, other transfer events (XferInProgress, XferComplete) may be received prior to the endpoint being halted. For a description of the Set Stall endpoint command and its effects, see Commands 4 and 5: Set Stall and Clear Stall (DEPSSTALL, DEPCSTALL).

On ClearFeature (ENDPOINT_HALT), software must first remove all pending transfers for the endpoint through the End Transfer command. It may then issue a Clear Stall command on the endpoint followed by Start Transfer to start transfers again. For a description of the Clear Stall endpoint command and its effects, see Commands 4 and 5: Set Stall and Clear Stall (DEPSSTALL, DEPCSTALL).

33.4.3.2.7 Handling L1 Event During a Transfer

This section discusses the scenario in which the software already issued the Start Transfer command but the LPM occurred before or during the data transfer.

When an IN transfer is in progress, (that is, started and not completed yet), and the software sees a link state change event to L1, then the software can use the GDBGFIFOSPACE register (write followed by a read) to determine if the TXFIFO is empty or not. If the TXFIFO is not empty, it can initiate a remote wakeup.

NOTE

- The GDBGFIFOSPACE register is a debug register which gives the 'Space Available" for the endpoint selected with a write prior to the read. If this space available is less than the particular TXFIFO's maximum size, then the TXFIFO is not empty.
- Software can enable the Link State change event for USB 2.0 as there are not a lot of events compared to SS.

33.4.3.3 Isochronous Transfer Programming Model

Isochronous endpoints get guaranteed service by the host during a "Service Interval". The service interval for an endpoint is communicated to the host via the endpoint descriptor, and it is configured in the hardware through the DEPCFG command. However, unexpected behavior on the bus may prevent the host from servicing the endpoint as frequently as expected.

Although the endpoint receives "guaranteed" service during the interval, there are no bus-level acknowledgments, which means that there is no guarantee that the host (IN endpoints) or the device (OUT endpoints) receives correct data.

Therefore, the isochronous programming model differs from the bulk programming model to accommodate these two factors:

- No bus-level acknowledgement of packet transmission or reception
- No guarantee that all the data setup for an interval has been transmitted or received

For isochronous endpoints, software has the following responsibilities:

- Set up enough TRBs to keep up with the rate of data transmission or reception
- For FIFO-based IN endpoints, guarantee the validity of the TX data at least 1 micro Frame before the beginning of the interval that the data will be transmitted

Hardware has the following responsibilities:

- Transmit or receive data at the appropriate bus time
- For FIFO-based IN endpoints, delay fetching until 1 micro Frame before the beginning of the interval that the data will be transmitted
- Maintain a 1-to-1 correspondence between Buffer Descriptors and the intervals on the bus
- Report missed isoc intervals and update the buffer sizes in TRBs to reflect the amount of data moved

33.4.3.3.1 Definitions

- bInterval: The field in an endpoint descriptor used to communicate the service interval of the device. Its value ranges from 1 to 16, however, the USB 3.0 core supports a range of 1 to 14.
- Service interval: An integral number of microframes (2^[bInterval-1]) during which the host will poll the device to move data.
- Beginning of interval: The interval begins when the least significant (bInterval-1) bits of the bus time are all 0's.
- End of interval: The interval ends when the least significant (bInterval-1) bits of the bus time are all 1's.
- Data payload: The number of bytes to be moved during the service interval. It will be transmitted using MaxPacketSize packets.
- Buffer descriptor: A set of one or more TRBs with CHN=1 and the last one having CHN=0. The group of TRBs represents the data buffers for one service interval.
- Last TRB of a Buffer Descriptor: The TRB within a Buffer Descriptor that has CHN=0. When an interval has completed, this TRB will contain the total remaining buffer size of the Buffer Descriptor andus also the completion status of the interval.
- Microframe (μF): A unit of time on the USB that lasts 125 μs.

- Start of Frame (SOF): The beginning of a microframe.
- FIFO-Based Isoc IN: A sub-type of Isoc IN endpoints that has the characteristic of always assuming the HWO bit in a TRB is '1' and delays fetching of transmit data.
- Prefetch delta: For FIFO-based IN endpoints, TX data may be prefetched as early as 1 μF before the beginning of a Service Interval, but never earlier.
- Retire a TRB: When HW sets the HWO bit to 0 in a TRB and writes it back.
- Retire a Buffer Descriptor: When HW retires at least the first and last TRB of a Buffer Descriptor.
  - Other TRBs of the Buffer Descriptor may not be retired if an interval was missed (IN or OUT) or if a short packet/last packet is received (OUT). In this case, software may reclaim those TRBs even though HWO=1.
- Missed interval: When the device does not move all the data in an interval. This may occur when the host does not poll for all the data, or because of application-side delays that prevent all the data from being moved.
  - For IN endpoints, this occurs when the host does not request all the data prepared in the Buffer Descriptor, or not all the data is fetched from the system bus in time.
  - For OUT endpoints, this occurs when the host does not send a packet, a packet is dropped due to CRC error, or the system bus is too busy to accept the data.

33.4.3.3.2 Endpoint configuration

An isochronous endpoint is setup in much the same way as a bulk endpoint, with the following exceptions:

- The bInterval_m1 value in DEPCFG Parameter 1 must be set to the value reported in the endpoint descriptor minus 1. When the core is operating in Full Speed, bInterval_m1 must be set to 0.
- The MaxPacketSize value in DEPCFG Parameter 0 must be set to the same value reported in the endpoint descriptor.
- The "FIFO-based" bit in DEPCFG Parameter 1, bit 31, must be set for FIFO-based isochronous endpoints.
- For the best performance, set the TXFIFOs corresponding to Isochronous IN endpoints to a high priority.

After the endpoint is configured, it can be enabled by setting the appropriate bit in the DALEPENA register.

33.4.3.3.3 Transfer configuration

Software describes isochronous transfers through a series of Buffer Descriptors. Each Buffer Descriptor corresponds directly to one service interval of data. The fields within an isochronous TRB are the same as bulk TRBs with the following exceptions:

- The SOF field in an IN endpoint TRB is a don't care. For an OUT endpoint, the core will write this field with the timestamp of the last packet received into the TRB's buffer. This information is not needed for normal operation, only for debug purposes.
- For High-Speed, High-Bandwidth IN endpoints, a maximum of 3 packets can be sent during an interval. The PktCntM1 field ([25:24] of the 3rd DWORD) must be set to the (number of packets in the Buffer Descriptor - 1). For example, if 3 packets are to be transmitted during the interval, this field must be set to 2. For Super-Speed and OUT endpoints, this field is a don't care.
- The first TRB in a Buffer Descriptor must have the TRBCTL field set to the "Isochronous-First" type while all others have this field set to "Isochronous".
- The ISP bit is renamed IMI (Interrupt on Missed Interval) and should be set if software wants to receive an XferInProgress event when at least 1 packet is missed within an interval.
- The IMI bit should be set to the same value in all TRBs of a Buffer Descriptor.
- The CSP bit must be set to 1 (short packets cause the hardware to move to the next Buffer Descriptor, they do not end an isochronous transfer).
- The LST bit should be set to 0 (isochronous transfers normally continue until the endpoint is removed entirely, at which time an End Transfer command is used to stop the transfer).

All other bits and fields (IOC, HWO, CHN, BPTR, BUFSIZ) retain the same behavior as they have for bulk endpoints. If software needs to receive an interrupt after every service interval, it should set the IOC bit to '1' in each TRB. However, if software wants to reduce the interrupt frequency and the application can tolerate some latency, the IOC bit can be set to '0' and the core will not generate a XferInProgress event when the TRB is completed.

For OUT endpoints, the Buffer Descriptor must describe a total buffer size of:

```
(Mult+1)*MaxBurst*MaxPacketSize
```

33.4.3.3.4 Starting a Transfer

The hardware will report the bus time that the host starts polling the endpoint inside the XferNotReady event. Software will use this value as a reference when issuing the Start Transfer command to serve as time synchronization with the host.

1. Set up TRBs and data buffer for the endpoint.
2. After configuring and enabling the endpoint, wait for an XferNotReady event.

The XferNotReady event will contain the time that the host started polling the endpoint in the upper 16 bits.

3. Issue the Start Transfer command to the endpoint with a future microframe time written into the upper 16 bits of the DEPCMD register. The future microframe time must be a value that is an integral multiple of intervals after the time reported in the XferNotReady event and aligned to the beginning of an interval. For example, if bInterval is 3 (4 microframes), and the XferNotReady time is 2, the value can be 4, 8, 12, and so on. The future microframe time must also be no greater than 4 seconds past the time reported in the XferNotReady event.
4. If the future microframe time has already passed when the command is received, the core will respond with an error (bit 13 in the Command Complete event).

In this case, software must issue End Transfer, then wait for another XferNotReady event and attempt the command again with a time that is further in the future.Otherwise, the first Buffer Descriptor will be used for the interval starting with the microframe time specified in the Start Transfer command.

33.4.3.3.5 Core Behavior During an Interval

After a transfer has been started, the hardware will perform the following functions for IN endpoints:

1. Fetch TX data as early as one interval prior to the beginning of the interval (call it A) if the HWO bit is set to one in the TRB.
2. Decrement the buffer size of each TRB as packets are transmitted.
3. Retire TRBs when their buffer size has reached 0, issuing an XferInProgress event if the IOC bit is set.
4. If the next interval (B) starts before all the packets have been transmitted for interval A:
   a. Flush the TXFIFO.
   b. Retire the Buffer Descriptor of interval A with a "Missed Isochronous" status.
   c. Retire the Buffer Descriptor of interval B with a "Missed Isochronous" status.
   d. See Checking interval status for a description of how software can determine that an interval ended unexpectedly.
   e. Go to step 1 to prepare for interval C
5. Otherwise, if all the TRBs of interval A are completed, the hardware will prepare for interval B.
6. If the host completes an interval by polling for all the data, but then it polls the endpoint again during the same interval, the hardware will respond with a zero-length packet and no interrupt will be made to software.

7. If the host polls the endpoint prior to the time specified in the Start Transfer command, the hardware will respond with a zero-length packet and no interrupt will be made to software.
8. If the host polls the endpoint during the expected interval and the hardware has no data prepared, the core will respond with a zero-length packet, but no XferNotReady event will be generated because the transfer is active. When the next interval starts, the XferInProgress event is generated (based on ISP and IMI) with the "Missed Isochronous" status, which is the way software finds out that this occurred.

For OUT endpoints, the hardware performs the following functions:

1. If a packet is received for the correct interval represented by the Buffer Descriptor (call it A).
   a. Write the packet into the buffer.
   b. Decrement the buffer size of the TRB.
   c. Write the timestamp of the received packet into the TRB.
   d. Retire a TRB when its buffer size reaches 0, issuing an XferInProgress event if the IOC bit is set.
2. If a short packet or packet with the last packet flag (lpf) is received for the correct interval.
   a. Write the packet into the buffer.
   b. Retire the Buffer Descriptor of interval A with the TRBSTS set to 0.
   c. If the IOC bit is set in the last TRB of the Buffer Descriptor, issue an XferInProgress event with bit [13] set.
   d. Go to step 1 to prepare for interval B.
3. If a packet is received for the correct interval but it has an error (CRC error, RXFIFO overflow), the core will not increment its expected sequence number value which causes future packets within the same interval to be dropped.
4. If a packet is received at a bus time prior to the time specified in the Start Transfer command, it will be dropped.
5. If a packet is received for the correct interval (A), but not enough buffer space is available for the core to write the packet (due to the HWO bit still '0' in one or more of the TRBs of the Buffer Descriptor), no XferNotReady event will be generated. The core will behave as follows:
   a. Wait until software sets the HWO bit to '1' and issues an Update Transfer command
   b. Retire the Buffer Descriptor of interval A with a "Missed Isochronous" status
   c. Go to step 1 to prepare for interval B.
6. If a packet is received for the next interval (B) before all the packets have been received for interval A:
   a. Retire the Buffer Descriptor of interval A with a "Missed Isochronous" status.
   b. Retire the Buffer Descriptor of interval B with a "Missed Isochronous" status.

c. See section Checking interval status for a description of how software can determine that an interval ended unexpectedly.
d. Go to step 1 to prepare for interval C.

Special Considerations for Isochronous OUT Endpoints

For OUT isochronous endpoints, the hardware detects a missed interval when the host sends a data packet in a future interval. It does not detect the missed interval at the exact interval boundary. Therefore, if the host abruptly stops sending isochronous OUT packets, there will be no interrupt or event indicating this to software. Software should use one of the following mechanisms to detect the interruption of isochronous OUT traffic:

1. The ultimate consumer of the isochronous OUT data will detect an underflow.
2. The device driver can use a timer detect that no XferInProgress events have been received for multiple intervals.

If this occurs, software should issue an End Transfer command to the endpoint and wait for a XferNotReady event which signals that the host is ready to resume isochronous traffic.

33.4.3.3.6 Checking interval status

As packets are transmitted or received during an interval, the buffer size of each TRB will be decremented. When the host is operating normally and polling for all the interval data, the buffer size of all the TRBs of an IN endpoint Buffer Descriptor will be zero after the interval has completed. For OUT endpoints, if the host sends less data than the Buffer Descriptor was setup for, the remaining buffer size may be greater than 0.

When the interval completes, the final remaining buffer size and missed isoc status is written to the last TRB of the Buffer Descriptor. If MissedIsoc is set, then it means the BUFSIZ is not accurate and may indicate that more data was transmitted or received than in reality. If the MissedIsoc is not set, it means the BUFSIZ field is correct.

When hardware detects the end of an interval (including normal and abnormal ends), it performs the following:

- If it has not already been retired, the first TRB of the Buffer Descriptor will be retired.
- Any non-first TRBs with CHN=1 that had not already been retired will not be written back. HWO will still be 1, and software can reclaim them for another transfer.
- The last TRB of the Buffer Descriptor will be retired with:
  - HWO = 0.

    - BUFSIZ = The total remaining buffer size of the Buffer Descriptor.
    - TRBSTS = "Missed Isoc" if any packets were missed, zero otherwise.
- If the IOC bit is set in the last TRB, or the IMI bit is set and packets were missed, hardware will issue an XferInProgress event

Software can get notification of an interval completing by setting the IOC bit in the last TRB of the Buffer Descriptor. The IOC bit may also be set in any other TRB of the Buffer Descriptor if software wants earlier notification that a TRB has completed. Software can also get only a notification of an interval completing unexpectedly by setting the IMI (Interrupt on Missed Isoc) bit in the last TRB of the Buffer Descriptor. When an interval completes unexpectedly and either the IOC or IMI bit is set in the TRB, the MissedIsoc bit (15) of the XferInProgress event will be set. The following table shows which events are generated in each scenario:

Table 33-21. TRB event generation events

Scenario	IOC	IMI	Action
TRB completed with a MaxPacketSize packet	0	0	No interrupt
	0	1	No interrupt
	1	0	XferInProgress (IOC=1)
	1	1	XrefInProgress (IOC=1)
TRB completed with a short packet	0	0	No interrupt
	0	1	No interrupt
	1	0	XferInProgress (Short=1, IOC=1)
	1	1	XrefInProgress (Short=1, IOC=1)
Interval completed due to missed packet (missed isoc)	0	0	No interrupt
	0	1	XferInProgress (MissedIsoc=1, IOC=1)
	1	0	XferInProgress (MissedIsoc=1, IOC=1)
	1	1	XrefInProgress (MissedIsoc=1, IOC=1)

It is normal to lose two intervals at a time when an error occurs during one interval. One interval is lost because of the host (which may not be polling enough) and the second interval is dropped because the device needs time to synchronize up to the next (third) interval.

33.4.3.3.7 Adding intervals to a transfer

Because isochronous endpoints represent a stream of data, the TRBs of an isochronous endpoint will normally be setup in a circular list, such as:

- TRB 1 (CHN=0, IOC=1)
- TRB 2 (CHN=0, IOC=1)
- Link (to TRB 1)

Example: Assume TRB 1 represents the data for the first interval and TRB 2 represents the data for the second interval. Because IOC=1, when the core completes the first interval, it will issue an XferInProgress event which indicates to software that TRB 1 can be analyzed to retrieve the results from interval 1. TRB 1 can the be re-used to represent the data for the third interval by setting HWO=1 and issuing an Update Transfer command.

Each time software sets up TRBs for a new interval, it must follow the guidelines in the Transfer Configuration section. Software will set up TRBs for future intervals when it obtains those buffers from another software layer or when TRBs are retired by the core.

33.4.3.3.8 Moderating Events

During a transfer, software may enable or disable any endpoint-specific event by re-issuing the DEPCFG command with the "Config Action" set to "Modify" and a modified DEPEVTEN field. All other field. All other fields in all other parameters must remain the same as the initial endpoint configuration.

33.4.3.3.9 Other Types of Isochronous Endpoints

The above description of the isochronous endpoint programming model assumes that software is setting up buffers in external memory that apply to certain intervals of data. This is called a "buffer-based" isochronous endpoint.

However, there is another mode that the hardware supports to accommodate other mechanisms of sourcing or sinking isochronous data, called "FIFO-based" and the types are defined by bits [31:30] of DEPCFG Parameter 1. This table illustrates the differences between the models:

Table 33-22. Isochronous Endpoints Models

Type	FIFO-Based [31]	Bulk-based [30]	TRB Fetch	TRB Write back	TX DMA	Retire Buffer Descriptor at passed interval
Buffer-based	0	0	When HWO=1	Yes	When HWO=1, no earlier than one interval ahead of time.	Yes
FIFO-based	1	0	When internal cache space available	No	No earlier than 1 µF before the interval	Yes
Invalid	1	1				

33.4.3.3.9.1 FIFO-based isochronous IN Endpoints

The software of some applications is unable to keep up with the short latency and high bandwidth of isochronous traffic and require the data to be sourced and done a sync through external FIFOs. One example is that there are two external TXFIFOs, one for each interval, and an address-to-FIFO-pop logic is implemented so that when the core attempts to read from interval 1's address, the translation logic converts this into a pop signal for interval 1's TXFIFO. When the core attempts to read from interval 2's address, the translation logic converts this into a pop signal for interval 2's TXFIFO.

To describe a FIFO-based implementation, software sets up an endpoint with the "FIFO-based" configuration bit set. This type of endpoint ignores the HWO bit in all TRBs, assuming that the TRB is always valid, and that the core should never write it back. Software chooses the buffer pointers within the TRBs to correspond to the address needed by the translation logic to specify which FIFO should be popped. By also using the CHN bit, headers and payload can be concatenated from different FIFOs. For example, if there are 4 external FIFOs: HA, PA, HB, PB, where HA/HB contain the 4 byte header for 2 intervals and PA/PB contain the 512 byte payload for 2 intervals, this can be described by using the following 5 TRBs:

1. BUFPTR=HA, IOC=0, CHN=1, BUFSIZ=4
2. BUFPTR=PA, IOC=1, CHN=0, BUFSIZ=512
3. BUFPTR=HB, IOC=0, CHN=1, BUFSIZ=4
4. BUFPTR=PB, IOC=1, CHN=0, BUFSIZ=512
5. Link to (1)

Every interval, the core will be creating a 516 byte packet that consists of 4 bytes from the header FIFO and 512 bytes from the payload FIFO. However, if the host does not poll for the packet, the core will skip 1 or 2 intervals of popping, depending on whether it has already started reading the next interval. External logic (or software) is responsible for flushing and refilling alternate FIFOs so that the core is always reading the correct data for the next interval. In normal intervals, the FIFOs will be empty, but when an interval is missed, there may still be data present in the FIFOs.

Software Requirements:

- The "FIFO-based" bit must be set in the DEPCFG when configuring the endpoint.
- No field within any TRB may be changed after the Start Transfer command is issued.
- Data must be valid in the external FIFO at least 1 uF before the beginning of the interval for which the data is intended.

Core Behavior:

- The earliest the core will read from the FIFO will be 1 µF before the beginning of the interval for which the data is intended.

- The core will not write back the TRB after it has completed it.
- The core will re-read the TRB from external memory even though it is not allowed to change.
- The only indication of missed intervals is the XferInProgress event if IOC or IMI is set in the TRBs. No indication will be made of how much data was actually transmitted.

33.4.3.3.9.2 FIFO-based isochronous OUT Endpoints

The FIFO-based model is also supported for OUT endpoints. In this case, core writes to specific addresses will be translated into external FIFO pushes. The same example from above (4 byte headers and 512 byte payloads) can be applied to OUT endpoints where the headers are split from the payload as the core receives 516 byte packets. The same software requirements and core behavior apply (for example, no writebacks and no indication of how much data was received). External logic assumes that if less than the expected amount of data is pushed into an interval's FIFO, the interval is invalid.

33.4.3.3.10 End a Transfer

Since isochronous endpoints have no bus-level acknowledgements, it is not necessary to set LST=1 in a TRB to gracefully end an isochronous transfer. Software can issue an End Transfer command to end an isochronous transfer. The core will wait until it can complete operations for the endpoint before returning the Command Complete event in response.

33.4.3.4 Control transfer programming model

Control transfers follow a flow of setup/data/status. On the USB bus, there are many error scenarios that can disrupt this flow. The hardware has a special mechanism that automatically recovers from these scenarios as long as software follows this single control transfer programming model.

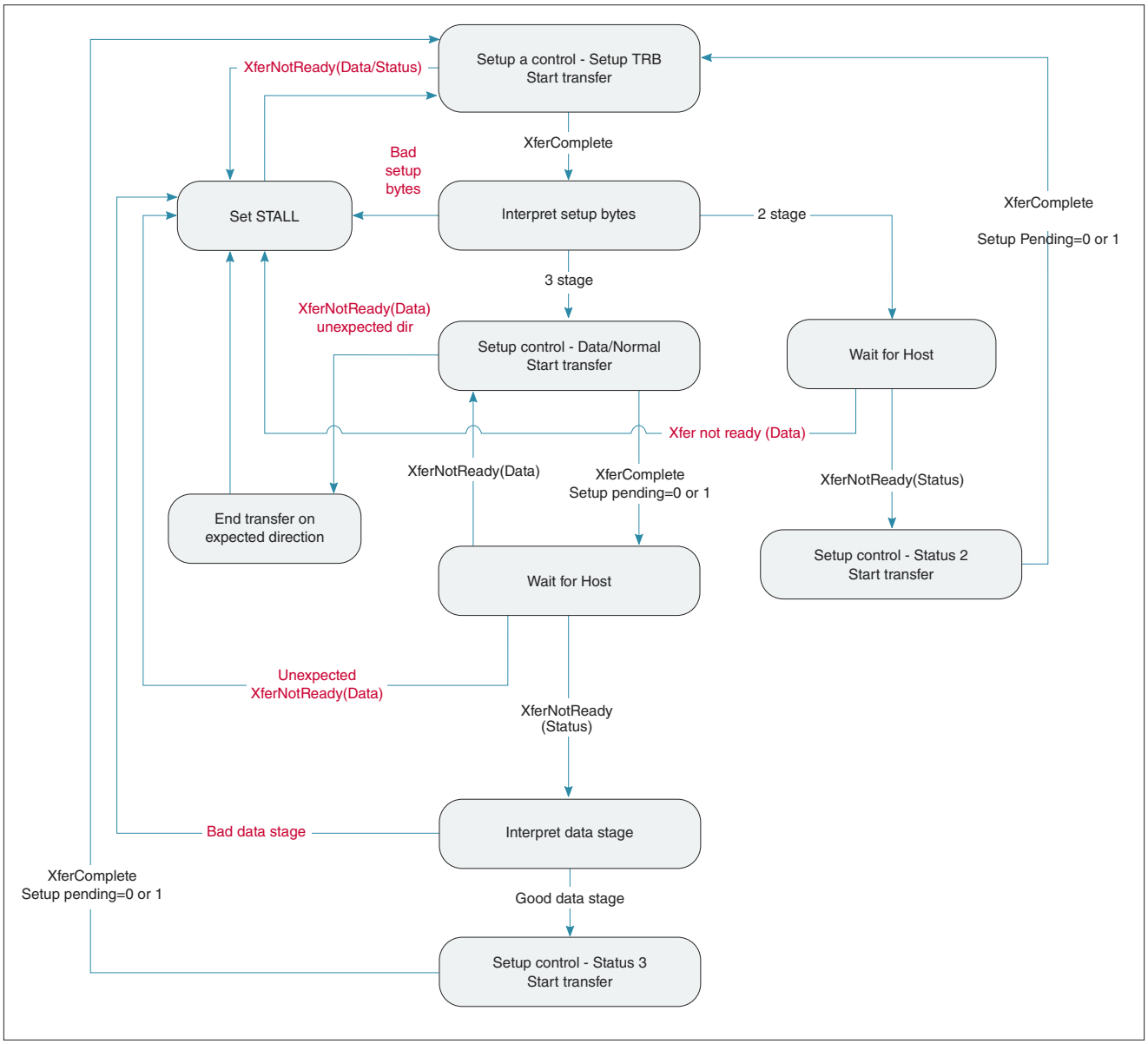


Figure 33-19. Software flow for control transfers

If hardware does not receive a setup packet, it discards the unexpected data and status stages. It does not issue XferNotReady to the application. The hardware will also not generate a XferNotReady event if it receives a setup packet before a TRB is prepared for it. As control endpoints are bi-directional, the IN and OUT directions of the endpoint are used for the different stages as follows:

Control read:

- EP0 - Setup OUT
- EP1 - Data IN
- EP0 - Status OUT/Status TP

Control write or 2-stage transfer:

- EP0 - Setup OUT
- EP0 - Data OUT (if present)
- EP1 - Status IN/Status TP

Set STALL is always issued on EP0, and the XferComplete/XferNotReady events are generated on the correct direction. For other control endpoints, substitute EP0 with the OUT direction (such as, 2, 4, 6) and EP1 with the IN direction (such as, 3, 5, 7). The same programming model is used for USB 2.0 and USB 3.0 control transfers.

33.4.3.4.1 Two-stage control transfer programming model

1. Software sets up a Setup TRB and issues Start Transfer on EP0 pointing to the Setup TRB.
2. After XferComplete is received, software interprets the setup bytes. If they are bad, issue Set Stall on EP0 and go back to the step 1. If a XferNotReady (Data/Status) event is received before the XferComplete event for the Setup stage, issue Set Stall. This is an error case where the host is attempting to move data or start the status stage for a previous control transfer that has already completed.
3. Wait for an XferNotReady event for the Status stage which will occur on EP1. If an XferNotReady event for the Data stage is received (either direction), issue Set Stall on EP0 and go back to the step 1. This is an error case where the host is attempting to start the data stage when the setup bytes did not indicate a data stage was present.
4. After XferNotReady (Status) is received, start the status stage by issuing Start Transfer on EP1 pointing to a valid Status-2 TRB.
5. After XferComplete is received on EP1, go back to the step 1.

Although not required, software may look at the Status TRB and examine the Setup Pending bit in the TRB status field. If it is set, it means that this control transfer was aborted on the USB bus and that the host did not receive the Status stage ACK.

33.4.3.4.2 Three-stage control transfer programming model

1. Software sets up a Setup TRB and issues Start Transfer on EP0 pointing to the Setup TRB.
2. After XferComplete is received, software interprets the setup bytes. If they are bad, issue Set Stall on EP0 and go back to the step 1. If a XferNotReady (Data/Status) event is received before the XferComplete event for the Setup stage, issue Set Stall. This is an error case where the host is attempting to move data or start the status stage for a previous control transfer that has already completed.
3. Start the data stage by issuing Start Transfer on EP0 (control write) or EP1 (control read) pointing to a valid Control-Data TRB.

   a. If an XferNotReady (Data) event is received for the incorrect direction, software must issue an End Transfer for the data stage it has already started, then issue Set Stall. This is an error case where the host is attempting to move data in the wrong direction.
   b. If a XferNotReady (Data) event is received for the correct direction, ignore the event and continue to wait for a XferComplete event.
4. After the XferComplete event is received, software waits for an XferNotReady event. Although not required, software may look at the completed Data TRB(s) and examine the Setup Pending bit in the TRB status field and the BUFSIZ. If the Setup Pending bit is set or the BUFSIZ is non-zero, it means that this control transfer was aborted on the USB bus and that the host did not complete the data stage.
5. If a XferNotReady (Data) event is received after the XferComplete for the Data stage, it could mean one of two things:
   a. This host is trying to complete the data stage by moving a 0-length packet. This can occur if the data stage was an exact multiple of max packet size. If this is the case, software sets up an extra TRB (with a BUFSIZ of max packet size for control writes, or 0 for control reads), issues Start Transfer, and goes back to the step 4.
   b. This host is trying to move more data than specified in the wLength field of the setup bytes. In this case, software issues Set Stall on EP0 and goes back to the step 1.
6. When the XferNotReady (Status) event is received, software interprets the data stage (for example, number of bytes transferred, content of the control write buffer) and decides if the data stage was successful. If not, it issues Set Stall on EP0 and goes back to the step 1.
7. Start the status stage by issuing Start Transfer on EP1 (control write) or EP0 (control read), pointing to a valid Status-3 TRB.
8. After XferComplete is received, go back to the step 1.

Although not required, software may look at the Status TRB and examine the Setup Pending bit in the TRB status field. If it is set, it means that this control transfer was aborted on the USB bus and that the host did not actually receive the Status stage ACK. When device software sets STALL with the Set Stall command, it is possible that this command can be delayed on the chip bus due to latency or the chip bus servicing other agents. In this case, software can receive an XferNotReady event if the core detects no TRB active before it detected the Set Stall. Software must service these events knowing that it stalled the control command.

33.4.3.4.3 Handling fewer requests than wLength

In a control IN transfer, if the host asks for less data than wLength, the core skips the TRBs of the data stage and generates an XferComplete event after encountering the TRB that has LST=1. Hardware sets HWO to 0 in those TRBs that it skipped and in the last TRB. In a control OUT transfer, if the host sends less data than wLength, the core treats this like a short packet received into a TRB with CSP=0. It immediately generates a XferComplete event, not setting the HWO bit in the remaining TRBs to 0. Software needs to reclaim those TRBs that still have HWO=1.

33.4.3.4.4 Control OUT transfer examples

The sequence in the following table describes two and three stage control OUT transfers.

Table 33-23. Control OUT transfer

Steps	Host	Device core	Device software
	Setup stage		
1	-	-	Software sets up the Setup TRB and enables DMA by issuing Start Transfer to EP0 (endpoint 0, OUT direction).
2	-	Device fetches the TRB and caches it.	-
3	Host sends SETUP packet.	-	-
4	-	SETUP packet is received in the RXFIFO. There are 24 bytes allocated for each control endpoint, which guarantees reception of the three back-to-back SETUP packets. An ACK handshake is returned in non-SS mode. An ACK TP with NumP=1 is returned in SS mode. After SETUP, bytes are transferred through the DMA, and the core issues an XferComplete event.	-
5	-	If the host starts a Data/Status stage, the core returns NAK in Non-SS mode or NRDY TP in SS mode. If the SETUP request is invalid, core returns STALL (when software sets STALL) in non-SS mode. In SS mode the core returns ERDY, and when the Data/Status stage is received it returns STALL.	Software processes the XferComplete event and decodes the SETUP bytes. If a Data stage is required, it will set up one Control-Data TRB and possibly subsequent Normal TRBs, then issue Start Transfer to EP0 (endpoint 0, OUT direction). If a Data stage is not needed, it will skip to the Status stage. If the SETUP request is invalid, software instead sets the STALL bit by issuing Set Stall command for EP0 (OUT direction).
	Data stage		
6	-	Device fetches the TRB and caches it. In SS mode only, the core will send ERDY.	-

Table continues on the next page...

Table 33-23. Control OUT transfer (continued)

Steps	Host	Device core	Device software
7	Host sends data packets.	-	-
8	-	The NumP for control transfer will be 1. In addition, if another SETUP is received before the data transfer happens on USB, the core will skip the data transfer that software sets up without waiting for it to happen on the USB bus, and send XferComplete (when LST=1 TRB encountered) with the SetupPndg status bit set. Software has to reclaim the TRBs with HWO=1 in the skipped TRBs.	-
9	-	If the host starts a Status stage, the core returns NAK in Non-SS mode or NRDY TP in SS mode.	On seeing the event: • Software decodes the control write data. If the Data stage is OK, then it will wait for XferNotReady with "Control Status Request" set, set up a Control-Status-3 TRB (0-bytes), and enable DMA by issuing Start Transfer to EP1 (endpoint 0, IN direction). • If the Data stage is not OK, then software sets STALL by issuing Set Stall command to EP0.
	Status stage		
10	-	Device fetches the TRB and caches it. In SS mode only, the core will send ERDY	Software sets up a Status stage TRB only after the Data stage of a three-stage control transfer is completed and an XferNotReady event occurred.
11	Host requests status packets.	-	-
12	-	In Non-SS mode, the device will send a zero-length packet if STALL is not set; otherwise, it will send STALL. In SS mode, if STALL is not set, then it sends ACK TP with NUMP=1 else it sends STALL. If another SETUP is received before the status transfer on USB, the core will send XferComplete with the SetupPndg status bit set. The STALL bit will be cleared by the core whenever it receives a SETUP packet. SETUPs are always accepted.	-
13	-	-	On seeing the event: • If there are no other errors, then the transfer has completed successfully.

33.4.3.4.5 Control IN transfer examples

The sequence in the following table describes two and three stage control IN transfers.

Table 33-24. Control IN transfer

Steps	Host	Device Core	Device Software
	Setup stage		
1-5	Steps 1-5 are same as Control OUT transfer examples, except that software sets up Tx buffers and Start Transfer is issued to EP1 (endpoint 0, IN direction).		
	Data stage		
6	-	Device fetches the TRB and caches it. It prefetches the data and puts it in the TXFIFO. The core sends ERDY only in SS mode.	-
7	Host requests data packets	-	-
8	-	This step is similar to the step 4 of Non-isochronous IN transfers. The NumP for control transfer is 1. In addition, if another SETUP is received before the data transfer happens on USB, the core will skip the data transfer software has set up without waiting for it to happen on the USB bus and send XferComplete (when LST=1 TRB encountered) with the SetupPndg status bit set. Software has to reclaim the TRBs with HWO=1 in the skipped TRBs and flush the TXFIFO.	-
9	-	If the host starts the Status stage, the core returns NAK in non-SS mode or NRDY TP in SS mode.	On seeing the event: • If the Data stage is OK, then it sets up Control-Status-3 TRB (zero-bytes) and enables DMA by issuing Start Transfer to EP0 (endpoint 0, OUT direction). • If the Data stage is not OK, then software will set a STALL by issuing Set Stall command to EP0 (OUT direction).
	Status stage		
	Same as in Control OUT transfer examples.		

33.4.3.5 Stream handling in SuperSpeed

The SuperSpeed stream protocol consists of a negotiation between the host and device concerning the selection of a stream, followed by data movement on the selected stream. The core automatically handles the stream protocol by initiating stream selection and also taking into account streams initiated by the host.

NOTE

This section is not applicable for USB 2.0 mode.

33.4.3.5.1 Stream IDs and transfer resources

A stream ID is a 16-bit value which represents an individual flow of data between host and device. For stream-capable endpoints, the stream ID must be non-zero and is used in:

- The Command Parameter field in the DEPCMD register used when issuing a Start Transfer command.
- The Stream ID field in a TRB.
- The EventParam field in an endpoint-specific event (DEPEVT): XferComplete, XferInProgress, XferNotReady, and StreamEvt.

It is necessary to allocate 1 Transfer Resource per endpoint. The cost of each transfer resource is (32 + N_TRBS_PER_XFER x 16) bytes of memory in the descriptor cache. Core can cache maximum four TRBs per transfer.

After the Start Transfer command completes, the hardware returns the 7-bit Transfer Resource Index in the command complete event (and Device physical endpoint-n command register (DEPCMD_0 - DEPCMD_7)). This index is used only for the Update Transfer and End Transfer commands. In all other places, the Stream ID is used.

33.4.3.5.2 Stream selection and stream programming model

Referring to the USB 3.0 Device Stream Protocol State Machine, the Idle state is the state where a device does not have a current stream to work on. When the hardware detects that it is in the Idle state, it will attempt to initiate and move data on a stream by transmitting ERDY to the host.

In order for this to occur, the endpoint must be "primed" and the stream must be both active and ready.

- **Primed**: After a stream-capable endpoint is configured, it is in the disabled state, which prevents the device from initiating any stream selection. After the host performs a Prime transaction, the endpoint is moved into the "Primed" state where it is eligible to initiate stream selection.

- **Active**: Software makes a stream active by issuing a Start Transfer command for the stream. A stream becomes inactive after an XferComplete event or after software issues an End Transfer command.
- **Ready**: After a stream is added via the Start Transfer command, its default state is "ready." The hardware will label the stream as "not ready" if the host gives a NoStream rejection to the hardware's attempt to initiate the stream. If the host performs a Prime, all currently active streams are automatically set to the "ready" state by hardware.

In the Idle state, the hardware selects a stream from its cache to initiate. Once the core has chosen an active stream, it attempts to initiate and move data on that stream until the endpoint enters the Idle state, which occurs when the host rejects the stream selection (NoStream) or the stream is terminated (PP=0) by the host or device. Therefore, software is required to prepare enough HWO=1 TRBs for at least one packet prior to issuing the StartXfer command.

In device-oriented stream selection class drivers (such as UASP), the host will not reject the device's stream selection. However, other class drivers may be developed in which the host initiates stream selection. In this situation, software will not start any transfers and will wait for a XferNotReady event. When that event is received, software will look at the StreamID in the event and start the transfer associated with the given Stream ID. If for some reason the class driver allows the host to reject a stream that has already been initiated by the host or device, software will receive a Stream (NotFound) event. If this occurs, software may replace the existing stream by issuing End Transfer for it and then issuing Start Transfer for another stream.

Some host implementation scenarios exist where the host and device become out of sync in the stream protocol, creating a deadlock condition where the device waits for the host to issue a Prime transaction while the host waits for the device to issue an ERDY. To resolve potential deadlock conditions, software should perform the following:

1. Implement a time out for stalled transfers on stream-capable endpoints.
2. When the timeout elapses with no data transfer for any stream after a StreamEvent (NotFound) event, restart one of the streams by issuing an End Transfer command followed by a Start Transfer command.

This places the stream in the Ready state and causes the core to transmit an ERDY to the host.

33.4.3.5.3 Data movement within a stream

Data movement within a stream is the same as data movement using the Bulk endpoint type (Start Transfer, Update Transfer, and updating TRBs). The XferInProgress, XferComplete, and XferNotReady events are issued as data is transferred between host

and device. The Stream ID field in the events indicate which stream within the endpoint is generating the events. If software receives an XferNotReady event with the EventStatus field indicating XferNotActive, it means that the host is attempting to initiate a stream that has not been added to the hardware's cache through Start Transfer. In response to this event, software should add the requested stream if it is available.

33.4.3.6 Low power operation

33.4.3.6.1 Low power operation of USB

The USB 3.0 protocol allows a SuperSpeed host and device to negotiate with each other and decide when to bring their link into a lower power state (U1 or U2). This is accomplished by each side making individual decisions about whether they have any traffic currently pending or if they expect any traffic soon. The device requests low power entry by transmitting an LGO_U1 (or LGO_U2) Link Command and exits low power by using an LFPS handshake. The USB 3.0 Device Controller uses information from the host (bus side) and from software (application side) to decide when the best time to request low power entry is and when to exit from low power. The same calculation is used when the controller decides whether to accept a low power request from the host. The controller maintains state information about each endpoint and whether it is "active" or "paused". If all enabled endpoints are paused from either the application side or the bus side, low power entry is allowed. If at least one endpoint is not paused, low power entry is not allowed. To indicate that it does not have any pending traffic and allow the link to go into a low power state, the software may pause all enabled endpoints using one of the following mechanisms:

- Allow transfers to complete and not start any new ones
- Not setting the HWO bit in the TRBs to '1'

Exception: Software may prepare a Setup TRB for control endpoints without affecting the ability of the link going into low power. Otherwise, an endpoint is considered active on the application side. For these endpoints, the host indicates that it is paused in one of the following ways:

- Setting the Packet Pending (PP) flag to '0' in the DPH
- Setting the PP flag to '0' and NumP to '0' in the ACK TP
- For interrupt endpoints, by not polling the endpoint for two service intervals
- Note: After an endpoint is configured using the DEPCFG command and before the host sends any packets to it, the endpoint is considered to be paused on the bus side

Exception: If the data packet payload has a CRC error, the core responds with an ACK (Retry=1), and the endpoint is not paused. Otherwise, if PP=1 or Deferred=1, the endpoint is considered to be active on the bus side. Control endpoints have an additional condition as follows:

The core does not allow low power entry during a control transfer (starting from the reception of setup packet and ending at the completion of the status stage).If the core is configured for isochronous support, the following additional rules apply:

- The controller enters U0 at SystemExitLatency microseconds prior to the beginning of every microframe to receive the Isochronous Timestamp Packet (ITP).
- A PING causes every isochronous endpoint to wake up and be considered active, preventing low power entry.
- An isochronous endpoint is paused automatically when it transmits or receives the last packet of its interval (lpf=1) or if two intervals pass and the endpoint has not transmitted or received its last packet. In this way, after the PING, the core does not enter low power until all isochronous endpoints have moved their last packet or two intervals pass without another PING.

If all enabled endpoints are paused by software or the host, the core issues an LGO_U1 (or LGO_U2) link command or it responds with an LAU link command to the low power request of the host or hub. Otherwise, the core rejects the low power requests with an LXU link command.

Additionally,

- For bulk endpoints, the USB 3.0 specification says that the device may go to U1/U2 500ms (tERDYTimeout) after transmitting ERDY. It is the responsibility of the software to detect no activity on a bulk endpoint for 500ms or more and issue End Transfer if there is no activity. When the transfer is removed, the endpoint allows U1/U2 entry again.
- Because U1/U2 entry is based on the presence of packets in the TX FIFO and not based on the TRBs that are available to the core, it is possible that an IN endpoint responds with a NRDY, goes to U1, then U0, then transmits an ERDY if the timing and TXFIFO size works out such that the core cannot keep at least one packet in the TXFIFO at all times.
- In addition, before the final decision made to either initiate or accept low power state, device checks the state of DCTL[INITU2ENA][ACCEPTU2ENA][INITU1ENA] [ACCEPTU1ENA]. If the device receives Set Link Function LMP with Force_LinkPM_Accept bit asserted, the device accepts low power request from its link partner independent of endpoints state and DCTL Power control settings.

33.4.3.6.2 Low power operation of core

When the link is in the U1, U2, U3, SS.Inactive, SS.Rxdetect, or SS.Disabled state (for USB 3.0) or the USB is in Sleep, Suspend, or V_{BUS}-off mode (for USB 2.0), the core can be put into a low power mode to save power. The core supports the following low power mode:

- **Clock-gating mode**: In this mode, the clocks connected to most of the core modules are gated off. Modules that still get clocks in the low power mode detect wakeup conditions and remove clock gating to other modules when a wakeup condition is detected. Clock-Gating mode can be used when the USB is in any low power state. Clock-Gating mode is enabled by setting the Disable Clock Gating bit in GCTL to 0.

33.4.3.6.2.1 Clock-Gating Mode

Clock-Gating mode is enabled if the Disable Clock Gating bit in the GCTL is set to 0. When the USB/link is in a low power state, the clocks to most of the core modules are gated off automatically by the core. When a wakeup condition is detected by the core, or when the software schedules new work to the core (in U1 or U2 states) or changes the USB/Link state, the core turns on clocks and the entire core goes into active mode.

33.4.4 Host programming model

Please refer xHCI specification for detailed information.

33.4.4.1 Initializing host registers

In order to initialize the core as a host, the application should perform the steps described in the xHCI specification. Global registers need to be re-initialized as described in Initializing global registers.

33.4.4.2 Host controller capability registers

For register definition, refer to Capability registers length and HC interface version number (CAPLENGTH) to Runtime register space offset (RTSOFF).

33.4.4.3 xHCI implementation details

This section discusses xHCI implementation details specific to the USB 3.0 controller.

33.4.4.3.1 LHCRST behavior

The xHCI Specification does not describe the programming model of light reset. It only specifies that the Operational and Runtime Registers that are not contained in the Aux Power well are at their default values.

Light reset must only be applied when USB_DCTL[Run_Stop] is equal to '0' to ensure that there is no discrepancy between hardware and software states. If light reset is applied during a transfer, the behavior on the USB is undefined, and may cause a packet to be terminated abruptly.

On light reset, the hardware resets all non-Aux registers to their default values which means that the port state (which is contained in PORTSC) does not change, but USB_DCTL[Run_Stop] is immediately set to '0' and all the context information about connected devices is lost. It is the responsibility of software to re-initialize the controller.

33.4.4.3.2 ENT requirements

In the xHCI Specification, the rules about when the ENT (Evaluate Next TRB) flag in a TRB must be set or cleared are not very strict. However, it states that the ENT flag must be set to '1' in the last Normal TRB when a TD ends with an Event Data TRB.

The controller requires software to follow these rules:

1. Regardless of whether the endpoint is stream-capable or not, if a TD ends with an Event Data TRB, the Normal TRB that precedes it must have ENT set to '1'. This Normal TRB may be separated from the Event Data TRB by a Link TRB
2. In all other cases, the ENT flag must be set to '0' Failure to follow these rules results in undefined behavior.

33.4.4.3.3 Behavior on babble error

If a babble error is detected and the received data passes all integrity checks, the host controller may write the received data (up to the expected data length) to the data buffer, and the value of the TRB transfer length field in the babble detected error transfer event shall be consistent with the number of data bytes written to the buffer.

The controller does not write the received data to the data buffer. The entire packet is discarded and the residual byte count is written to the TRB transfer length field.

33.4.4.3.4 Max_exit_latency_too_large message

If periodic transfers are on every microframe (Binterval 1), the controller reports max_exit_latency_too_large if PING scheduling is enabled. The xHCI scheduler must prefetch data from the system memory ahead of next microframe, manage non-periodic transfer, and schedule PING before ISOC. Because the system memory access and USB responses are not predictable Quality of Service (QoS) is not guaranteed if the xHCI host needs to perform all of the above for every microframe when U1/U2 is enabled. Therefore, the expectation is to disable U1/U2 if periodic data needs to be scheduled every microframe. In addition, because the U1 exit latency is in the order of 10usec, only very low system-level power saving is achieved even if the scheduler allows U1 transition by reducing QoS.

U1, U2, and U3 link states and exit time observed during lab testing:

- U1: Lowest power saving
  - In this state the chip is active, USB controller is inactive, only the Tx transmitter of the PHY is inactive.
  - U1 exit time per link partner: ~10 - 19 usec
- U2: Higher power saving than U1
  - In this state the chip is active, USB controller is inactive, both the Rx and Tx transceivers of the PHY are inactive.
  - U2 exit time per link partner: ~85 - 115 usec
- U3: Highest power saving
  - In this state, the chip and USB controller/PHY are inactive.
  - U3 exit time per link partner: 200 - 450 usec

Following are the recommendations for driver to handle periodic endpoints and "max_exit_latency_too_large" message:

- If Binterval of any endpoint is '1', then disable U1/U2 and do not schedule PING ("max_exit_latency" set to 0).
- If periodic scheduling is expected on every microframe, then disable U1/U2 and do not schedule PING.
- If periodic scheduling is expected only on every other microframe, then disable U2 and use U1 exit latency for "max_exit_latency" calculation.
- If ((number of hubs +1) * U2 exit latency) >= (2**(Binterval-1) * 125 usec), then disable U2.
- For all other periodic scheduling use U2 exit latency for "max_exit_latency" calculation.
- If the "max_exit_latency_too_large" error is sent to software, then disable U2 and re-issue ep- config with U1 exit latency.
- If "max_exit_latency_too_large" still happens, then disable U1 and re-issue ep_config with max_exit_latency set to 0 (no PING).

33.4.5 Device physical endpoint-specific commands

33.4.5.1 Command 1: Set endpoint configuration: DEPCFG

This command sets the physical endpoint configuration information.

NOTE

If GUSB2PHYCFG[6] is set to '1', it must be set to '0' prior to issuing this command and may be set to '1' after the command completes.

Table 33-25. Command 1: Set endpoint configuration parameters: DEPCFG

Field	Description
Parameter 2	
31-0	Set to endpoint state when the Config Action field of Parameter 0 is 1. Otherwise, this is Reserved.
Parameter 1	
31	FIFO-based Set to '1' if this isochronous endpoint represents a FIFO-based data stream where TRBs have fixed values and are never written back by the core.
30	Bulk-based Set to '1' if this isochronous endpoint represents a bulk data stream that ignores the relationship of bus time to the intervals programmed in TRBs.
29-25	USB Endpoint Number bit[29:26]: Endpoint number bit[25]: Endpoint direction: 0 OUT 1 IN Physical endpoint 0 (EP0) must be allocated for control endpoint 0 OUT. Physical endpoint 1 (EP1) must be allocated for control endpoint 0 IN.
24	Stream Capable (StrmCap) Indicates this endpoint is stream-capable (MaxStreams != 0).
23-16	bInterval_m1 Set to the bInterval value minus 1. The valid values for this field are 0 through 13. The bInterval value is reported in the endpoint descriptor. When the core is operating in Full-Speed mode, this field must be set to 0.
15	Set to '1' if this bulk endpoint utilizes the External Buffer Control (EBC) mode. This limits the number of outstanding DMA transfers to one read and one write. This is a requirement for the EBC to operate properly. Using EBC without setting this bit to '1' results in undefined behavior.
14	Reserved

Table continues on the next page...

Table 33-25. Command 1: Set endpoint configuration parameters: DEPCFG (continued)

Field	Description
13-8	Bits 13-8 are used for Device Endpoint Specific Event Enable (DEPEVTEN). If an enable bit is set to 0, the corresponding event is not generated. Bit 13 Stream Event Enable (StreamEvtEn) Bit 12 Reserved Bit 11 Reserved Bit 10 XferNotReady Enable (XferNRdyEn) Bit 9 XferInProgress Enable (XferInProgEn) Bit 8 XferComplete Enable (XferCmplEn)
7-5	Reserved
4:0	Interrupt number (IntrNum) Applies to IN and OUT endpoints. Indicates interrupt/Event Buffer number on which endpoint related interrupts for this endpoint are generated. This must be set to 0.
Parameter 0	
31-30	Config Action 0 Initialize endpoint state: This action is used when an endpoint is configured the first time. It will cause the data sequence number and flow control state to be reset. DEPCMDPAR2 will be ignored. The encoding of this action is backward compatible with software that previously set "Ignore Sequence Number" to 0. 1 Reserved. 2 Modify endpoint state: This action is used when modifying an existing endpoint configuration, such as changing the DEPEVTEN event enable bits, interrupt number, or MaxPacketSize. The data sequence number and flow control state will be unchanged, and DEPCMDPAR2 will be ignored. The encoding of this action is backward compatible with software that previously set "Ignore Sequence Number" to 1.
29-26	Reserved
25-22	Burst Size (BrstSiz) When field is set to- 0 Burst length = 1 1 Burst length = 2, and so on, up to, 15 Burst length = 16 For IN transfers, this value represents the maximum length of the burst that the device attempts when the Host initiates an IN transfer with a TP_ACK. If BrstSiz < the NumP value in the initiating TP_ACK, then the device controller limits the burst length to BrstSiz. If BrstSiz >= the NumP value in the initiating TP_ACK, then the device controller attempts a burst length of NumP. For OUT transfers, this value represents the NumP value utilized in the response TP_ACK from the device controller. The NumP value indicates (to the host) the burst length the device desires. Note: In the special case of BrstSiz=0 (burst length = 1), the TP_ACK from the device controller has NumP=0; which is a flow-control condition. If this value is utilized, then the endpoint enters flow

Table continues on the next page...

Table 33-25. Command 1: Set endpoint configuration parameters: DEPCFG (continued)

Field	Description
	control for each DP received and a subsequent ERDY is transmitted (according to the USB Specification). However, this may have an undesirable impact on the system throughput. To avoid this impact, it is recommended to set BrstSize=1 to prevent the flow-control condition. The trade-off is that the host could potentially burst two OUT data packets to the device, resulting in an ACK for the first packet, and an NRDY for the second packet.
21-17	FIFO Number (FIFONum) Indicates which transmit FIFO is assigned to this endpoint. For control endpoints, the FIFONum value in the OUT direction must be programmed to the same value as the IN direction. This field should be set to 0 for all other OUT endpoints. Even though there may be more than 16 TXFIFOs in DRD mode, the device mode must use lower 16 TXFIFOs.
16-14	Reserved
13-3	Maximum Packet Size (MPS) Applies to IN and OUT endpoints. The application must program this field with the maximum packet size (in bytes) for the current USB endpoint. USB 3.0 supports up to 1024 bytes.
2-1	Endpoint Type (EPType) This is the transfer type supported by this USB endpoint. 00 Control 01 Isochronous 10 Bulk 11 Interrupt
0	Reserved

33.4.5.2 Command 2: Set endpoint transfer resource configuration (DEPXFERCFG)

There must be only one transfer resource allocated per endpoint. Start transfer causes the use of the transfer resource. End Transfer or an XferComplete event releases the transfer resource. If software attempts to allocate more transfer resources than have been configured in the hardware, this command will return an error in the CmdStatus/ EventStatus field.

NOTE

If GUSB2PHYCFG[SUSPENDUSB20] is set to '1', it must be set to '0' prior to issuing this command and may be set to '1' after the command completes.

Table 33-26. Command 2: Set endpoint transfer resource configuration parameters: DEPXFERCFG

Field	Description
Parameter 2	
31-0	Reserved
Parameter 1	
31-0	Reserved
Parameter 0	
31-16	Reserved
15-0	Number of Transfer Resources (NumXferRes) Defines the number of Transfer Resources allocated to this endpoint. This field must be set to 1.

33.4.5.3 Command 3: Get endpoint state (DEPGETSTATE)

This command is not used.

33.4.5.4 Commands 4 and 5: Set Stall and Clear Stall (DEPSSTALL, DEPCSTALL)

These commands apply to non-isochronous IN and OUT endpoints only.

The application issues a Set Stall command to stall all tokens from the USB host to this endpoint. If the endpoint is in the NRDY state, the STALL state takes priority.

If a transaction is currently in progress when the software issues Set Stall, the behavior depends on the direction of the endpoint:

- For OUT endpoints, the current packet will complete. The core will respond with STALL to the next OUT DP or token.
- For IN endpoints in SuperSpeed, the current burst transaction will complete, and the core will respond with STALL to the next ACK TP. For other speeds, the core will complete the current packet and respond STALL to the next IN token.

For non-control endpoints, the application is responsible for both setting and clearing STALL via the Set Stall/Clear Stall commands. When the application clears the STALL, the endpoint's data sequence number is reset to zero.

For control endpoints, the application issues only the Set Stall command, and only on the OUT direction of the control endpoint. The controller automatically clears the STALL when it receives a SETUP token for the endpoint.

The application must not issue the Clear Stall command on a control endpoint. If the endpoint is in flow control (NRDY) and also in the STALL state, it responds with a STALL for any packet. The only exception is that the controller always responds to SETUP data packets with an ACK handshake, independent of the STALL state.

NOTE

- If GUSB2PHYCFG[6] is '1', it must be set to '0' prior to issuing this command and may be set to ‘1’ after the command completes.
- When issuing Clear Stall command for IN endpoints in SuperSpeed mode, the software must set the "ClearPendIN" bit to '1' to clear any pending IN transcations, so that the device does not expect any ACK TP from the host for the data sent earlier.

33.4.5.5 Command 6: Start transfer (DEPSTRTXFER)

This command applies to IN and OUT endpoints.

A Transfer Descriptor (TD) in Device mode is a list of TRBs that comprises one or more transfers. The pointer to a TD is added to the core's cache by the Start Transfer command and removed by an XferComplete event or by the End Transfer command.

Software issues this command indicating that a descriptor is ready to be processed and DMA can start for this endpoint/stream combination. For IN endpoints, this causes the descriptor to be processed, and data is moved from the corresponding memory buffer to corresponding transmit FIFO when the transfer is started. For OUT endpoints, this causes the descriptor to be processed and data is moved from the receive FIFO to corresponding memory buffer.

In response to the Start Transfer command, the hardware assigns this transfer a resource index number (XferRscIdx) and returns the index in the DEPCMDn register and in the Command Complete event. This index must be used in subsequent Update and End Transfer commands.

It is illegal to issue a Start Transfer command for the same TD if it is already present in the core's cache.

Non-stream capable endpoints rules:

- The Stream ID field must be set to 0.
- The HighPri field is reserved.

Stream-capable endpoint rules:

- The Stream ID field must be set to non-zero and match the Stream ID passed into the Start Transfer endpoint command associated with this transfer. In all the TRBs of a transfer, the Stream ID fields must be the same.
- The HighPri field is reserved.

NOTE

- If GUSB2PHYCFG[6] is set to '1', it must be set to '0' prior to issuing this command and may be set to '1' after the command completes.
- Before issuing a start transfer command, software needs to verify whether the link is in U0. If the link is not in U0, software needs to bring the link to U0 by performing a remote wakeup. This is applicable to both SS and USB 2.0 speeds.
- When an IN transfer is in progress, (that is, started and not completed yet), and the software sees a link state change event to L1, then the software can use the GDBGFIFOSPACE register (write followed by a read) to determine if the TXFIFO is empty or not. If the TXFIFO is not empty, it can initiate a remote wakeup.

Table 33-27. Command 6: Start transfer parameters: DEPSTRTXFER

Field	Description
Parameter 2	
31-0	Reserved
Parameter 1	
31-0	Transfer Descriptor Address (TDAddr Low) Indicates the lower 32 bits of the external memory's start address for the transfer descriptor. Because TRBs must be aligned to a 16-byte boundary, the lower 4 bits of this address must be 0.
Parameter 0	
31-0	Transfer Descriptor Address (TDAddr High) Indicates the higher 32 bits of the external memory's start address for the transfer descriptor.

33.4.5.6 Command 7: Update transfer (DEPUPDXFER)

If software uses circular TRB buffers and updates a TRB, whose Hardware Owner (HWO) bit was 0, by setting HWO=1, it must execute the Update Transfer command, specifying the transfer resource index of the TRB in the DEPCMD register. The hardware uses this information to re-cache the TRB.

Software may issue a special “No Response Update Transfer” command by setting CmdAct=0 and CmdIOC=0. In this case, the hardware does not generate a Command Complete event, does not set the CmdAct bit to '0' (because it will be '0'), and software may immediately issue another command to the same endpoint following this one. This special type of Update Transfer may not be used when software depends on the XferNotReady event to setup TRBs (see “On Demand” transfers in Transfer setup recommendations).

Software may issue an Update Transfer command for a transfer resource that has already completed (either due to XferComplete event or an End Transfer command), and the core will detect that the Update Transfer is unnecessary. However, software must not issue an Update Transfer command for a transfer resource index that has never been started.

NOTE

If GUSB2PHYCFG[6] is set to '1', it must be set to '0' prior to issuing this command and may be set to '1' after the command completes.

33.4.5.7 Command 8: End transfer (DEPENDXFER)

This command applies to IN and OUT endpoints. Software issues this command requesting DMA to stop for the endpoint/stream specifying the transfer resource index of the TRB and the ForceRM parameter to be set to 1 in the DEPCMD register.

When issuing an End Transfer command, software must set the CmdIOC bit (field 8) so that an Endpoint Command Complete event is generated after the transfer ends. This is necessary to synchronize the conclusion of system bus traffic before the End Transfer command is completed.

For IN endpoints, this command causes descriptor processing to stop and the core stops fetching new data for the endpoint and stops transfers on the USB. The core may truncate a packet being transmitted with the DPPABORT ordered set, does not wait for any pending ACKs from the USB, and does not update the TRB status. For OUT endpoints, this command causes descriptor processing to stop. If there is currently data, it is moved from the receive FIFO to corresponding memory buffer and completed at packet boundary. The core does not update the TRB status.

Use this command under the following conditions:

- When handling USBReset or SetConfiguration, endpoints are closed using this command.
- After receiving a ClearFeature (STALL) control transfer, software issues End Transfer followed by Clear Stall, followed by Start Transfer.

- After an XferInProgress event when the TRB after the one that caused the XferInProgress event has its HWO bit set to '0', this command can be used.
- For isochronous endpoints, if the host stops moving data for many intervals, software may force the end of the transfer and wait for the host to restart.

The hardware does not issue a XferComplete event on End Transfer, but only issues a CommandComplete event.

NOTE

If GUSB2PHYCFG[6] is set to '1', it must be set to '0' prior to issuing this command and may be set to '1' after the command completes.

Table 33-28. Command 8: End Transfer Parameters: DEPENDXFER

Field	Description
Parameter 2	
31-0	Reserved
Parameter 1	
31-0	Reserved
Parameter 0	
31-0	Reserved

33.4.5.8 Command 9: Start new configuration (DEPSTARTCFG)

Software issues this command under the following conditions:

- After power-on-reset with XferRscIdx=0 before starting to configure Endpoints 0 and 1. CmdIOC must be set to '0' and software must poll the CmdAct bit to determine when the command is complete because Endpoint 0 is not yet configured with a valid interrupt number.
- With XferRscIdx=2 when it receives SetConfiguration before starting to configure Endpoints > 1. CmdIOC may be set to '0' or '1'.

This command should always be issued to Endpoint 0 (DEPCMD0).

Hardware resets the transfer resource allocation to the value in the XferRscIdx parameter (must be 0 or 2) upon receiving this command.

NOTE

If GUSB2PHYCFG[6] is set to '1', it must be set to '0' prior to issuing this command and may be set to '1' after the command completes.

33.4.6 OTG

This section describes OTG for USB 3.0, and the programming requirements for the USB 3.0 core in OTG mode.

33.4.6.1 OTG 2.0 for USB 3.0 Functionality

The following sections describe the OTG 2.0 functions for USB 3.0.

33.4.6.1.1 Core OTG functions

The OTG Interface block within the U2MAC handles the core OTG functions: Session Request Protocol (SRP) and Host Negotiation Protocol (HNP). These OTG protocols are implemented through the regular UTMI+ OTG interface.

The MAC handles HNP for host and peripheral role swapping.

The MAC also handles SRP, which allows an A-Device to turn off VBUS to save power when the USB is not used, and provides a means for a B-Device to monitor VBUS and request the A-Device to activate VBUS.

It includes the following logic necessary to achieve SRP and HNP:

- Control of VBUS through Vbus drive enable (utmiotg_drvvbus) as A-Device
- Control of IDDIG line sampling enable control output (utmiotg_idpullup)
- Control of D+/D- pull-down resistor enables (utmiotg_dppulldown/ utmiotg_dmpulldown)
- Generates SRP (data line pulsing) as B-Device when the session is off.

NOTE

- When the application programs the USB_GCTL[PRTCAPDIR] as 2'b11 to enable OTG mode, then in the A- Device mode, the core will only enumerate as a HS device and hence will not support any SS-capable device to be connected. This restriction is not valid when the USB_GCTL[PRTCAPDIR] is programmed to 2'b01 or 2'b10.
- The utmisrp_chrgvbus and utmisrp_dischrgvbus outputs are provided for legacy PHY connections but they are both

set to 1'b0 from the core since VBUS charging is not supported in OTG 2.0 specifications.
- The application should not program GCTL[0] as 1'b0 when it wants to do a ADP or HNP. In such a case, the application should disable the Clock Gating feature by programming GCTL[0] as 1'b1. The application should re-enable the Clock Gating feature only when the core returns to its original role of operation.

33.4.6.1.1.1 HNP Polling and Enable

This HNP Polling activity involves an OTG device acting as the current host to periodically poll the remote device to check if the remote device wishes to take the host role. It will then grant the role swap opportunity at the earliest opportunity. Being a software activity, HNP Polling is expected to be implemented through software timers and periodic exchange of SetFeature.SetHNPEnable packets.

The core is then informed of successful exchange of these packets to Enable HNP activity within the core.

33.4.6.1.2 ADP functions

The ADP functions involve the following two processes:

- ADP sensing
- ADP probing

ADP probing capability is required in both A and B devices, while sensing is a must only for B-Device. The main functional unit is the ADP controller.

33.4.6.1.2.1 Internal ADP controller

- All ADP timers are maintained in pwrm (Power) module and generate the enable and disable signaling to PHY for ADP probing and sensing.
- PHY contains the following circuitry related to ADP functionality:
  - Comparators for PRB and SNS
  - I_ADP_SRC and I_ADP_SNK
  - Vbus circuitry
- Application software programs the ADP timing registers residing in the pwrm module.
- pwrm module provides a mechanism to application through the interrupts to log and report events pertaining to ADP probing and sensing.

- Only pwrm module needs to be always powered on in this option. USB 3.0 controller may or may not be powered on.
- As a product, both the ADP controller logic and OTG controller are packaged into USB3 core. In the USB3 core, ADP sensing and probing is loosely coupled with the rest of the logic and is directly under software control. This achieves maximum flexibility to interact with OTG core functions.

The ADP controller in the block diagram is a fully digital controller. This has mainly timers inside it to help ADP operation. All ADP related timers are part of pwrm module.

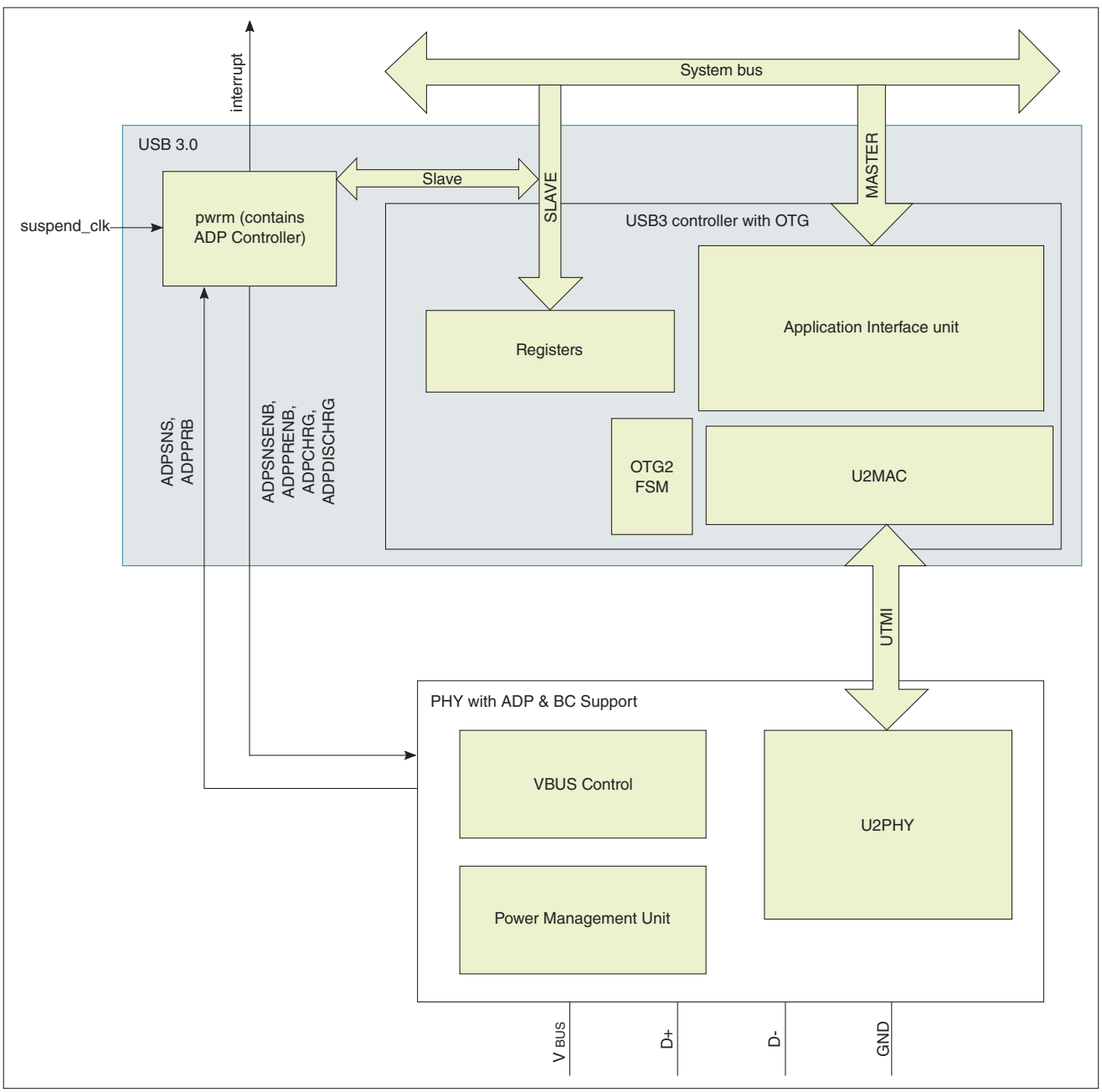


Figure 33-20. Internal ADP controller

For the PHY to generate ADPPRB and ADPSNS, the PHY needs to know the value to compare with VADP_PRB and VADP_SNS respectively. These values can be hardcoded in the PHY or the PHY can have a limited programmable option by which these reference voltage values can be changed. It is assumed that the PHY takes care of this. The pwrm module also implements SRP detection in A-Device mode. This is required if pwrdwn module of the USB 3.0 controller is powered down completely while ADP is in progress and B-Device initiates SRP.

NOTE

- The ADP block works with the suspend clock.
- The ADP timers run with the suspend clock along with the programmed USB_GCTL[PWRDNSCALE] value. The USB_GCTL[PWRDNSCALE] value is used internally to generate 32 KHz pulse for the timers.

33.4.6.1.3 Software flow

The software flow diagram is illustrated in the following figure.

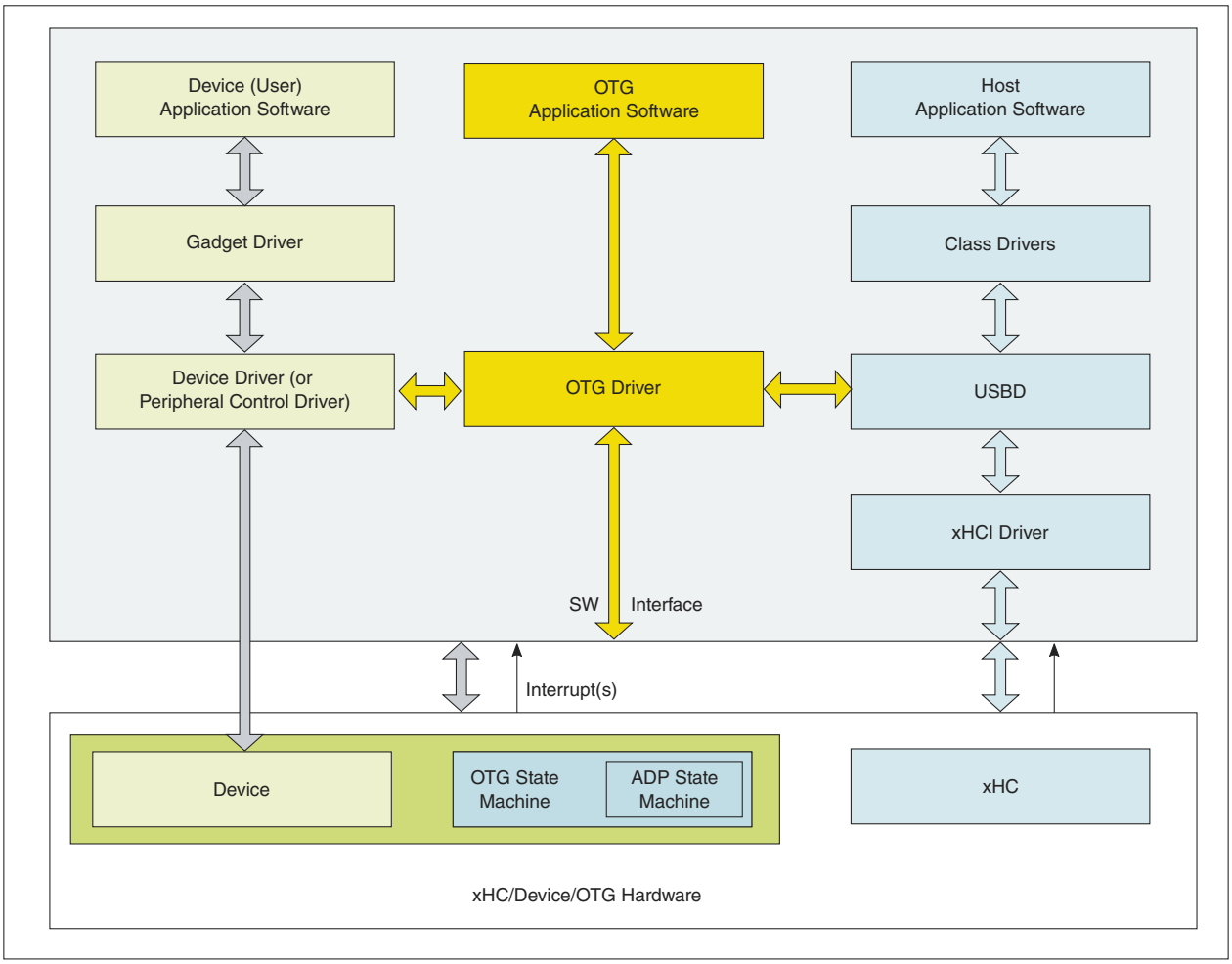


Figure 33-21. Software flow diagram

The following two sections describe the A-device and B-device flows briefly with respect to the above flow diagram.

NOTE

The following concise flows are not very detailed, and are provided here to help understand at an abstract level.

33.4.6.1.3.1 A-device activity concise flow

- Power-on-reset
- Only the OTG driver is active after power-on-reset. The host and device drivers are inactive after power-on-reset since at this point of time it is unsure what role the device assumes.
- The OTG driver initializes ADP and waits until a successful ADP event is received.
- The OTG driver senses ID pin = 0 and initializes the OTG registers.
- The OTG driver waits until a successful SRP event is received and turns on the VBUS.

- The OTG driver requests the USBD (USB driver) to bring up the xHC.
- After successful xHC bring up, device enumeration (and data transfers) occurs during which the device capabilities are exchanged for determining a possible role switch.
- In the event where a role switch is possible, the OTG application software or the B-device may initiate a role switch (The OTG application software may request via the Host application software to create favorable conditions for a role switch).
- The xHC is shut down and the control is passed on to the OTG driver.
- The OTG driver requests the PCD (Peripheral driver) to bring up the Peripheral.
- After successful Peripheral bring-up, device enumeration (and data transfers) occurs.
- In an event where a role switch is possible, OTG application software may initiate a role switch (The OTG application software may request via the Device application software for a role switch).
- The PCD is shut down and the control is passed on to the OTG driver.
- The OTG driver requests the USBD to bring-up the xHC, thus reverting to the original roles.

33.4.6.1.3.2 B-device activity concise flow

- Power-on-reset
- Only the OTG driver is active after power-on-reset. The host and device drivers are inactive after power-on-reset since at this point of time it is unsure what role the device assumes.
- The OTG driver initializes ADP and waits until a successful ADP event is received.
- The OTG driver senses ID pin = 1 and initializes the OTG registers.
- The OTG driver initiates a SRP and waits for the VBUS to be turned on by the A-device.
- The OTG driver requests the PCD to bring up the peripheral.
- After successful peripheral bring up, device enumeration (and data transfers) occurs during which the device capabilities are exchanged for determining a possible role switch.
- In an event where a role switch is possible, the OTG application software or the A-device may initiate a role switch (The OTG application software may request through the device application software for a role switch).
- The PCD is shut down and the control is passed on to the OTG driver.
- The OTG driver requests the USBD to bring up the xHC.
- After successful xHC bring-up, device enumeration (and data transfers) occurs.
- In an event where a role switch is possible, the OTG application software or A-device may initiate a role switch (The OTG application software may request the host application software for a role switch).
- The xHC is shut down and the control is passed on to the OTG driver.
- The OTG driver requests the PCD to bring-up the Peripheral, thus reverting to the original roles.

33.4.6.1.4 Programming model

This section describes the programming requirements for the USB 3.0 core in OTG mode. The core can be configured either as a DRD-device or an OTG, based on bit [13:12] of the Global Control Register's (GCTL) Power-On Initialization Value. If power-on configuration option is chosen as DRD-device, then PrtCapDir has to be explicitly programmed to 2'b11 for operating as an OTG device. The premise for the programming model is as follows.

1. The OTG driver is expected to run concurrently and independently with the PCD and xHCI drivers.
2. The OTG driver doesn't get involved in the actual data transfers but is responsible for communicating messages between the xHCI driver and PCD during role changes.
3. The OTG driver is responsible for handling the HNP, SRP and ADP (internal) events from the core. After power on, the OTG driver is responsible for loading, starting and switching between the xHCI driver and PCD. When started, the xHCI driver initializes the xHCI register set before enumerating the connected device. Similarly, when started, the PCD initializes the device register set and waits for the USB reset event to continue. Once the connection is established between the xHCI driver or the PCD by the OTG driver, the corresponding driver takes over for data transfers.

After every successful HNP switching, it is necessary that the active driver is changed from the xHCI driver to PCD or vice-versa. In such cases, the application may also unload the active driver from the memory and re-use the same memory area for loading the next active driver. The subsequent flow described in this section assumes that the core can get enumerated in any of the speeds (HS/FS) and bases its discussion accordingly. For example, if the core enumerates in HS/FS, follow the OTG 2.0 flow.

33.4.6.1.4.1 Initializing global registers

The global registers of the USB 3.0 core are shared between the device and host modes. It is the responsibility of the individual driver to initialize the core's global register specific to its functions. This is not explicitly mentioned in the flows described in this chapter. This specifically includes initializing the GTXTHRCFG/GRXTHRCFG, GEVNTEN, and GPRTBIMAP registers, and the Global FIFO Size and Global Event Buffer Registers. For more details, refer to Global registers in Table 33-2. However, these global registers that are programmed once at power on can be initialized in the global initialization routine in the OTG driver. Examples of one-time programmable registers are GSBUSCFG, GCTL, GSNPSID, and GUCTL. Examples of fields that are one-time programmable are the PHY interface (UTMI), AXI parameters like burst size, and so on.

The standard xHCI host driver does not get involved in the core-specific global register programming, and therefore, needs to be handled by the Board Support Package software.

33.4.6.1.4.2 Initializing host registers

In order to initialize the core as a host, the application should perform the steps described in the xHCI specification. Global registers need to be re-initialized as described in Initializing global registers.

33.4.6.1.4.3 Initializing device registers

In order to initialize the core as a device, the application should perform the steps described in the section Device Programming Model. Global registers need to be re-initialized as described in Initializing global registers. For specific registers, refer to Device Programming Model.

33.4.6.1.4.4 Initializing OTG registers

The application should initialize the OTG registers based on the initial value of the OSTS.ConIDSts after power on. The following sections depict the programming flow for the OTG application in detail.

33.4.6.1.4.5 Programming flow for OTG in USB 3.0

The following figure shows the programming flow after power-on reset.

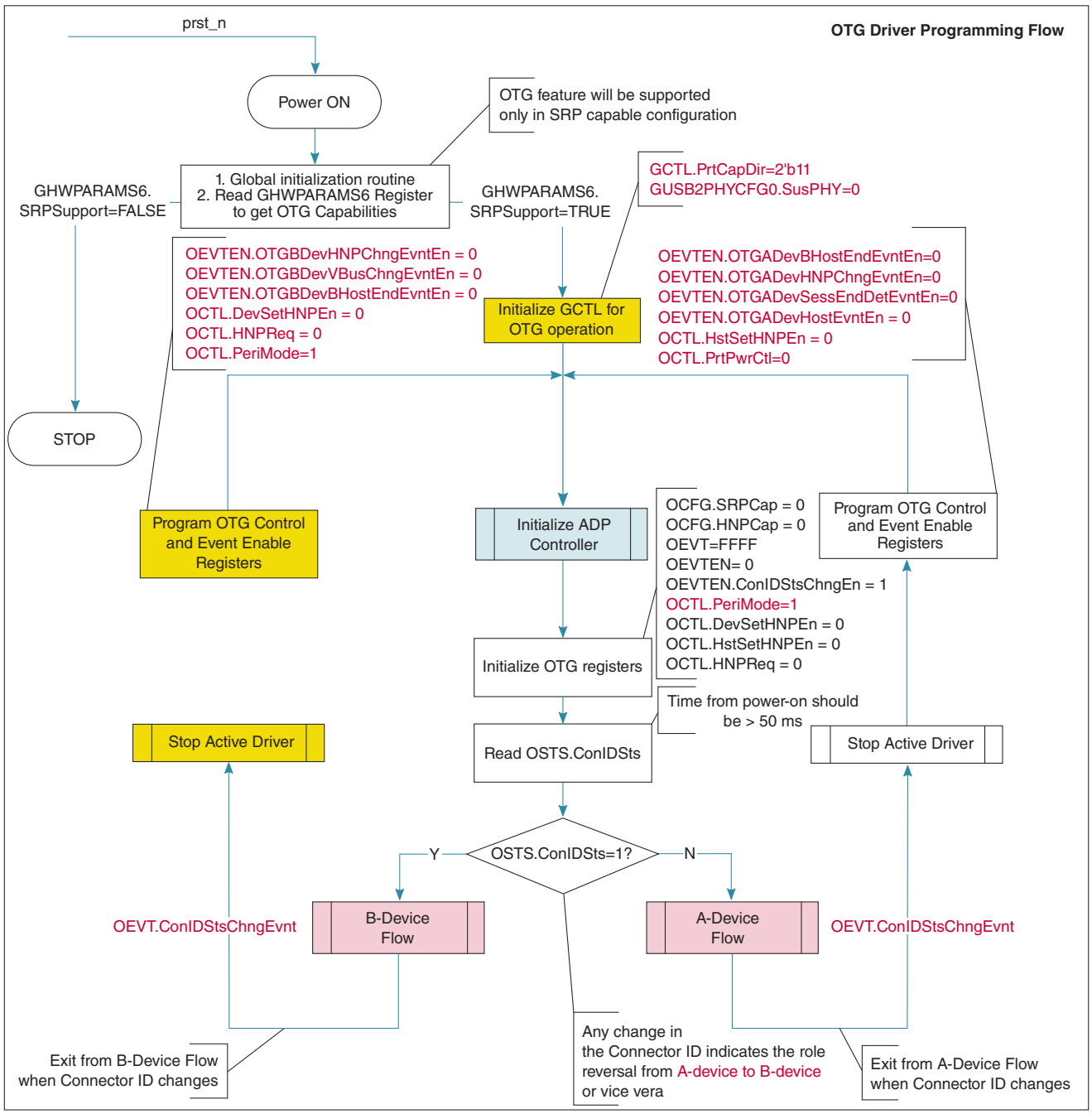


Figure 33-22. OTG driver overall programming flow

Following steps describe overall programming flow:

1. During global initialization, read the GHWPARAMS6 register to see if SRP support is enabled in the configured core. If SRP support is enabled, then proceed with the OTG programming flow by programming USB_GCTL[PRTCAPDIR] as 2'b11. If SRP support is not enabled, do not proceed further in the OTG programming flow. The configured device must at least support SRP for cable connection-based (Connector ID) role of operation.

2. Initialize the ADP controller as explained in ADP programming flow.
3. The OTG 2.0 state machine is initially in B-IDLE if IDDIG is 1, or A-IDLE if IDDIG is 0. Enable USB_OEVTEN[ConIDStsChngEvntEn] for any change in the Connector ID status.
4. Initialize the OTG control and configuration register to default values.
5. Read the OSTS register to find out the current status of the Connector ID (ConIDSts). After power-on or soft reset, the ConIDSts will be valid only after the PHY delay from IDPULL=1 to IDDIG active and any filter delay for IDDIG inside or outside the USB 3.0 core.
6. If USB_OSTS[ConIDSts] is 1, the OTG 2.0 state machine is in B-IDLE, follow the B-Device flow. Otherwise, if OTG 2.0 state machine is in A-IDLE, follow the A-Device flow.
7. When there is any connector ID (IDDIG) change resulting in USB_OEVT[ConIDStsChngEvnt], then exit the A-Device/B-Device flow. Stop the active driver and re-initialize the OTG control and status registers.

33.4.6.1.5 Common driver tasks

The common tasks for both A-Device and B-Device flow are as follows:

- Start xHCI driver
- Start peripheral control driver (PCD)
- Switch peripheral
- Switch host
- Stop active driver
- Enable HNP change

The Stop Active Driver is already introduced in the common flow diagram. Other tasks are useful for the A-Device/B-Device flows.

Start xHCI driver

The OTG driver, being responsible for starting the xHCI driver, executes this task. The OTG driver loads the xHCI driver after initializing the global registers, and the xHCI driver is responsible for all the functions thereafter.

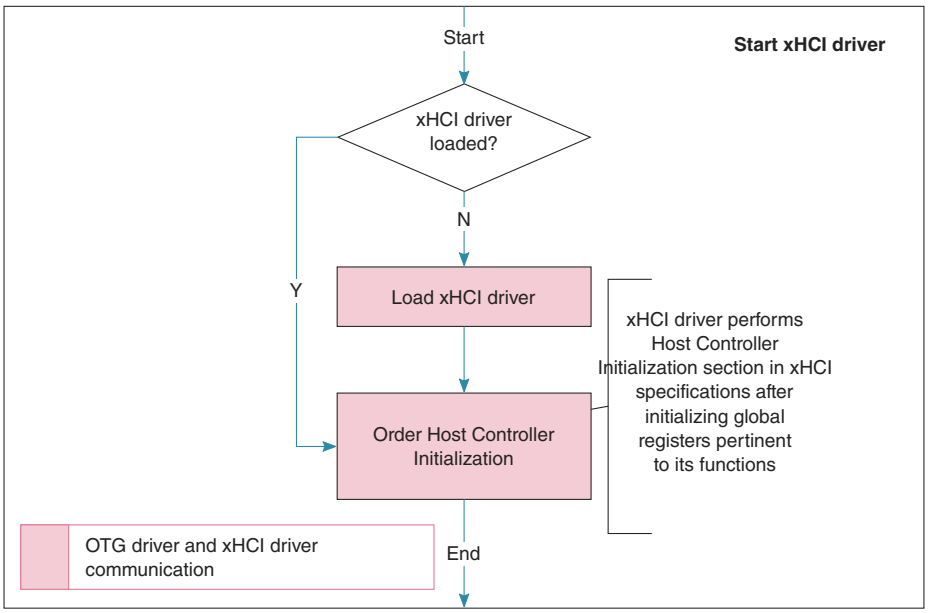


Figure 33-23. Start xHCI driver task

Start peripheral control driver

The OTG driver, being responsible for starting the Peripheral Control Driver (PCD), executes this task. The OTG driver only loads the PCD, which is responsible for all the functions thereafter, including the global registers initialization specific to the USB 3.0 core. Note that except the GTXFIFOSIZ and GEVNTADR, none of the other global registers need to be potentially re-programmed with different values.

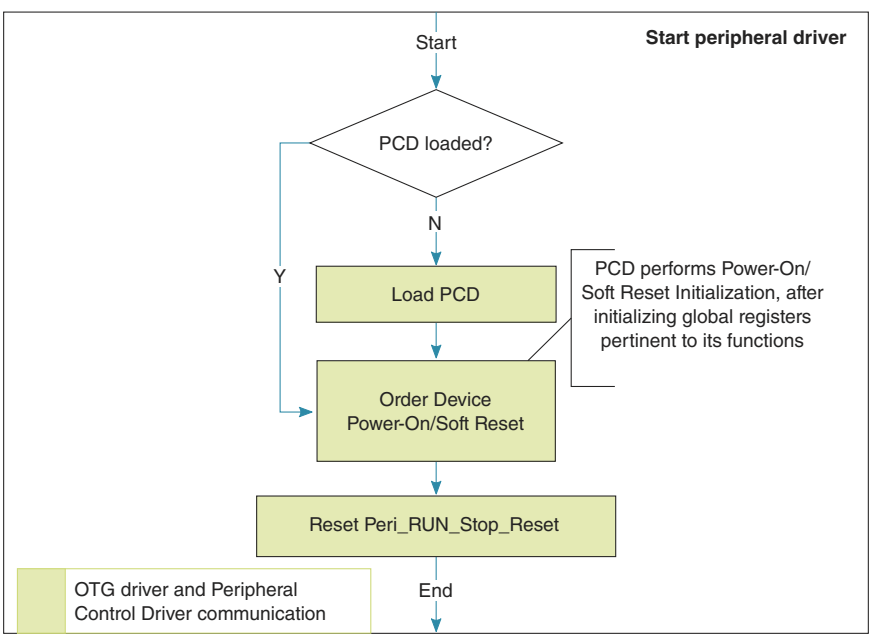


Figure 33-24. Start peripheral driver task

Stop active driver

This task checks which driver is currently active and stops the active driver. Optionally, it may unload the driver if the OS supports dynamic loading of drivers.

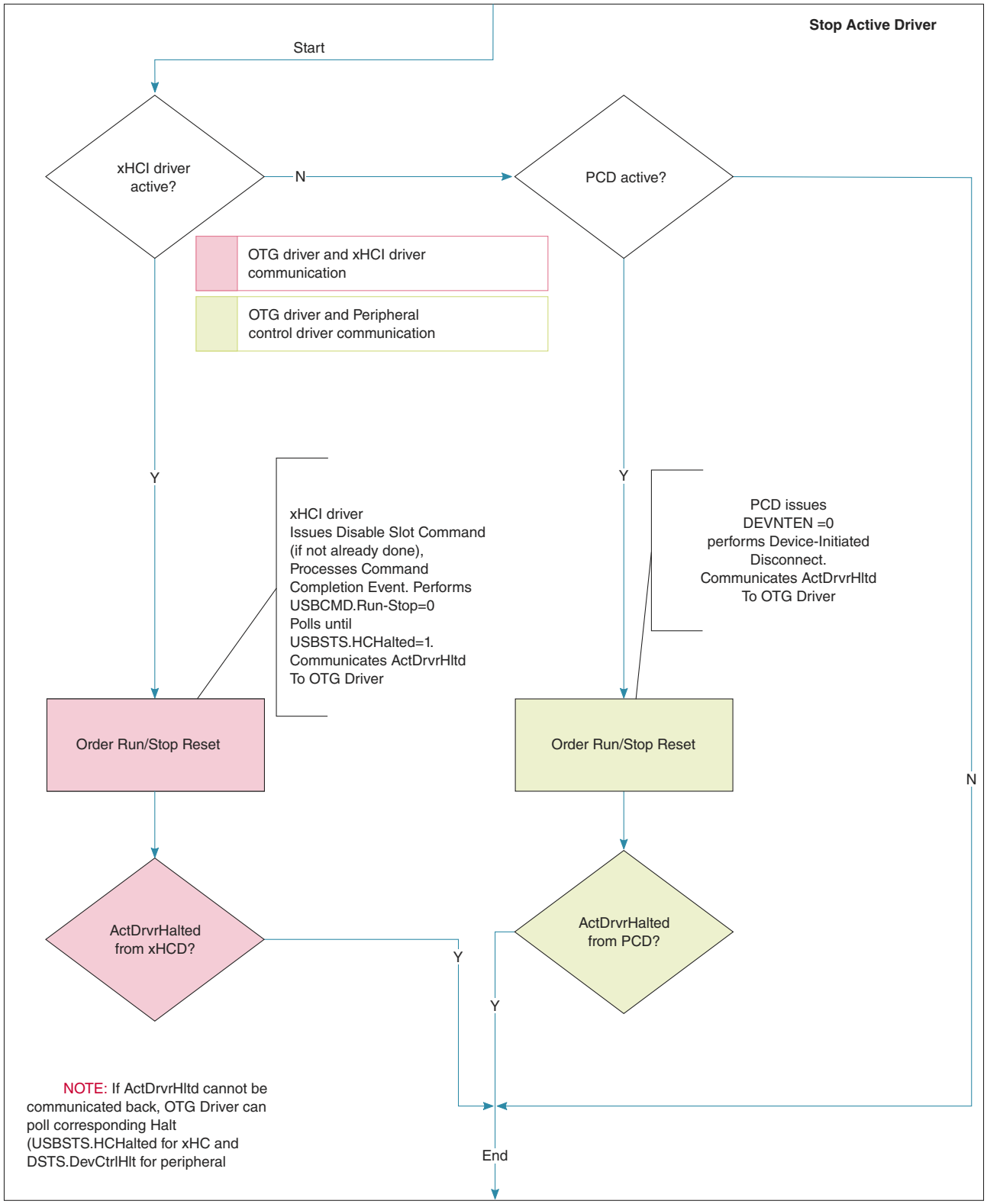


Figure 33-25. Stop active driver task

Switch peripheral and switch host

These tasks are used during HNP and each consists of two individual tasks.

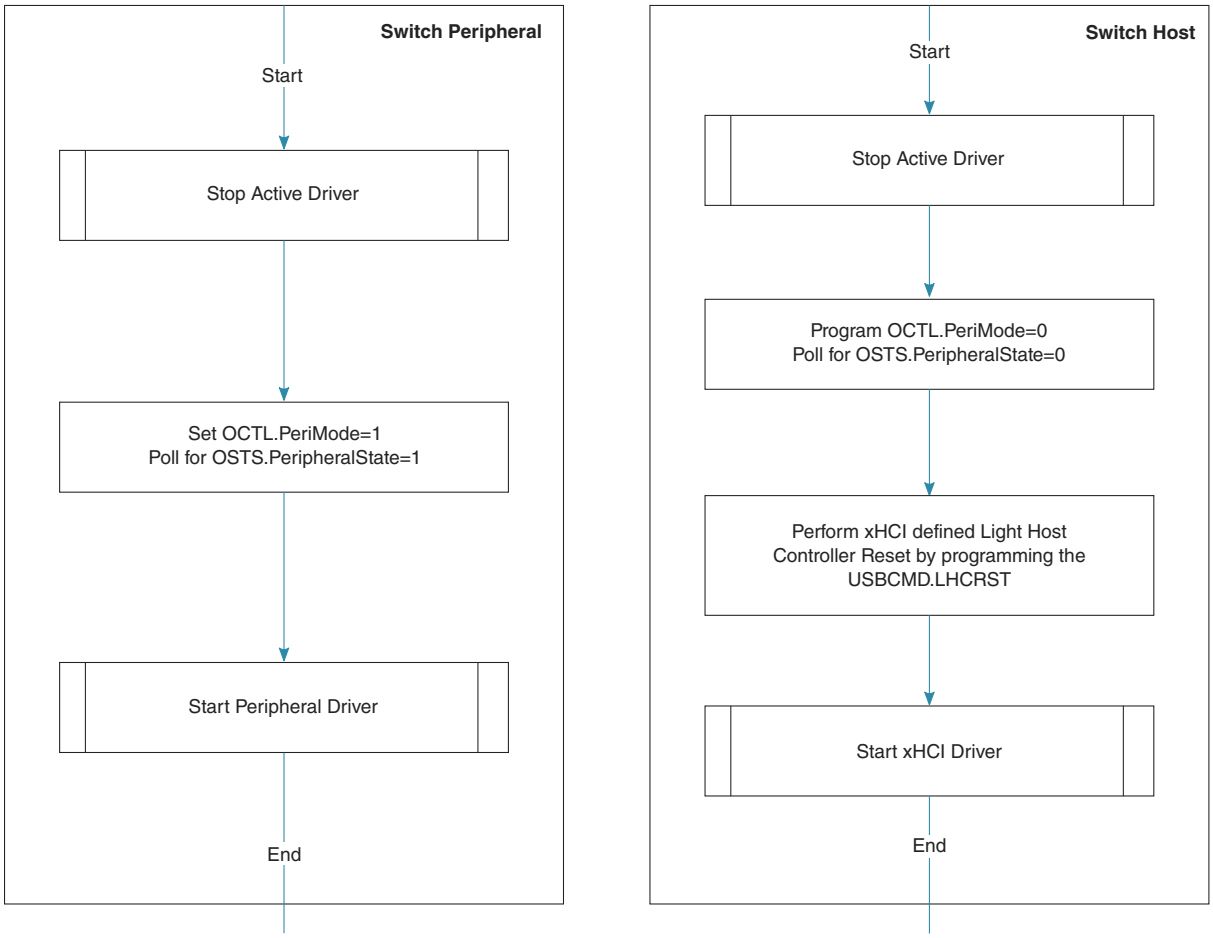


Figure 33-26. Switch peripheral task and switch host task

Enable HNP change

For HS/FS, this task is called when SetFeature.SetHNP is successfully exchanged between the host and the device. The set Peri_Run_Stop_Reset indicates to the device driver to automatically disable all events and perform a peripheral-initiated disconnect when the host puts the device in Suspend. Program the OCTL.DevSetHNPEn as 1, and OCTL.DevSetHNPReq as 1 immediately after the successful completion of SetFeature.SetHNP on the USB. This ensures the HNP attempt from the OTG core in the following Suspend state.

NOTE

By default, at power-on, program GUSB2PHYCFG.SusPHY as 1 to ensure that the PHY enters low power mode during normal suspend/resume without role switch.

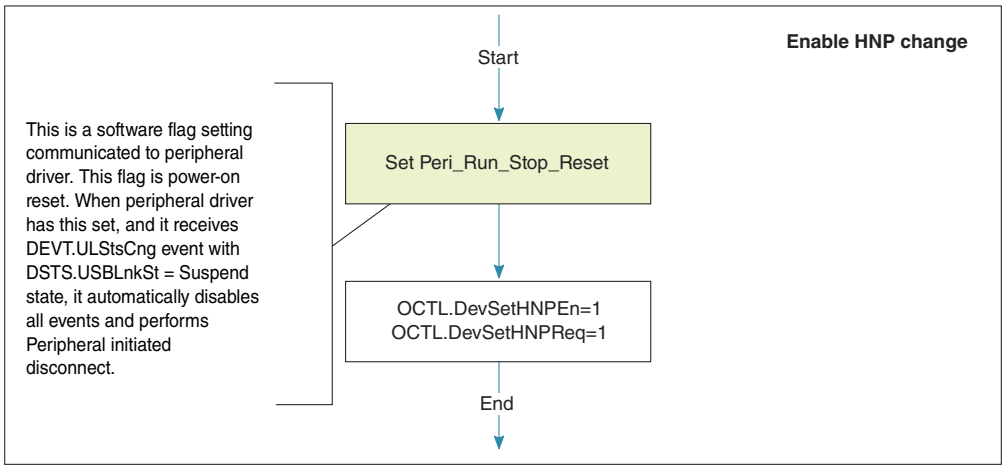


Figure 33-27. Enable HNP change

33.4.6.1.6 A-device flow

The following figure shows the A-device flow.

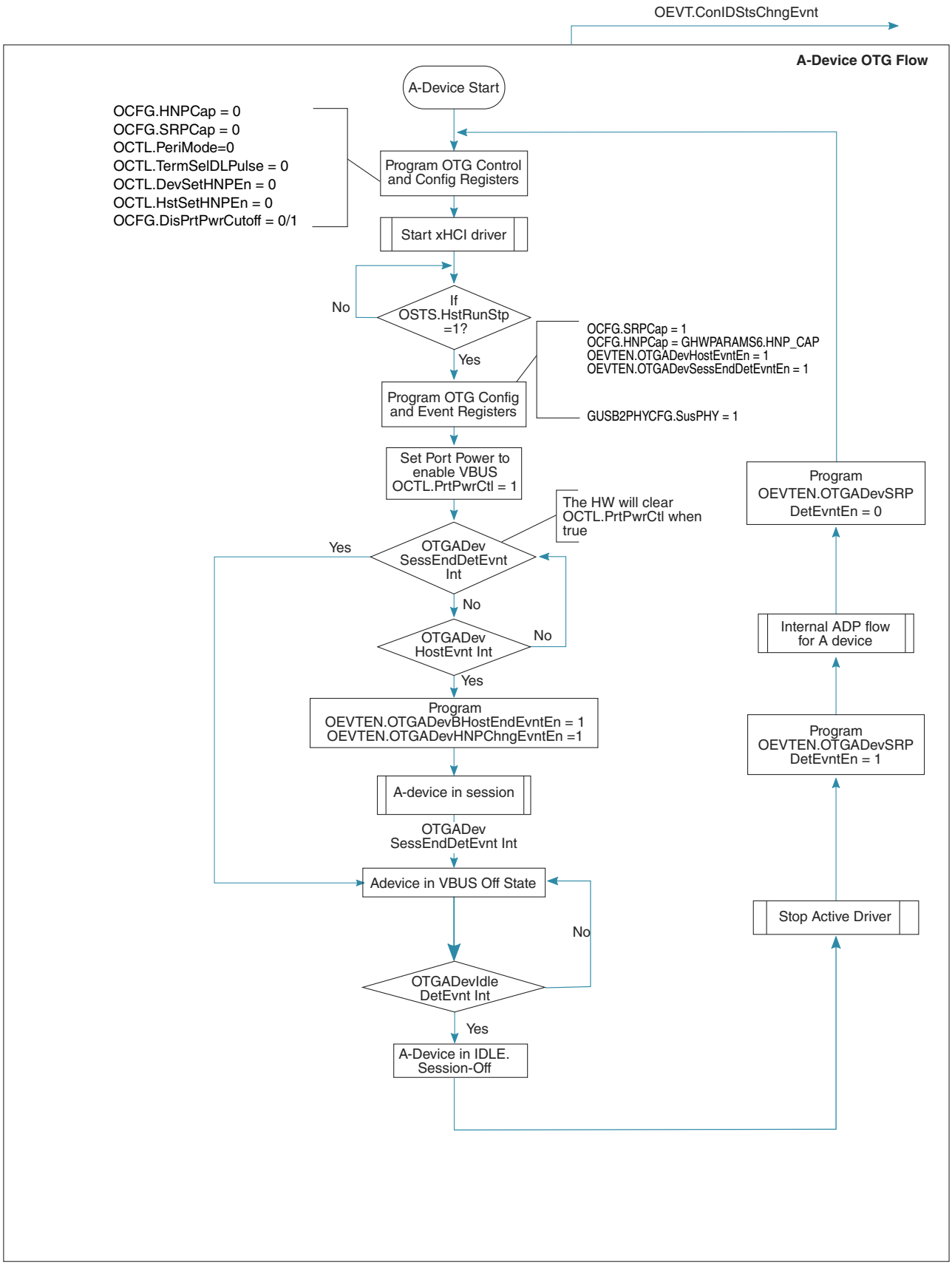


Figure 33-28. A-Device flow diagram

A-device flow:

1. At Start, the A-device core will be in A_IDLE VBUS off state.
2. The OTG software should program the OTG control and configuration registers for A-device operation:
   - OCFG[HNPCAP] = 0
   - OCFG[SRPCAP] = 0
   - OCTL[PERIMODE] = 0
   - OCTL[TERMSELDLPULSE] = 0
   - OCTL[DEVHNPEN] = 0
   - OCTL[HSTSETHNPEN] = 0
   - OCFG[DISPRTPWRCUTOFF] = 0 or 1, based on whether the application requires the core disable the feature to automatically switch off VBUS after 1 second if port is disconnected.
3. Start the xHCI driver.
4. The OTG software should wait for OSTS[HSTRUNSTP] to become 1 to continue. If OSTS[HSTRUNSTP] is 0, the OTG Software should wait for a software event that indicates the change in the xHCI RUN_STOP bit to continue. Alternatively, the OTG Software can poll for OSTS[HSTRUNSTP] bit to be set.
5. OTG Software should program the OTG configuration and event registers to enable A-device events to detect peripheral connect and session end. Refer to A-device flow diagram.
6. Program GUSB2PHYCFG0[SUSPHY] = 1 to enable the core to switch off the PHY clock in the possible non-HNP suspend scenarios.
7. Set OCTL[PRTPWRCTL] when OSTS[XHCIPRTPOWER] is 1. This enables driving of VBUS to the B-device.
8. The A-device may enumerate in HS/FS based on the device signaling.
9. The OTG 2.0 state machine enters a3_ds_host/A-host for HS (FS) if there is a successful B-device connect (ADEVHOSTEVNT).
10. The OTG software should enable the ADEVHNPCHNG and ADEVBHOSTEND events to detect the transition from A-host to A-peripheral and the end of A-peripheral events.
11. If there is an ADEVSESSENDDETEVNT interrupt indicating that the core has now stopped driving the VBUS, the software should wait for ADEVIDLEDETEVNT interrupt and then disable all the OTG events.
12. The Software can enable the ADP flow.
13. After an ADEVHOSTEVNT interrupt, the core is in session. The flow for the A-device in session is shown in earlier in this chapter.

NOTE

- At any step, if CONIDSTSCHNGEVNT occurs, A-device flow is terminated as per the figure "OTG Driver Overall Programming Flow".
- Over current is not detected by the OTG state machine. The xHCI controller detects it, and the OTG driver in turn sees an ADEVSESSENDDETEVNT (from PORTSC[PP] = 0) due to an overcurrent.

OTG state machine mapping to the software flow

- Steps 1-5 are A-IDLE.
- Step 7 is A_WAIT_VRISE in which A-device has detected SRP and therefore waits for VBUS to stabilize.
- Step 8 and step 10 are primarily A_HOST through A_WAIT_VRISE and A_WAIT_BCON. If there is an error in either A_WAIT_VRISE or A_WAIT_BCON, ADEVSESSENDDETEVNT occurs making the state-machine fall back to A-IDLE.

Handling over-current in OTG mode of operation

- When the core is operating in USB 2.0 mode, an over-current condition causes PORTSC[OCA] = 1 and PORTSC[OCC] = 1 on the corresponding USB 2.0 port number. The core also updates a port status change event TRB for the corresponding USB 2.0 port number.
- When the core is operating in USB 3.0 mode, an over-current condition causes PORTSC[OCA] = 1 and PORTSC[OCC] = 1 on the corresponding USB 2.0 and USB 3.0 port numbers. The core also updates a port status change event TRB, one each for the corresponding USB 2.0 port number and the corresponding USB 3.0 port number.
- Note that PORTSC[OCA] may transition quickly from 0 to 1 and 1 to 0 before xHCI driver can see two distinct interrupts. This may occur due to over-current condition being removed quickly. In such a case, the xHCI driver will receive a single interrupt (PORTSC[OCC]) for over-current.

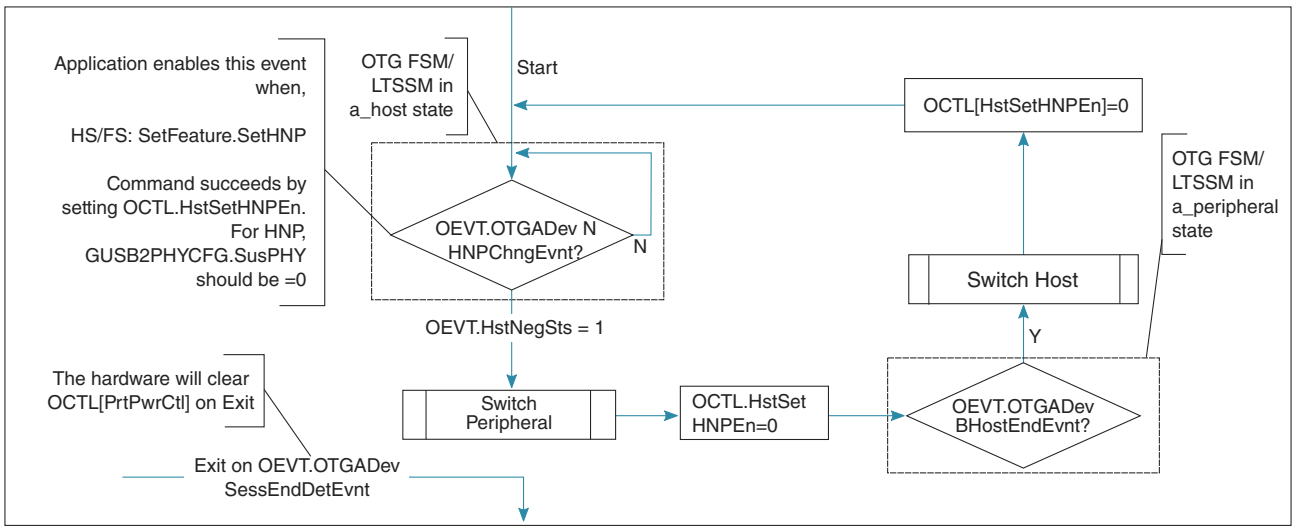


Figure 33-29. A-host -> A-peripheral -> A-host flow diagram

For HS/FS mode:

1. When the core starts the session with A-device as host, the application may periodically exchange SetFeature.SetHNPEnable packets. If B-device wants to become the host, this packet exchange results in the host application knowing about B-Device's intent to become the host, and it directs the OTG driver to set OCTL[HSTSETHNPEN]. This enables the core to cater for HNP from B-device when suspend is initiated. During suspend, ADEVHNPCHNGEVNT is set.
2. If ADEVHNPCHNGEVNT is set, the OTG driver reads OEVT[HSTNEGSTS]. If OEVT[HSTNEGSTS] is set, then the core transitions to A-Peripheral state by executing Switch Peripheral task. Otherwise, wait for ADEVSESSENDDETEVNT and go to ADP probing.
3. When in A-peripheral state, if there is ADEVBHOSTENDEVNT, it signifies that the B-device no longer wants to be the host. Therefore, the driver goes back to Step 1 by executing the switch host task and assumes an A-host role.

   Alternatively, if there is an xHCI[PRTPOWER] de-assertion, the state machine goes to start ADP probing due to the occurrence of ADEVSESSENDDETEVNT.

4. If ADEVBHOSTENDEVNT is set, then the core would be in A-host state by executing switch host task. Otherwise, perform either of the following:
   a. Drop VBUS and go to ADP probing.

      If the role change is successful, the OTG driver hands over the control to the xHCI driver.

Internal ADP flow for A-device

The internal ADP flow of A-device is as follows:

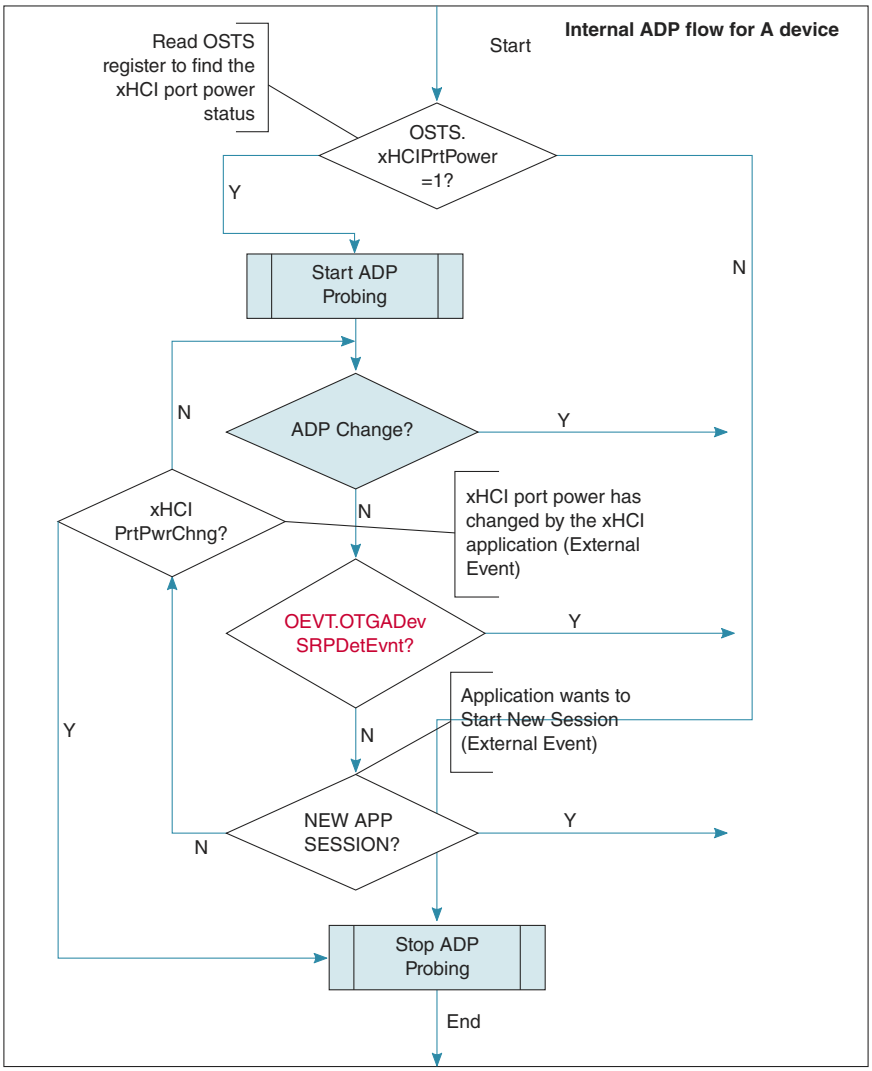


Figure 33-30. Internal ADP flow for A-device

- ADEVSESSENDDETEVNT occurs due to:
  - PORTSC[PRTPOWER] = 1'b0
  - OTG driver stopping VBUS due to:
    - BCON_TOUT when in A_WAIT_BCON
    - A_AIDL_BDIS_TOUT when in A_SUSPEND
- At any step, if ADEVSESSENDDETEVNT occurs, the software starts with ADP probing.
- Once the ADP probing (both external and internal) is started, the software should wait for one of the following events to occur (to stop ADP probing and proceed to Step 2 in A-device flow) to continue.
  - ADP change event
  - SRP detect event
  - New session event from A-device application
  - xHCI port power change event from A-device application

33.4.6.1.7 SRP detection by the core (Timeline for ADevSRPDetEvnt)

ADevSRPDetEvnt is triggered by a valid DLINE pulsing on the USB D+ line by B-Device. The A-Device looks at both the positive and negative edge transition occurrence for a valid data-line pulse time before detecting SRP.

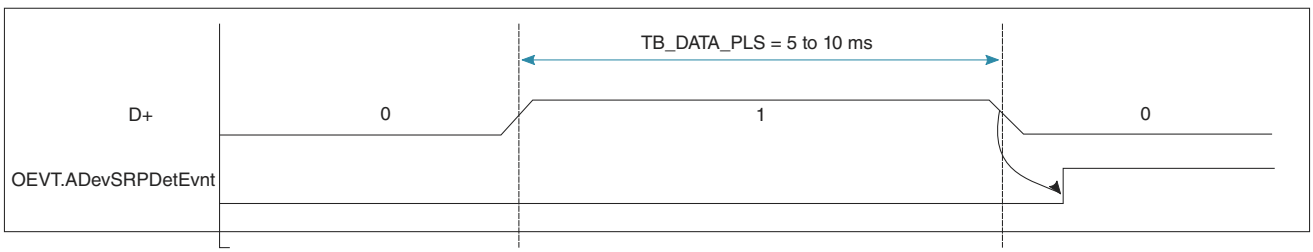


Figure 33-31. Timeline for ADevSRPDetEvnt

33.4.6.1.8 VBUS turned ON by the core (Timeline for ADevBSessEndEvnt)

When the A-device turns on the VBUS, it is unsure at this point of time at what speed the USB enumerates to. So, both the LTSSM and the OTG 2.0 FSMs start concurrently. The LTSSM proceeds with the training sequence after Rx_detect and the OTG 2.0 FSM waits for A_WAIT_BCON.

HS/FS Mode: Before entering A-Host, there is a check done in the OTG state machine to ensure B-Device is connected. If B-Device is not connected within TA_WAIT_BCON, ADevBSessEndDetEvnt is set. The core resets OCTL.PrtPwrCtl, and the OTG state machine enters A_WAIT_VFALL. Then VBUS to B-Device is removed and this results in ADevSessEndEvnt. Note that the A_WAIT_VFALL transition can occur due to over current, which is not depicted in the following figure.

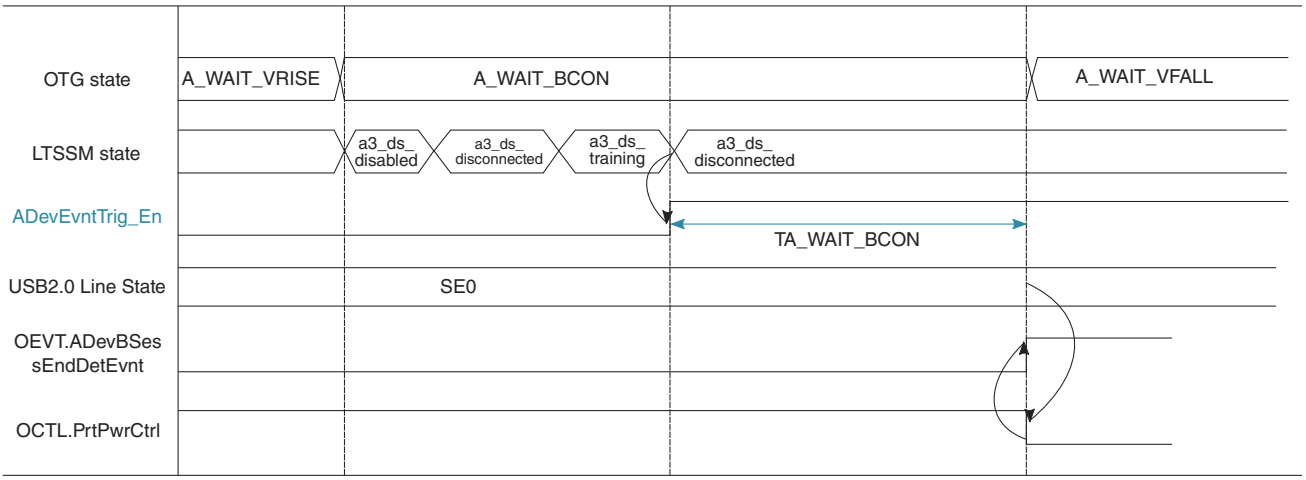


Figure 33-32. Timeline for ADevBSessEndEvnt

33.4.6.1.9 Core entering A-Host in HS/FS mode (Timeline for ADevBHostEvnt)

HS/FS Mode: If the B-Device is connected within TA_WAIT_BCON, ADevBHostEvnt is set.

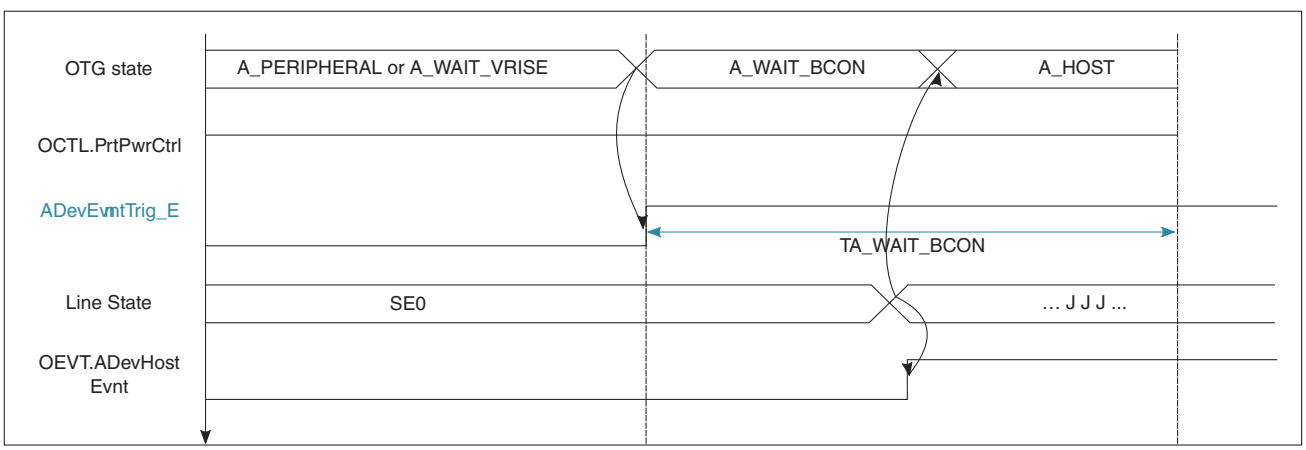


Figure 33-33. Timeline for ADevBHostEvnt

33.4.6.1.10 B-device flow

The following figure shows the B-device flow.

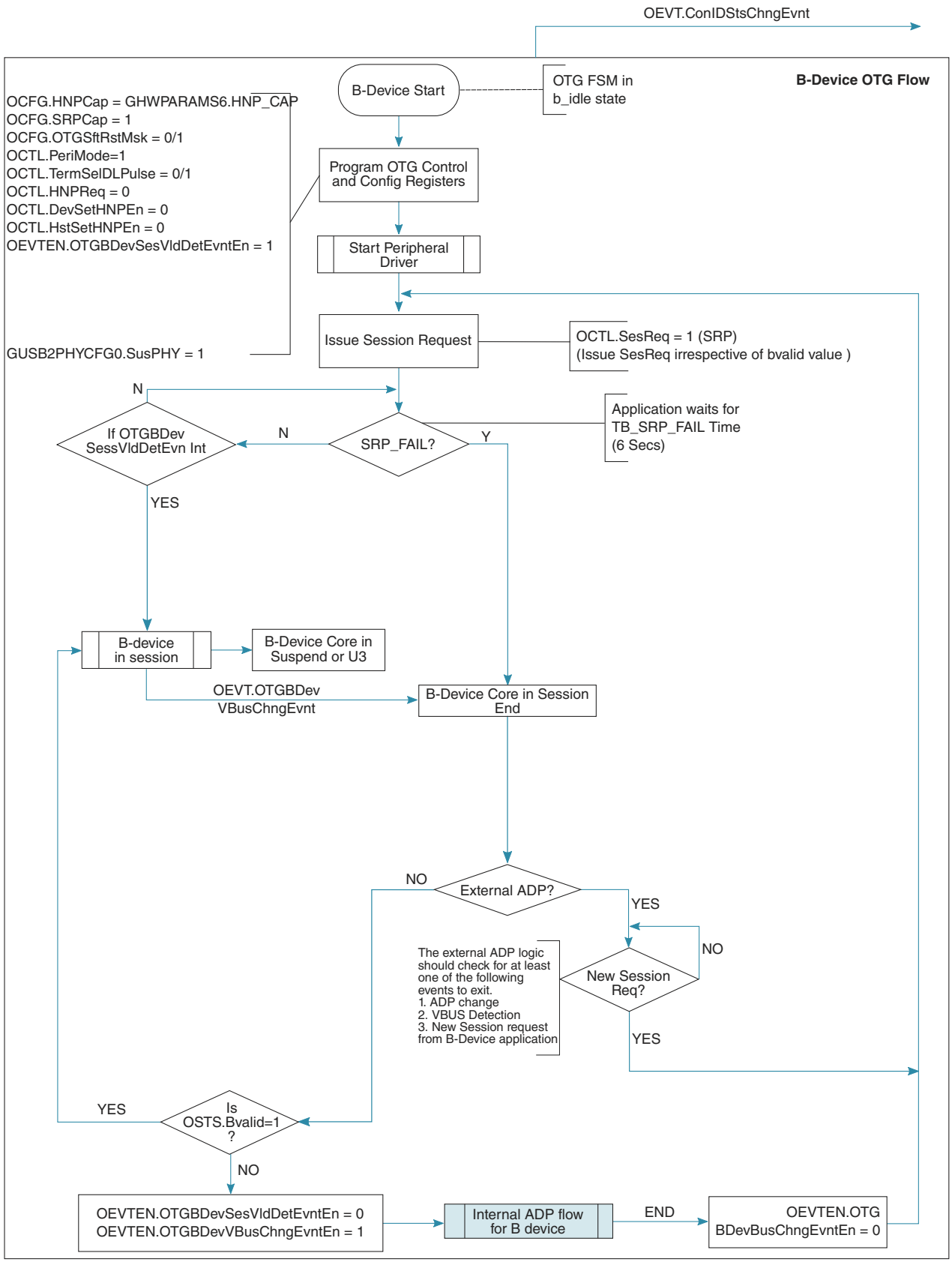


Figure 33-34. B-device flow diagram

1. At start, the core is in B-IDLE state. Note that for OTG functionality in peripheral mode, DCTL[RUN_STOP] should be set, otherwise the core as B-peripheral does not initiate SRP pulsing. Also, it is important for the device driver to program the speed of operation as B-peripheral (DCFG[SPD]).
2. The OTG software should program the following OTG control registers for B-device operation:
   - OCFG[HNPCAP] = GHWPARAMS6[HNPSUPPORT]
   - OCFG[SRPCAP] = 1
   - OCTL[PERIMODE] = 1
   - OCTL[TERMSELDLPULSE] = 0
   - OCTL[HNPREQ] = 0
   - OCTL[DEVHNPEN] = 0
   - OCTL[HSTSETHNPEN] = 0
   - OEVTEN[OTGBDEVSESVLDEVNTEN] = 1
   - GUSB2PHYCFG0[SUSPHY] = 1
3. Start peripheral driver.
4. Program OCTL[SESREQ] = 1 to initiate SRP by D-line pulsing. The core initiates SRP and waits for a valid VBUS (BVALID).
5. The B-device core continues to wait for BVALID if the A-device does not respond. The application should timeout if SESSVLDDETEVNT does not come within the SRP fail time (TB_SRP_FAIL is 6 seconds).
6. When SRP fails, the application should enable ADP probing.
7. If the A-device asserts a valid VBUS in response to SRP, the core reports an BDEVSESSVLDDETEVNT to indicate the start of a session.
8. If there is a BDEVSESSVLDDETEVNT, the core enters a session on state. For HNP flows in B-peripheral -> B-host -> B-peripheral flow diagram.
9. If there is a BDEVVBUSCHNGEVNT anytime, indicating VBUS is no longer valid, the core enters session-end state.

NOTE

- For OTG 2.0, during HNP process where the B-device is going to assume the role of a host, the B-device application needs to ensure that a USB reset process is programmed within 150 ms (TB_ACON_BSE0) of getting a device connect interrupt.
- At any step, if CONIDSTSCHNGEVNT occurs, the A-device flow is terminated as per "A-device Flow Diagram".

The OTG state machine mapping to the software flow is as follows:

- Steps 1-3 and 6 are B-IDLE
- Step 4 is B-SRP-INIT
- Step 7 and 8 are B-PERIPHERAL

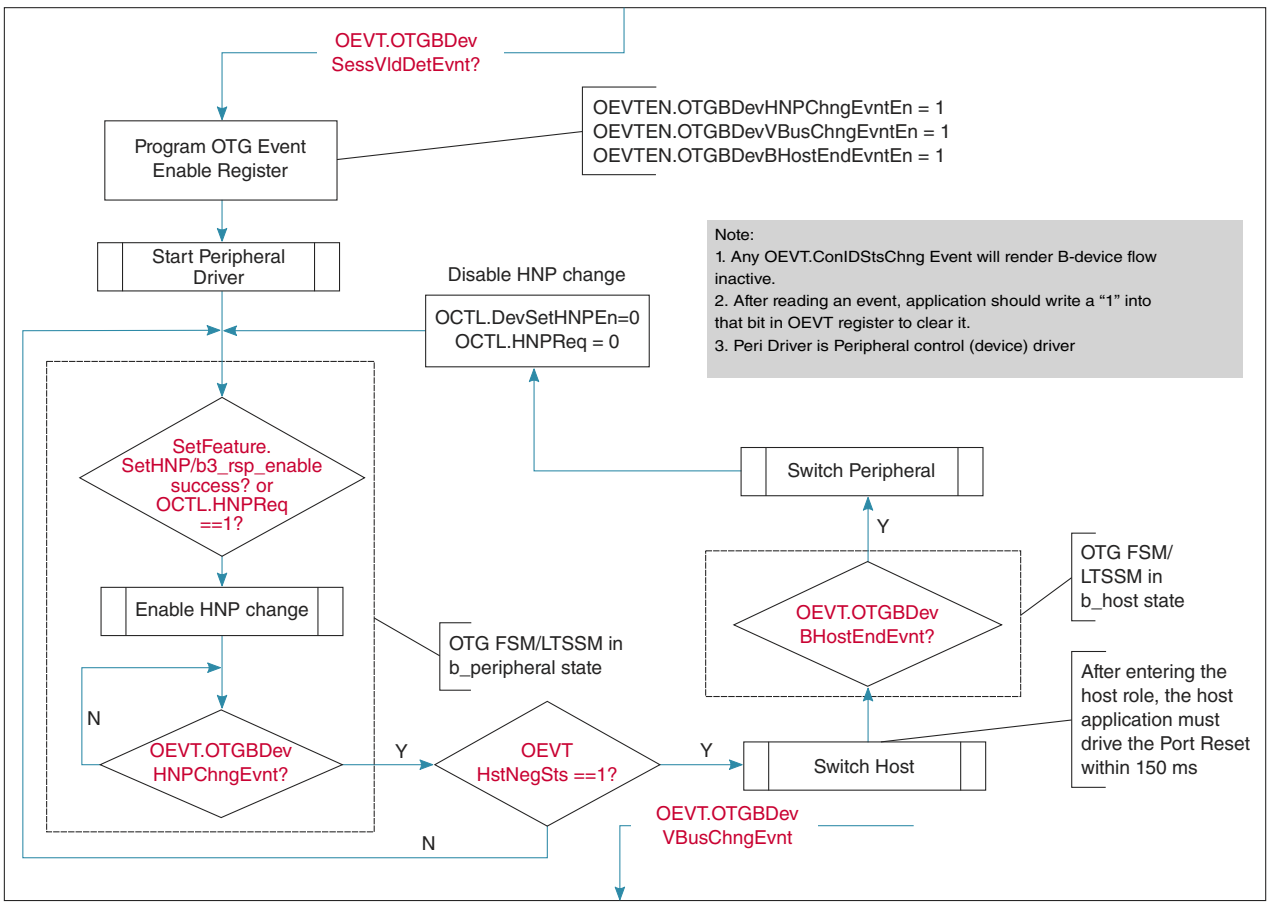


Figure 33-35. B-peripheral -> B-host -> B-peripheral flow diagram

For HS/FS mode:

1. The OTG state machine starts operation as a B-peripheral. The following events are enabled.
   - OEVTEN[OTGBDEVHNPCHNGEVNTEN] = 1
   - OEVTEN[OTGBDEVVBUSCHNGEVNTEN] = 1
   - OEVTEN[OTGBDEVBHOSTENDEVNTEN] = 1

   In this state, the application periodically exchanges SetFeature.SetHNPEnable packets. If the B-device wants to become host, this packet exchange results in a device application sending an ACK to this packet, and it directs the OTG driver to execute Enable HNP change. Ensure that GUSB2PHYCFG0[SUSPHY] is 1'b0 when SetFeature.SetHNP command succeeds. This enables the core to initiate HNP from B-device when a suspend is initiated. During a suspend, BDEVHNPCHNGEVNT is set. OTGBDEVHNPCHNGEVNT starts the transition from B-peripheral ->B-WAIT-ACON to either B-host or B-peripheral.

2. If BDEVHNPCHNGEVNT is set, the OTG driver reads OEVT[HSTNEGSTS]. If OEVT[HSTNEGSTS] is set, it indicates successful connection of A-device as

peripheral, the core therefore transitions to B-host state. After entering the host role, the host application must drive the port reset within 150 ms. If OEVT[HSTNEGSTS] is not set, it indicates failure of HNP and it stays as B-peripheral.

3. When in B-host state, if there is BDEVBHOSTENDEVNT, it signifies that the B-device no longer wants to be the host, therefore suspended the bus and A-device has stopped signaling connect. Therefore, the driver goes back to step 1 and assumes B-peripheral role.
4. The core generates BDEVBHOSTENDEVNT.

NOTE

- Step 3 above is B-host and B-host to B-peripheral.
- OTGBDEVVBUSCHNGEVNT indicates transition back to B-IDLE.

NOTE

- Step 3 above is B-host and B-host to B-peripheral.
- OTGBDEVVBUSCHNGEVNT indicates transition back to B-IDLE.
- If there is any unexpected error scenario like VBUS drop after the OCTL[DEVSETHNPEN] bit is set, the OTG driver must clear the OCTL[DEVSETHNPEN] bit.

Internal ADP flow for B-device

The internal ADP flow of B-device is as follows:

Figure 33-36. Internal ADP flow for B-device

While probing, the application should wait for one of the following events to exit probing and start SRP request again.

- ADP change
- VBus detection
- New session request from B-device application

While sensing, the application should wait for one of the following events to exit sensing.

- ADP sensed failed
- VBUS detection
- New session request from B-device application

The application should continue with ADP probing if ADP sense failed. Else, the application should exit sensing and start SRP request again

33.4.6.1.11 Core entering b3_us_peripheral in B-Peripheral in HS/FS mode (Timeline for BDevSessVldDetEvnt)

The PCD decides the speed in which the USB3 core needs to operate by programming DCFG.DevSpd bits. Accordingly, the core decides to operate in FS to start with.

HS/FS Mode: BDevSessVldDetEvnt is triggered by a valid VBUS detection by B-Device in B-IDLE state. The B-Device will transition to B-PERIPHERAL state before triggering this event. Note that this is common to both SS and HS(FS).

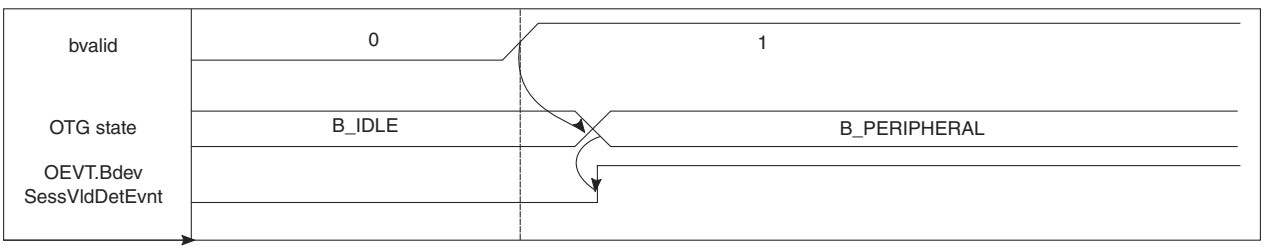


Figure 33-37. Timeline for BDevSessVldDetEvnt in HS/FS mode

33.4.6.1.12 VBUS change detected on USB (Timeline for BDevVBusChngEvnt)

BDevVBusChngEvnt is triggered when a change in bvalid is detected irrespective of OTG state machine.

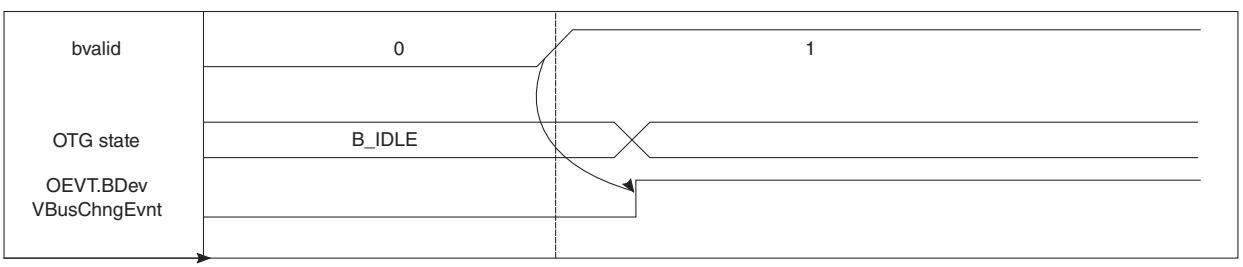


Figure 33-38. Timeline for BDevVBusChngEvnt

33.4.6.1.13 Internal ADP controller logic

The ADP controller has timers that assist in ADP operation. The ADP-related registers are a part of the pwrm_otgif module. For the PHY to generate ADPPRB and ADPSNS, the PHY needs to know the value to compare with VADP_PRB and VADP_SNS, respectively. These values can be hardcoded in the PHY or the PHY can have limited programmable option by which these reference voltage values can be changed. The

"Wakeup Logic" module implements SRP detection in A-Device mode. This is required if the USB 3.0 controller is powered down completely while ADP is still in progress and B-Device initiates SRP.

NOTE

ADP controller logic is inferred along with, but outside USB 3.0 controller.

- As a product, both ADP controller logic and OTG controller are packaged into USB 3.0 core.
- All ADP timers are maintained in the ADP controller module, which generates the enable and disable signaling to the PHY for ADP probing and sensing.
- PHY contains the following circuitry related to ADP functionality:
  - Comparators for PRB and SNS
  - I_ADP_SRC and I_ADP_SNK
  - V_{BUS} circuitry
- Application software programs the ADP timing registers that resides in the pwrm_otgif module.
- The pwrm_otgif module provides a mechanism to the application via interrupts to log and report events pertaining to ADP probing and sensing.
- In this operation, only the pwrm module needs to be always powered on. The pwrdwn module can be either powered on or off.

33.4.7 Initialization/application information

The PHY can be configured for fixed equalization by programming relevant control registers in the USB 3.0 PHY.

In order to initialize the USB 3.0 PHY, software should perform the following steps:

1. Write 1'b0 to RX_OVRD_IN_HI.RX_EQ_EN [address 16'h1006: bit 6].
2. Write 1'b1 to RX_OVRD_IN_HI.RX_EQ_EN_OVRD [address 16'h1006: bit 7].
3. Write a fixed value to RX_OVRD_IN_HI.RX_EQ [address 16'h1006: bits 10–8] [equalization setting] (generally 2–4, based on testing in customer environment).
4. Write 1'b1 to RX_OVRD_IN_HI.RX_EQ_OVRD [address 16'h1006: bit 11].

33.4.8 Power management overview

The following power management features are available in the USB 3.0 core:

- Hardware-controlled LPM in USB 2.0 host
- Clock gating in device (U1/U2/U3) and host (U1/U2/U3)

The power saving mechanism is hardware-controlled LPM. This is a USB 2.0 host-only feature where the host controller automatically detects the downstream port is idle for a certain amount of time and autonomously initiates an LPM token to the connected device. If the device accepts, the link is put into the standard USB 2.0 Suspend state.

Finally, clock gating is also supported in the host controller as well as in the device controller. The following table gives the list of power saving mechanisms available in USB 3.0 core.

Table 33-29. Power savings support

USB 3.0 State	USB 2.0 State	Device Power Savings	Host Power Savings
U1 (Hardware Initiated)	None	Core: Clock Gating PHY: P1	Core: Clock Gating PHY: P1
U2 (Hardware Initiated)	LPM-L1 (Hardware Initiated)	Core: Clock Gating, PHY: P2/Sleep	Core: Clock Gating, PHY: P2/ Sleep
U3 (Software Initiated)	Suspend (Software Initiated)	Core: Clock Gating, PHY: P3/Suspend	Core: Clock Gating, PHY: P3/ Suspend
Rx.Detect (Software Initiated)	None	None	Core: Clock Gating, PHY: P3/ Suspend
SS.Disabled (Software Initiated)	None	Core: Clock Gating, PHY: P3/Suspend	None

33.4.8.1 Clock gating

The core implements clock gating in the following situations:

- In USB 2.0 device mode
  - When the UTMI suspend or l1_suspend signal is asserted and the core is idle
- In USB 3.0 device mode, when the link is in U1, U2 or U3 state and the core is idle.

In host mode, clock gating is enabled in the following situations:

- When all USB 2.0 ports are in the suspend or L1 Suspend state, and all USB 3.0 ports are in the U1/U2/U3 state, and the core is idle.

Internal clock gating will apply to:

- Modules that use the ram_clk
- Some modules that use bus_clk

Internal clock gating will not apply to:

- Modules that use mac3_clk: When the USB 3.0 PHY is suspended, these modules switch to using suspend_clk, so their power consumption is reduced but the clock is not completely stopped.
- Modules that use mac2_clk: When the USB 2.0 PHY is suspended, these modules switch to using suspend_clk, so their power consumption is reduced but the clock is not completely stopped.
- The bus_gs module, which uses bus_clk. This module detects wakeup on the slave interface and thus needs a non-gated clock.

Clock gating can be enabled or disabled using the GCTL register. The GCTL register is "sticky," it retains its value across soft resets.

33.4.8.2 Hardware-Controlled LPM

In USB 2.0, the main link power management mechanism was for the host to suspend the link. However, suspending the link requires a long period of idle and was wasteful of power, especially because the host knows ahead of time whether it is going to schedule any traffic. It also required a long period of resume signaling.

The USB 2.0 Link Power Management Addendum [LPM-ECN] added a new token to the USB 2.0 specification, the LPM token. The host has the option of suspending the link quickly instead of waiting for a long period of idle, and communicates its desire to the device by sending an LPM token. The ECN also introduced a terminology for different link states. This new link state was called L1. The traditional USB 2.0 suspend was called L2. The L1 state requires a much shorter resume signaling period too.

An xHCI driver can initiate LPM by writing '2' to the PORTSC.PLS field. However, there is a lot of overhead for an xHCI driver to be constantly trying to keep the link in a low power state. Because of this, the xHCI specification also allows a compatible xHC to be enabled with "Hardware-Controlled LPM" by setting PORTPMSC.HLE as 1. When this happens, the xHC is expected to automatically send LPM tokens when it has no pending transfers and thinks that the link is idle enough to enter L1. This feature is supported by the host core.

33.4.8.2.1 Special consideration for OTG

When the core is configured as an OTG device, it will not initiate LPM when acting in a reverse role as a B- Host. This is because suspend on the link is treated as a request to switch back to its original role.

33.4.8.3 USB PHY power gating

The USB PHY power gating is used to save the leakage power of PHY if it is not required. Follow the steps given below to take PHY into power gating state:

1. Assert PHY_RESET by writing 1'b0 to the USB_PHY_CTRL[RST] bit.
2. Set the USB_PHY_CTRL[PDN_SSP] and USB_PHY_CTRL[PDN_HSP] to 2'b11.

NOTE

Both the above steps must have separate writes.

Chapter 34
Watchdog Timer (WDOG)

34.1 Overview

The Watchdog Timer (WDOG) protects against system failures by providing a method by which to escape from unexpected events or programming errors.

The chip has two WDOG timers. 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 timeout signal (wdog_rst_b) as reset request to COP.

The WDOG1 timer is used for A53 core. When WDOG1 timer expires, an interrupt is raised to GIC-400. If the interrupt is not serviced, the chip reset request is raised. The WDOG2 timer is used for TrustZone. When WDOG2 timer expires, an interrupt is raised to GIC-400. If the interrupt is not serviced by secure software, a security violation is raised to security monitor that results in a reset request.

Following figure shows system level representation of watchdog.

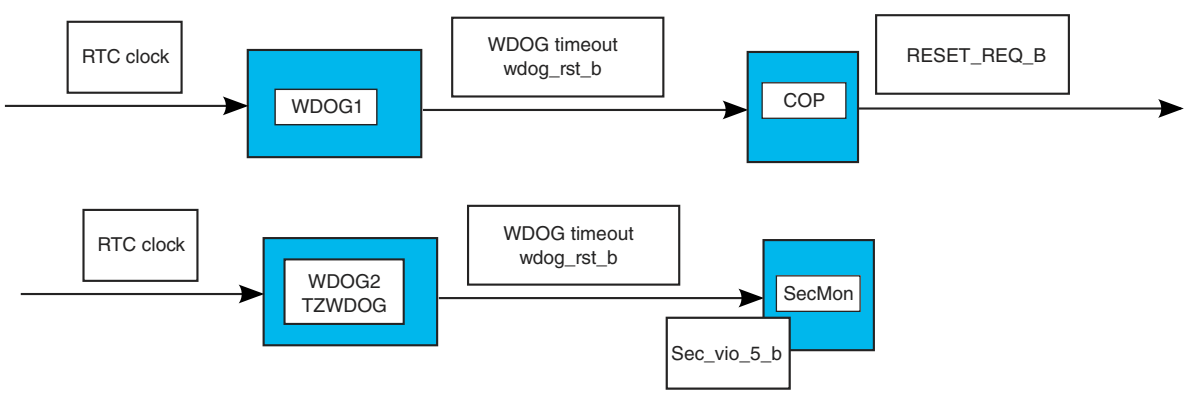


Figure 34-1. WDOG system level diagram

NOTE

There is no 1-to-1 relationship of WDOG counters with the cores. It is recommended that the secure-software should use

WDOG2, while the non-secure software should use the other WDOG block. During reset, all the WDOG blocks are disabled.

There is also a provision for WDOG signal assertion by time out 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 time out is programmable.

All WDOG modules use 32 KHz clock, driven at device input RTC pin for their counters.

Flow diagrams for the timeout counter and interrupt operations are shown in Figure 34-2

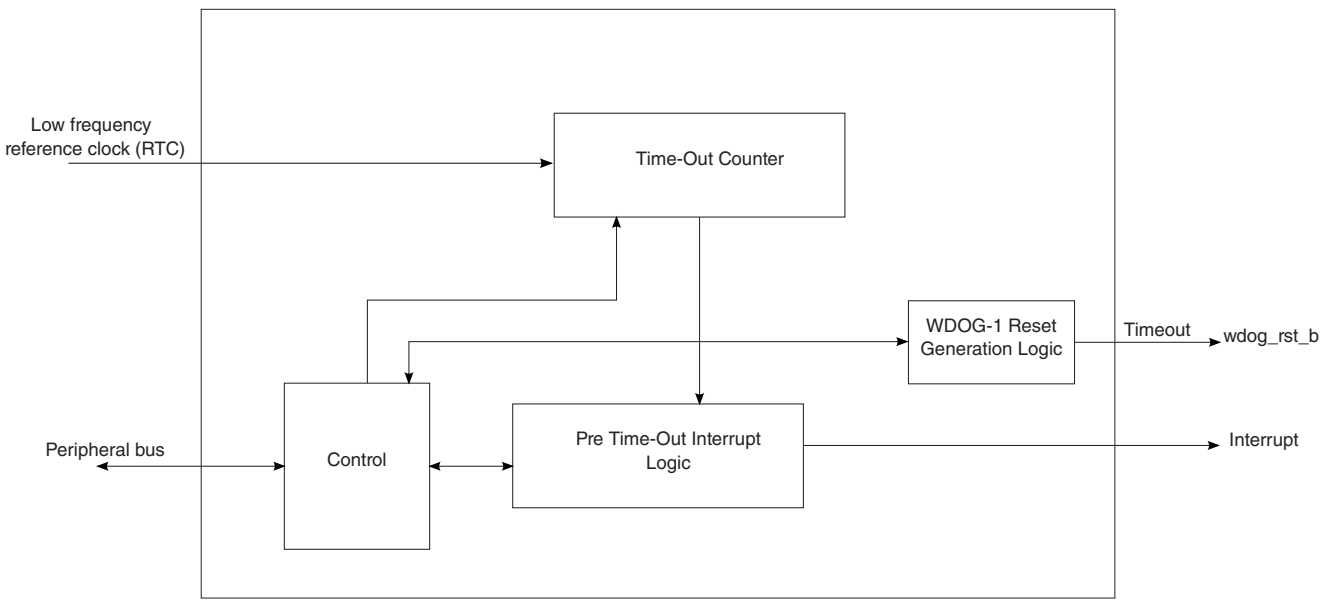


Figure 34-2. WDOG Diagram

34.1.1 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 timeout signal (wdog_reset_b).
- Time resolution of 0.5 seconds
- 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.

34.2 Clocks

This section describes clocks and special clocking requirements of the block.

The WDOG uses the low frequency reference clock (RTC) for its counter and control operations.

The low frequency reference clock is a free-running clock and can't be gated.

34.3 Functional description

This section provides a complete functional description of the block.

34.3.1 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 either a system reset signal, wdog_rst_b to COP.

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 Figure 34-4 .

34.3.1.1 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.

34.3.2 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.

34.3.3 Operations

34.3.3.1 Watchdog reset generation

The WDOG generated reset signal wdog_rst 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 wdog_rst_b 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.

34.3.4 Reset (PORESET)

The block is reset by a system reset and the WDOG counter will be disabled.

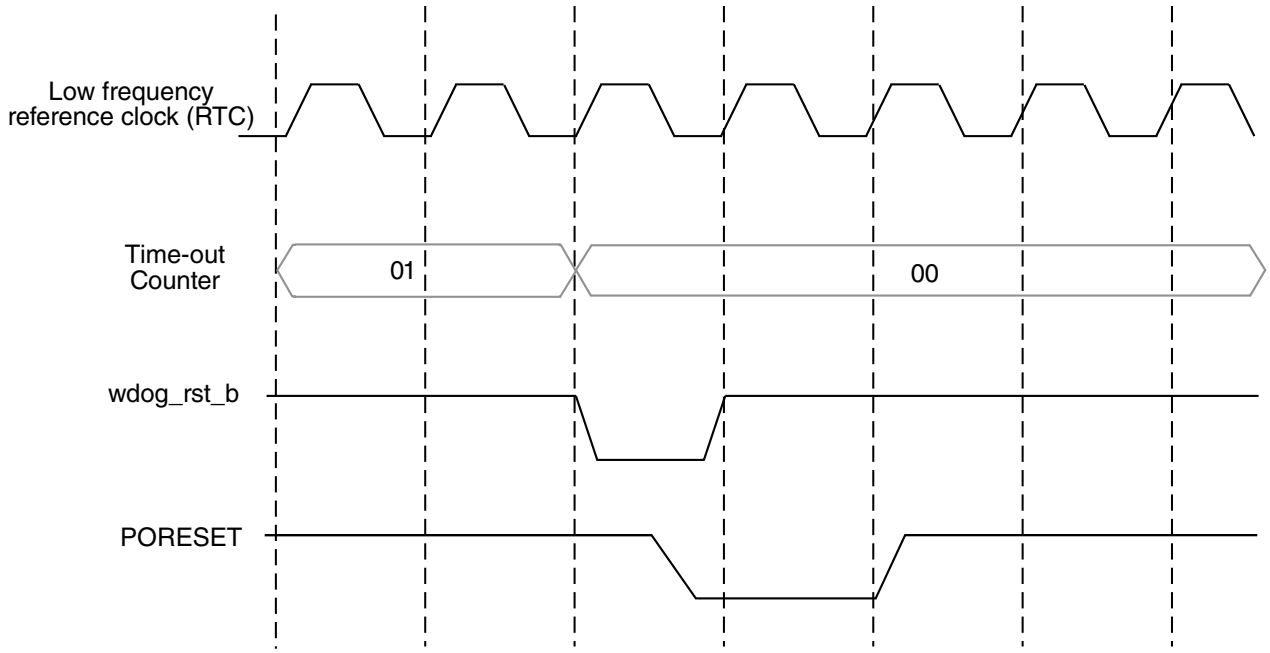


Figure 34-3. WDOG timeout/reset generation

34.3.5 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.

34.3.6 Flow Diagrams

A flow diagram of WDOG operation is shown below.

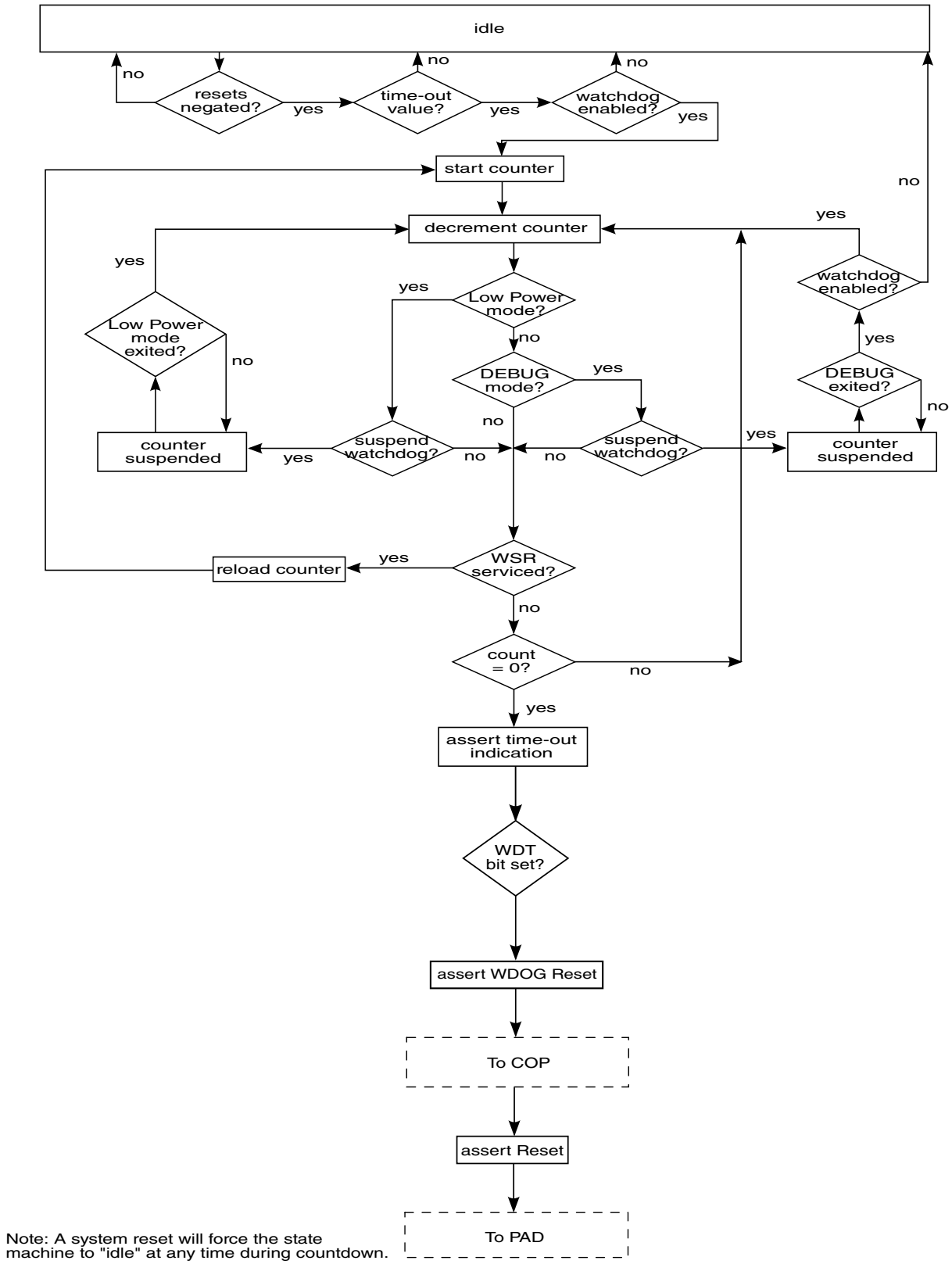


Figure 34-4. Time-Out Counter Flow Diagram

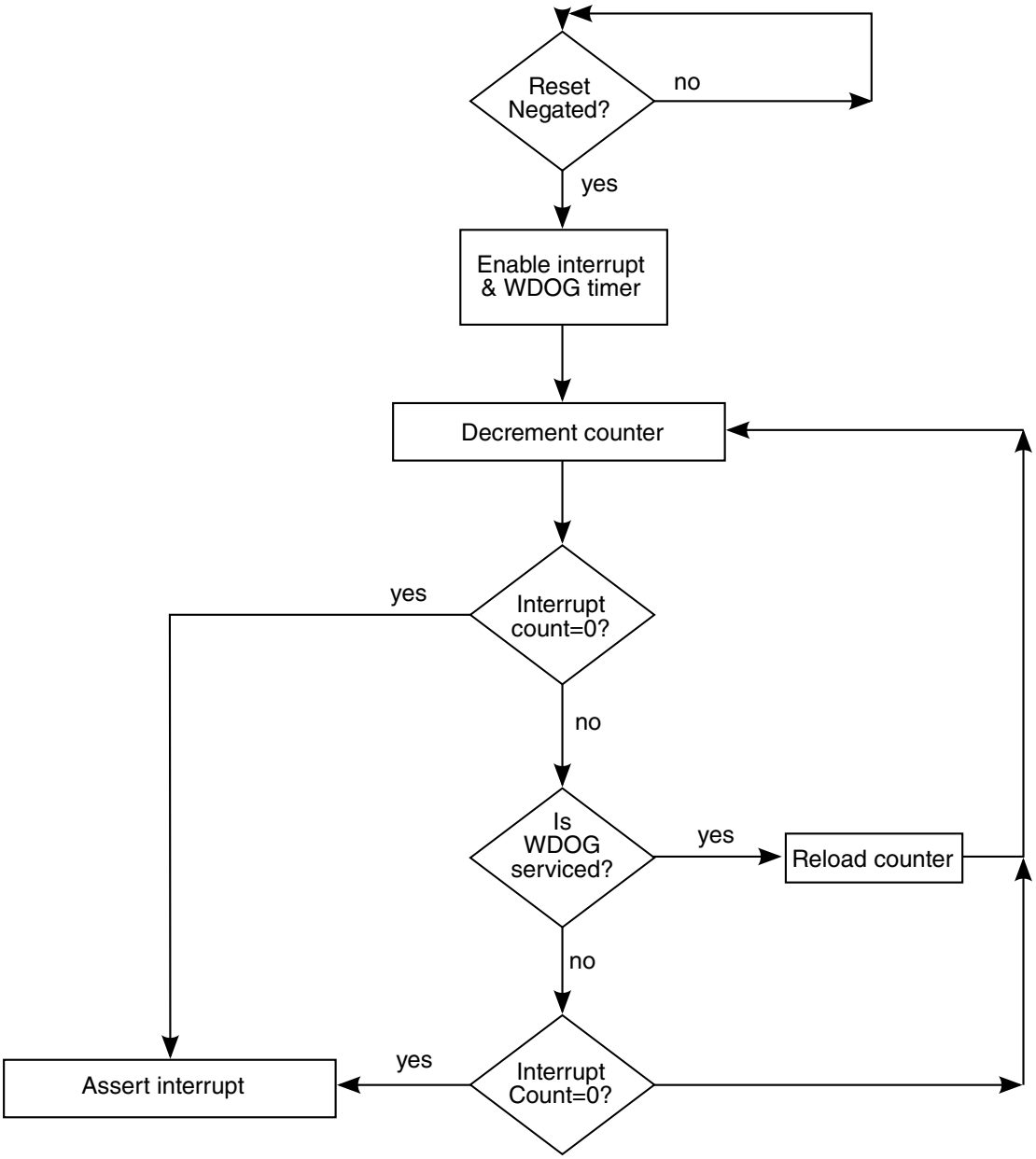


Figure 34-5. Interrupt Generation Flow Diagram

34.4 Initialization

The following sequence should be performed for WDOG initialization.

- WT 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 WT field value of Watchdog Control Register (WDOG_WCR) and starts counting.

34.5 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. A 32-Bit access should be avoided, as the system may go to an unknown state.

NOTE

The WDOG2 registers can be accessible only by the TrustZone software

WDOG memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
2AD_0000	Watchdog Control Register (WDOG1_WCR)	16	R/W	0030h	34.5.1/2340
2AD_0002	Watchdog Service Register (WDOG1_WSR)	16	R/W	0000h	34.5.2/2342
2AD_0004	Watchdog Reset Status Register (WDOG1_WRSR)	16	R	0010h	34.5.3/2342
2AD_0006	Watchdog Interrupt Control Register (WDOG1_WICR)	16	R/W	0004h	34.5.4/2343
2AE_0000	Watchdog Control Register (WDOG2_WCR)	16	R/W	0030h	34.5.1/2340
2AE_0002	Watchdog Service Register (WDOG2_WSR)	16	R/W	0000h	34.5.2/2342
2AE_0004	Watchdog Reset Status Register (WDOG2_WRSR)	16	R	0010h	34.5.3/2342
2AE_0006	Watchdog Interrupt Control Register (WDOG2_WICR)	16	R/W	0004h	34.5.4/2343

34.5.1 Watchdog Control Register (WDOGx_WCR)

The Watchdog Control Register (WDOG_WCR) controls the WDOG operation.

- WDZST is write-once only bit. Once the software does a write access to this bit, it 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.

Address: Base address + 0h offset

Bit	0	1	2	3	4	5	6	7
Read Write	WT							
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	0	Reserved		SRS	Reserved	WDE	Reserved	WDZST
Write								
Reset	0	0	1	1	0	0	0	0

WDOGx_WCR field descriptions

Field	Description
0–7 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
8 Reserved	This read-only field is reserved and always has the value 0.
9–10 -	This field is reserved.
11 SRS	Software Reset Signal. Controls the software assertion of the WDOG-generated reset signal wdog_rst_b. 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 wdog_rst_b to COP 1 No effect on the system (Default)
12 -	This field is reserved.
13 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
14 -	This field is reserved.
15 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 mode). 0 Continue timer operation (Default) 1 Suspend the watchdog timer

34.5.2 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. The write access to this register is with one wait state, provided that the write data is 0xaaaa.

NOTE

Executing the service sequence will reload the WDOG timeout counter.

Address: Base address + 2h offset

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
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
0–15 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)

34.5.3 Watchdog Reset Status Register (WDOGx_WRSR)

The WRSR is a read-only register that records the source of the output reset assertion. 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.

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	0	1	2	3	4	5	6	7
Read Write	Reserved							
Reset	0	0	0	0	0	0	0	0

Bit	8	9	10	11	12	13	14	15
Read	Reserved						TOUT	SFTW
Write								
Reset	0	0	0	1	0	0	0	0

WDOGx_WRSR field descriptions

Field	Description
0–13 -	This field is reserved.
14 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
15 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

34.5.4 Watchdog Interrupt Control Register (WDOGx_WICR)

The WDOG_WICR controls the WDOG interrupt generation.

Address: Base address + 6h offset

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
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
0 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
1 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
2–7 Reserved	This read-only field is reserved and always has the value 0.

Table continues on the next page...

WDOGx_WICR field descriptions (continued)

Field	Description
8–15 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 Time duration between interrupt and time-out is 0 seconds 0x01 Time duration between interrupt and time-out is 0.5 seconds 0x04 Time duration between interrupt and time-out is 2 seconds (Default) ... 0xff Time duration between interrupt and time-out is 127.5 seconds

Appendix A
Terminology, conventions, and resources

A.1 About this content

The primary objective of this document is to define the functionality of the chip. LS1012A provides integration of processing power for networking and communications peripherals, resulting in higher device performance. It contains a Arm® Cortex®-v8 A53 processor cores each with 32 KB of instruction and data L1 cache.

NOTE

The diagrams in this content are provided to aid in understanding the overall functionality and features of this product. They do not depict the implementation details of the product, which are subject to change.

A.2 Audience

It is assumed that the reader understands operating systems, microprocessor system design, and the basic principles of RISC processing.

A.3 Acronyms and abbreviations

This table describes commonly-used acronyms and abbreviations used in this document.

Table A-1. Acronyms and abbreviations

Acronym/ Abbreviation	Meaning
8b/10b	8-bit/10-bit encoding
AIC	Antenna interface controller
AMC	Advanced mezzanine card

Table continues on the next page...

Table A-1. Acronyms and abbreviations (continued)

Acronym/ Abbreviation	Meaning
ATM	Asynchronous transfer mode
ATMU	Address translation and mapping unit
BD	Buffer descriptor
BTB	Branch target buffer
BUID	Bus unit ID
CAM	Content-addressable memory
CCB	Core complex bus
CCSR	Configuration control and status register
CLASS	Chip-level arbitration and switching system
CRC	Cyclic redundancy check
DDR	Double data-rate
DIP	Dual inline package
DPLL	Digital phase-locked loop
DTLB	Data translation lookaside buffer
DUART	Dual universal asynchronous receiver/transmitter
ECC	Error checking and correction
EDMA	Enhanced direct memory access
EEST	Enhanced Ethernet serial transceiver
EHCI	Enhanced host controller interface
FCS	Frame-check sequence
FIFO	First in, first out
GCI	General circuit interface
GMII	Gigabit media-independent interface
GPIO	General-purpose I/O
GPR	General-purpose register
I2C	Inter-integrated circuit
IPG	Interpacket gap
IrDA	Infrared Data Association
ISDN	Integrated services digital network
ITLB	Instruction translation lookaside buffer
IU	Integer unit
HSSI	High-speed serial interface
LAE	Local access error
LAW	Local access window
LIFO	Last-in-first-out
LRU	Least recently used
LSB	Least significant byte
lsb	Least significant bit
LSU	Load/store unit

Table continues on the next page...

Table A-1. Acronyms and abbreviations (continued)

Acronym/ Abbreviation	Meaning
MAC	Multiply accumulate, media access control
MDI	Medium-dependent interface
MII	Media independent interface
MSB	Most significant byte
msb	Most significant bit
NMI	Non-maskable interrupt
NMSI	Nonmultiplexed serial interface
No-op	No operation
OCeaN	On-chip network
OC	
OSI	Open systems interconnection
PCS	Physical coding sublayer
PIC	Programmable interrupt controller
PMA	Physical medium attachment
PMD	Physical medium dependent
PLL	Phase-locked loop
POR	Power-on reset
PRI	Primary rate interface
PWM	Pulse-width modulation
RAID	Redundant array of independent drives
RGMII	Reduced gigabit media-independent interface
RTOS	Real-time operating system
RWITM	Read with intent to modify
RMW	Read-modify-write
Rx	Receiver
RxBD	Receive buffer descriptor
SATA	Serial advanced technology attachment
SCP	Serial control port
SDLC	Synchronous data link control
SD/MMC	Secure digital/multimedia card
SEC	Security Engine
SerDes	Serializer/Deserializer
SFD	Start frame delimiter
SI	Serial interface
SIU	System interface unit
SMC	Serial management controller
SPI	Serial peripheral interface
SPR	Special-purpose register
SRAM	Static random access memory

Table continues on the next page...

Table A-1. Acronyms and abbreviations (continued)

Acronym/ Abbreviation	Meaning
TAP	Test access port
TBI	Ten-bit interface
TLB	Translation lookaside buffer
TSA	Time-slot assigner
Tx	Transmitter
TxBD	Transmit buffer descriptor
UART	Universal asynchronous receiver/transmitter
UPM	User-programmable machine
uTCA	Micro telecommunications computing platform
UTP	Unshielded twisted pair
VA	Virtual address
ZBT	Zero bus turnaround

A.4 Notational conventions

This table shows notational conventions used in this content.

Table A-2. Notational conventions

Convention	Definition
General	
Cleared	When a bit takes the value zero, it is said to be cleared.
Set	When a bit takes the value one, it is said to be set.
mnemonics	Instruction mnemonics are shown in lowercase bold.
italics	Italics can indicate the following: • Variable command parameters, for example, **bcctr***x* • Titles of publications • Internal signals, for example, $\overline{core\ int}$
0x	Prefix to denote hexadecimal number
h	Suffix to denote hexadecimal number
0b	Prefix to denote binary number
b	Suffix to denote binary number
rA, **r**B	Instruction syntax used to identify a source GPR
rD	Instruction syntax used to identify a destination GPR
REGISTER[FIELD]	Abbreviations for registers are shown in uppercase text. Specific bits, fields, or ranges appear in brackets. For example, MSR[LE] refers to the little-endian mode enable bit in the machine state register.
x	In some contexts, such as signal encodings, an unitalicized x indicates a don't care.
x	An italicized *x* indicates an alphanumeric variable

Table continues on the next page...

Table A-2. Notational conventions (continued)

Convention	Definition
n	An italicized *n* indicates either: • An integer variable • A general-purpose bitfield unknown
Â¬	NOT logical operator
&	AND logical operator
|	OR logical operator
||	Concatenation, for example, TCR[WPEXT] || TCR[WP]
Signals	
$\overline{\text{OVERBAR}}$	An overbar indicates that a signal is active-low.
lowercase_italics	Lowercase italics is used to indicate internal signals
lowercase_plaintext	Lowercase plain text is used to indicate signals that are used for configuration.
Register access	
Reserved	Ignored for the purposes of determining access type
R/W	Indicates that all non-reserved fields in a register are read/write
R	Indicates that all non-reserved fields in a register are read only
W	Indicates that all non-reserved fields in a register are write only
w1c	Indicates that all non-reserved fields in a register are cleared by writing ones to them

A.5 Related resources

This table shows related resources that may be helpful. Additional literature is published as new processors become available. For current literature, visit www.nxp.com or contact the NXP FAE.

Table A-3. Related resources

Resource	Purpose
Data sheet	Provides specific data regarding bus timing, signal behavior, and AC, DC, and thermal characteristics, as well as other design considerations.
SEC reference manual	Provides specific data regarding the security engine implementation in the chip.
Application note	Addresses specific design issues useful to programmers and engineers working with NXP processors.

Appendix B
Differences between silicon revisions 1.0 and 2.0

B.1 Introduction

This appendix provides the silicon revision 1.0 specific implementation details.

B.2 Signals Description

Update the multiplexing of the CLK_OUT signal.

Table B-1. LS1012A signal reference by functional block

Name	Description	Alternate Function (s)	Pin type
System Control. Remove the following row for RESET_REQ_B.			
RESET_REQ_B	Reset Request	GPIO1_31 cfg_rcw_src	O
Clocking. Update the CLK_OUT multiplexing as follows:			
CLK_OUT	Output clock	GPIO1_31 cfg_rcw_src	O

B.3 Reset, Clocking, and Initialization

RCW field definitions

In the RCW field descriptions table, update the bits as follows:

Table B-2. RCW Field Descriptions

Bit(s) (of 0-511)	Field Name	Description	Notes/comments
26-31	CGA_PLL1_RAT	Cluster Group A PLL 1 Multiplier/Ratio	Options: 00_0110 6:1 (Asynchronous mode) 00_0111 7:1 (Asynchronous mode) 00_1000 8:1 (Asynchronous mode)
244	CGA_PLL1_SPD	Cluster Group A PLL1 Speed Select	This bit must be set based on the cluster group A PLL1 speed. Must be set to 1.
382	Reserved. Must be set to 0.		
383	CLK_OUT_BASE	This field configures the functionality of the CLK_OUT pin.	Options: 0 GPIO1[31] 1 CLK_OUT
424-425	QSPI_IIC2	This field selects the functionality of the QSPI pins for additional data bits for the 4-bit interface.	Options: 00 GPIO1[13], GPIO1[14] 01 IIC2_SCL, IIC2_SDA 10 QSPI_A_DATA2, QSPI_A_DATA3 11 GPIO1[13], RESET_REQ_B **NOTE:** The QuadSPI multiplexing can be changed by software through SCFG_PMUXCR0 register. **NOTE:** The RESET_REQ_B is muxed on secondary function. For the trust architecture, it is a requirement to implement the RESET_REQ_B. This is done by using the security fuse (ITS) as an override. If ITS = 0 and QSPI_IIC2 = 11, then RESET_REQ_B gets selected on QSPI_A_DATA3 and no selection is done on QSPI_A_DATA2. If ITS = 1, then independent of QSPI_IIC2, RESET_REQ_B must be selected on QSPI_A_DATA3 and GPIO1[13] gets selected on QSPI_A_DATA2.

RCW settings for hard-coded options

Update the QSPI_IIC2 bit as follows:

Table B-3. RCW settings for hard-coded RCW options

RCW field	cfg_rcw_src
QSPI_IIC2	2'b00 (GPIO) **NOTE:** If ITS=1, the configuration corresponding to ITS=1 must have the precedence over the default value present in the RCW. For details, see the QSPI_IIC2 bit description in RCW Field Definitions.

System control signals

Update the RESET_REQ_B signal description to add the following:

Table B-4. System control signals: Detailed signal descriptions

Signal	I/O	Description
RESET_REQ_B	O	Reset request (optional, if ITS=0, refer Secure Boot and Trust Architecture chapter for details on ITS). It is the primary functionality only for secure device, else this needs to be selected through RCW.

Reset_REQ_B behavior

Remove the following paragraph:

The RESET_REQ_B is multiplexed on secondary function. For trust architecture, it is a requirement to implement the RESET_REQ_B. The RESET_REQ_B can be selected through RCW:

- on CLK_OUT pin
- on the multiplexed RESET_REQ_B/QSPI/I2C/GPIO pin

SGMII use cases

Update the clocking scheme as follows for silicon revision 1.0.

For the systems with SGMII PHY, an external clock oscillator/generator is required, as shown in the figure below:

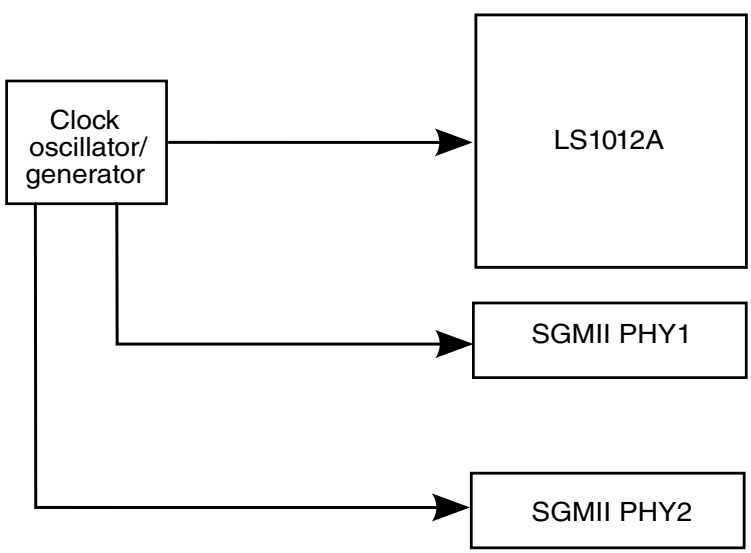


Figure B-1. MAC to SGMII PHY clocking

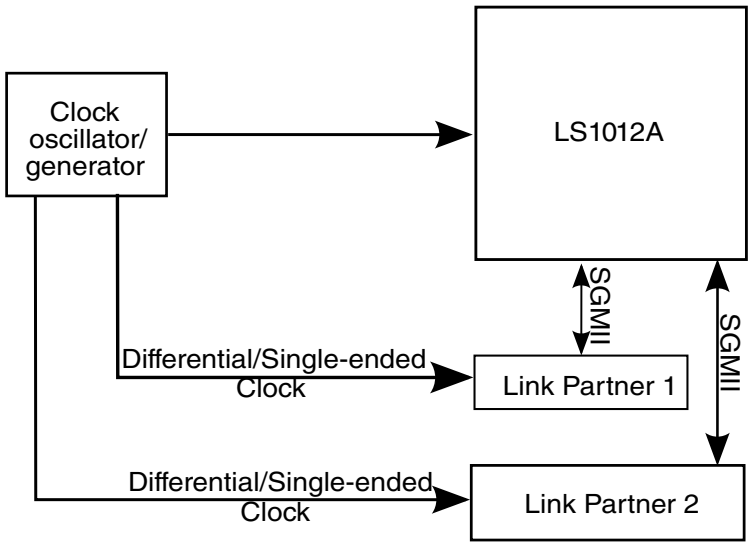


Figure B-2. MAC to MAC clocking

NOTE

The clock source should be common for both LS1012A SerDes and SGMII PHY/link partner.

B.4 Supplement Configuration Unit

Update the PMUXCR0[QSPI_IIC2_OVRD] bit description as follows:

Field	Function
QSPI_IIC2_OVRD	Software configures alternate functionality of the QuadSPI pins for additional 2-bit interface **NOTE:** The QuadSPI multiplexing can be changed by software through SCFG registers. This configuration is valid only if ITS=0. If ITS=1, software cannot change the mux control of QSPI_A_DATA2 and QSPI_A_DATA3 and these will stay as GPIO_1[13] and RESET_REQ_B, respectively. 00b - GPIO_1[13], GPIO_1[14] (default) 01b - IIC2_SCL, IIC2_SDA 10b - QSPI_A_DATA2, QSPI_A_DATA3 11b - Reserved, RESET_REQ_B

B.5 Device Configuration and Pin Control

Update the DCFG_CCSR[SVR] bit description as follows:

Field	Function
MAJOR_REV	Major Revision Number For silicon 1.0, the major revision is 0x1.

Appendix C
Revision History

C.1 Substantive changes from revision 0 to revision 1

Substantive changes from revision 0 to revision 1 are as follows:

C.1.1 Overview Revision History

Reference	Description
Introduction Features summary	Updated the core speed from 800 MHz to 1000 MHz.
Features summary	Added "one eDMA controller" in the additional peripheral list.

C.1.2 Memory Map Revision History

Reference	Description
System memory map, DDR remapping	Updated the 30 GB space at address 8_8000_0000 to reserved. Removed the DDR remapping section.
CCSR address map	Updated the CCSR configuration bus endianness of Security Monitor from big-endian to little-endian.

C.1.3 Signal Descriptions Revision History

Reference	Description
I^2C1, GPIO1, and FTM signal multiplexing	Updated the IIC1_SCL and IIC1_SDA bit settings.

Table continues on the next page...

Substantive changes from revision 0 to revision 1

Reference	Description
TA_TMP_DETECT_B and GPIO2 signal multiplexing	Added settings for TA_TMP_DETECT_B and GPIO signals configuration.
Signals Overview	Added RESET_REQ_B multiplexed on CLK_OUT pin. Add the following USB signals: USB1_RESREF, USB1_RX_M, USB1_TX_M, and USB1_VBUS.
Output Signal States During Reset	Removed EMI1_MDC from output-only signals list in the second paragraph.

C.1.4 Reset, Clocking, and Initialization Revision History

Reference	Description
RCW Field Definitions	Updated the note in RCW[SerDes_INT_REFCLK] description.
Figure 4-12	Added 1/4 divider for system counter clock.
RESET_REQ_B behavior	Updated this section for RESET_REQ_B recommendation and restrictions.
System control signals	Updated the RESET_REQ_B description.
Table 4-10	Updated the CLK_OUT_BASE bit to 2-bit (382-383).
Table 4-10	Updated the QSPI_IIC2 bit description.
RCW settings for hard-coded options	Removed the footnote from the QSPI_IIC2 bit.
RCW Field Definitions	Updated the CGA_PLL1_RAT bit description to add the settings for 0b001001 and 0b001010. Updated the CGA_PLL1_SPD bit description settings for high-speed operation (>= 1GHz).
SGMII use cases	Updated the SGMII PHY clocking scheme.

C.1.5 CSU, OCRAM, and MSCM Revision History

Reference	Description
Central Security Unit	Added the chapter.

C.1.6 SCFG Revision History

Reference	Description
Pinmux control register (PMUXCR0)	Updated the QSPI_DATA1_GPIO_OVRD and QSPI_IIC2_OVERD bit settings.

Table continues on the next page...

Reference	Description
USB3 parameter 3 control register (USB3PRM3CR)	Updated the reset value of USB3PRM3CR register to 0x00190000.

C.1.7 Device Configuration and Pin Control Revision History

Reference	Description
DCFG_CCSR register descriptions	Removed notes from DEVDISR2 and DEVDISR3 registers.
Core Reset Status Register n (CRSTSR0)	Added the CRSTSR register at offset 0x400.
Fuse Status Register (FUSESR)	Updated the note as follows: "VID is not applicable for this chip, which means the boot software is not required to read this register and adjust voltage.".
System Version Register (SVR)	Updated the major revision to 0010.
Reset Request Mask Register (RSTRQMR1) and Reset Request Status Register (RSTRQSR1)	Removed the following statement from the note: "This register is not applicable when RESET_REQ_B is not selected".

C.1.8 DMAMUX Revision History

Reference	Description
Throughout	Removed the TRIGGER functionality as it is not supported on any eDMA channel. Updated the complete chapter to remove internal information.

C.1.9 eDMA Revision History

Reference	Description
TCD Transfer Attributes (TCD0_ATTR - TCD31_ATTR)	Added the note to TCDn_ATTR[SSIZE] description about privileged data access.
Channel Priority Register (DCHPRI0 - DCHPRI31)	Removed note about reset value from DCHPRIn[CHPRI] field description.
Throughout	Updated the complete chapter to remove internal information.
TCD Control and Status (TCD0_CSR - TCD31_CSR)	Changed access of TCDn_CSR[DONE] from RW to W0C and access of TCD_CSR[ACTIVE] from RW to RO.
Suspend/resume a DMA channel with active hardware service requests	Added this section.
Enable Request Register (ERQ) and Clear Enable Request Register (CERQ)	Added note, beginning with "Disable a channel's hardware servcie request..." to these register descriptions.

C.1.10 I2S/SAI Revision History

Reference	Description
SAI Receive Configuration 1 Register (I2S_RCR1)	Updated TCR3[TCE] and TCR3[CFR] bit descriptions.

C.1.11 MMDC Revision History

Reference	Description
Address decoding	Removed the "Chip select settings" section. Updated the "Address decoding" section to remove references of MDASP register.

C.1.12 PCI Express Revision History

Reference	Description
DBI Read-only Write Enable Register (MISC_CONTROL_1_OFF)	Changed reset value for MISC_CONTROL_1_OFF register to 0000_0000h.
iATU Limit Address Register (IATU_LIMIT_ADDR_OFF_INBOUND_0) and iATU Limit Address Register (IATU_LIMIT_ADDR_OFF_OUTBOUND_0)	Changed the IATU_LIMIT_ADDR_OFF_INBOUND_0[LWR_BASE_HW] and IATU_LIMIT_ADDR_OFF_OUTBOUND_0[LWR_BASE_HW] bit descriptions.
Symbol Timer Register and Filter Mask 1 Register (SYMBOL_TIMER_FILTER_1_OFF)	Changed reset value for the SYMBOL_TIMER_FILTER_1_OFF register.
PCI Express Interrupt Pin Register (Interrupt_Pin_Register)	Removed values for INTB, INTC, and INTD from Interrupt_Pin_Register[Interrupt_pin] bit description.
PCI Express Uncorrectable Error Status Register (Uncorrectable_Error_Status_Register)	Removed the TE bit (bit 0).
PCI Express Uncorrectable Error Mask Register (Uncorrectable_Error_Mask_Register)	Removed the TEM bit (bit 0).
Lane reversal	Removed this section.
Throughout	Updated the number of inbound and outbound iATU address regions to 2.

C.1.13 QuadSPI Revision History

Reference	Description
Peripheral Bus Register Descriptions	• In QSPI_FR[TBUF] description, added a sentence "Here valid 'underrun' means.....are written in TX buffer". • Added a sentence, "This bit should be programmed two times to clear the sequence pointers" in QuadSPI_SPTRCLR[IPPTRC] and QuadSPI_SPTRCLR[BFPTRC] description respectively. • In QSPI_FR[ILLINE] description, added a sentence, "As soon as this flag is set, user should....correct sequence instruction". • In QSPI_FLSHCR[TCSH] description, added a sentence, "Default value '3' should be used in case AHB prefetch size is less than or equal to 32-bytes DDR/16-bytes SDR".
Features	Updated the first, fourth, and seventh bullets.
Modes of Operation	Moved the QuadSPI mode descriptions from functional description to this section.
Throughout	Removed the flash B-specific information. Removed support for parallel flash mode. Removed support for dual-die flashes.
Throughout	Added the following note: Access to location 0X4 and 0x190 does not give transfer error.
Module Configuration Register (QuadSPI_MCR)	Updated the register description to add "SCLKCFG: Disabled Mode".
Buffer0 Top Index Register (QuadSPI_BUF0IND)	Updated the second and third paragrapha in the register description.
Sampling Register (QuadSPI_SMPR)	Added notes in the FSDLY and FSPHS bit descriptions.
Look-up Table register (QuadSPI_LUT*n*)	Updated the register description.
Look-up Table register (QuadSPI_LUT*n*)	Updated the register description.
Flash memory mapped AMBA bus	Updated the section title and first paragraph. Removed the second paragraph of the note.
AHB Bus Access Considerations	Updated the first bullet to add "At present, the QuadSPI does not support AHB writes so any.....".
Programmable Sequence Engine	Removed the paragraph (starting with "A sequence of such instruction operand.....") next to the Instruction set table.
Flash Programming	Updated the first bullet.

C.1.14 SerDes Module Revision History

Reference	Description
Throughout	Added 1000Base-KX support.
Memory map	Added MDIO registers.
SerDes protocols	Updated the first note. Removed the last note mentioning the extra requirements for PCI Express.
SerDes Memory map	In PCCR8[SGMIIA_CFG] - changed "Lane 0 on WRIOP Mac 9" to "Lane A on PFE MAC 1" In PCCR8[SGMIIB_CFG] - changed "Lane 1 on WRIOP Mac 5" to "Lane B on PFE MAC 2"
Valid reference clocks and PLL configurations for SerDes protocols	updated the first note for spread-spectrum reference clock.
SerDes Memory map	• In PLLn_CR1, changed VCO_EN field (bit 21) to "Reserved." VCO_EN does not apply to this SerDes. • In PLLn_CR5[SEL_REFCLK_AMP_DIS], revised the field description to state, "Default value set by RCW." • In TCALCR1, revised the field descriptions for ANA_OUT_REFCLK_EN and DIG_OUT_REFCLK_EN. • In RCALCR1, revised the field descriptions for ANA_OUT_REFCLK_EN and DIG_OUT_REFCLK_EN.

C.1.15 Secure Boot and Trust Architecture Revision History

Reference	Description
	Removed the section Timer Facilities. Moved the sections CSU, Platform Control, MSCM, ACTZS (Access Control) to individual chapter.
Central Security Unit	Removed last sentence.
LS1012A SecMon module special consideration	Added this section.
Security monitor	Removed information specific to features not available in the chip, such as LP-mode and ZMK. Removed registers from following offsets: 8h, 24h, 28h, 2Ch, 30h, 50h, 54h, 58h, 5Ch, and 60h. Updated SecMon registers to reflect that they are accessed as little-endian.
Config Security Level (CSU_CSL)	Updated CSU registers to reflect that they are accessed as big-endian.

C.1.16 SPI Revision History

Reference	Description
Status Register (SR)	Updated SR[TFFF] description to "The TFFF bit can be cleared by writing 1 to it or by acknowledgement from the

Table continues on the next page...

Reference	Description
	DMA controller to the TX FIFO full request, when the TX FIFO is full."
Status Register Extended (SREX)	Swapped SREX[RXCTR4] and SREX[TXCTR4].
Throughout	Reformatted the complete chapter.
	Updated the note in the memory map description.
Status Register (SR)	Added the following note in the SPI_SR[TXNXTPTR] bit description: When TX FIFO Is cleared by setting SPI_MCR[CLR_TXF] to 1, this field does not update immediately to 0. Only when the next transfer starts, this field reflects the latest value TXNXTPTR =1.

C.1.17 Thermal Management Unit Revision History

Reference	Description
TMU register descriptions	Removed IPBRR0 register.
Table 31-2	Updated the calibration values.

C.1.18 Universal Serial Bus 2.0 Interface Revision History

Reference	Description
PHY clocks	Updated the first sentence: "The USBn_CLK input is the clock from the external ULPI PHY to the controller." Updated the note: "A write to registers in the USB controller memory map may cause the system to hang if PORTSC[PHCD]=0 and there is no clock from the ULPI PHY."

C.1.19 Universal Serial Bus 3.0 Interface Revision History

Reference	Description
USB PHY SuperSpeed register descriptions	Removed registers and bitfields except SUP IDCODE HI, SUP IDCODE LO, MPLL_LOOP_CTL bits 4-7 and LANE0RX_OVERD_IN_HI bits 6-11.
SUP_IDCODE_HI (SUP_IDCODE_HI)	Updated the reset value from 921Ch to A21Ch.

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Document Number LS1012ARM
Revision 1, 01/2018