Release notes

The current content maturity status of each chapter is noted in the table below. Please refer the Revison History Appendix for more details.

Table 0-1. Content maturity matrix

Ch #	Chapter Name	Chapter Maturity			Comments
		Preliminary	Developing	Mature/Stable	
1	Overview			x	
2	Memory Map			x	
3	Signal Descriptions			x	
4	Reset, Clocking, and Initialization			x	
5	Pre-Boot Loader (PBL)			x	Added bullet stating that the Flush command is restricted to CCSR space.
6	Secure Boot and Trust Architecture			x	
7	e6500 Core Integration			x	
8	CoreNet Platform Cache (CPC)			x	Removed LOADEC. Modified bit descriptions of CPC partition allocation register 0 (CPC_CPCPAR0), CPC partition allocation register n (CPC_CPCPARn).
9	Prefetch Manager (PMAN)			x	See Appendix B.
10	CoreNet Coherency Fabric (CCF)			x	See Appendix B.
11	Peripheral Access Management Unit (PAMU)			x	
12	DDR Memory Controller			x	See Appendix B.
13	Controller (IFC)			x	See Appendix B.
14	I2C Modules			x	
15	Enhanced Serial Peripheral Interface			x	
16	Enhanced Secured Digital Host Controller (eSDHC)			x	Defeatured DAT3 as a CD pin.
17	Universal Serial Bus Interfaces			x	See Appendix B.
18	DUART			x	See Appendix B.
19	SerDes Module			x	Added options 67-1F, AB-02, DB-1F, DB-2D, 66-16.

Table 0-1. Content maturity matrix (continued)

Ch #	Chapter Name	Chapter Maturity			Comments
		Preliminary	Developing	Mature/Stable	
20	PCI Express Interface Controller			x	Added TRGT mapping in registers.
21	Serial RapidIO Interface			x	See Appendix B.
22	SATA Controller			x	
23	DMA Controller			x	Defeatured ATMU bypass mode.
24	Data Path Acceleration Architecture (DPAA) Overview and DPAA Implementation			x	
25	Multicore Programmable Interrupt Controller (MPIC)			x	See Appendix B.
26	Interrupt Assignments			x	
27	Device Configuration and Pin Control			x	See Appendix B.
28	General Purpose I/O (GPIO)			x	
29	Thermal Monitoring Unit (TMU)			x	
30	Run Control and Power Management (RCPM)			x	See Appendix B.
	T2081 Appendix			x	Added options 6C and 70.

T2080 Integrated Multicore Communications Processor Family Reference Manual

Also supports T2081

Document Number: T2080RM
Rev C.1, 9/2013

freescale™

Chapter 1
T2080 Overview

1.1 Introduction

The T2080 QorIQ multicore processor combines four, dual-threaded e6500 Power Architecture® processor cores with high-performance datapath acceleration logic and network and peripheral bus interfaces required for networking, telecom/datacom, wireless infrastructure, and mil/aerospace applications.

1.2 Summary of benefits

This chip can be used for combined control, datapath, and application layer processing in routers, switches, gateways, application and storage servers, and general-purpose embedded computing systems. Like other QorIQ SoCs, this chip's high level of integration offers significant space, weight, and power benefits compared to multiple discrete devices.

1.3 Chip features

This section describes the key features and functionalities of the chip.

1.3.1 Block diagram

This figure shows the major functional units within the chip.

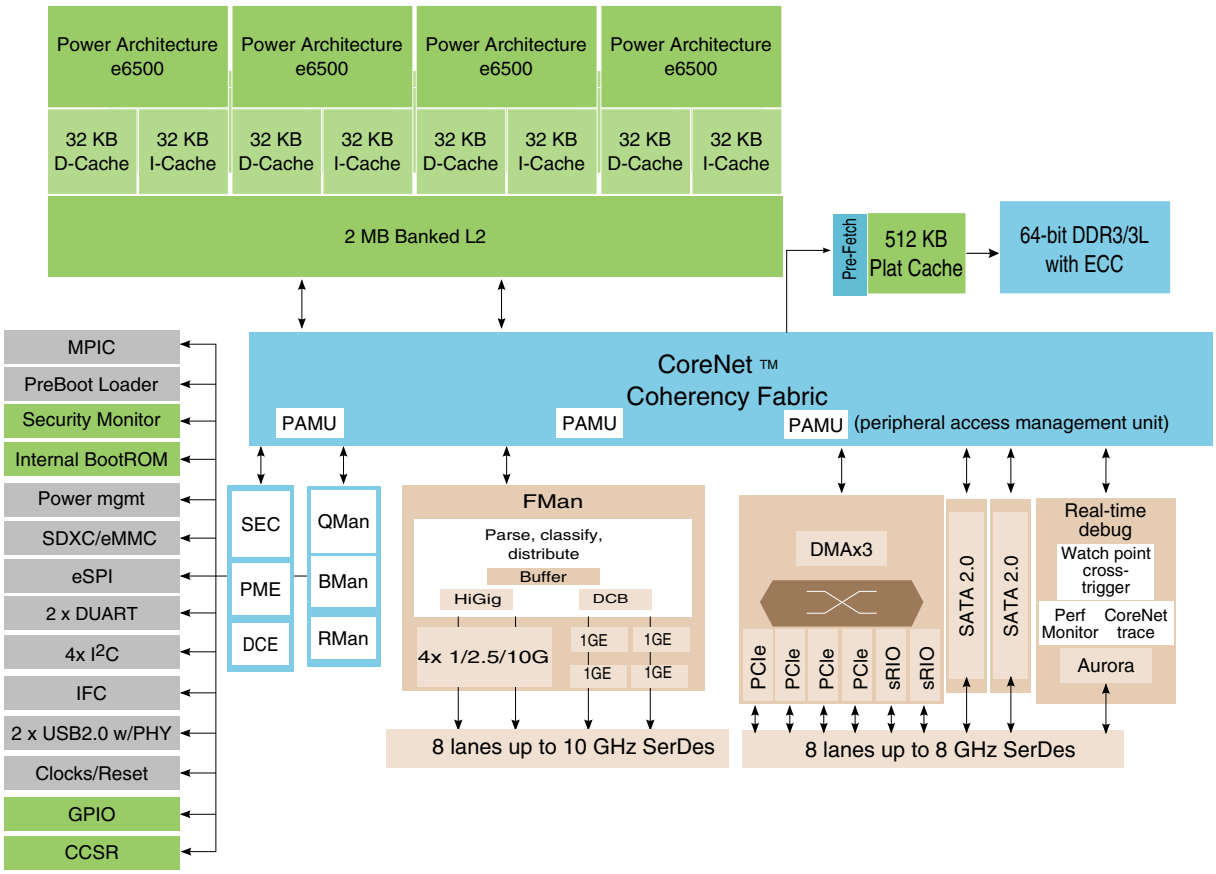


Figure 1-1. T2080 block diagram

1.3.2 Features summary

This chip includes the following functions and features:

- 4, dual-threaded e6500 cores built on Power Architecture® technology sharing a 2 MB L2 cache
  - Up to 1.8 GHz with 64-bit ISA support (Power Architecture v2.06-compliant)
- 512 KB CoreNet platform cache (CPC)
- Hierarchical interconnect fabric
  - CoreNet fabric supporting coherent and non-coherent transactions with prioritization and bandwidth allocation amongst CoreNet end-points
  - Queue Manager (QMan) fabric supporting packet-level queue management and quality of service scheduling
- One 32-/64-bit DDR3/3L SDRAM memory controllers with ECC and interleaving support
  - Memory pre-fetch engine

- Data Path Acceleration Architecture (DPAA) incorporating acceleration for the following functions:
  - Packet parsing, classification, and distribution (Frame Manager)
  - Queue management for scheduling, packet sequencing, and congestion management (Queue Manager)
  - Hardware buffer management for buffer allocation and de-allocation (BMan)
  - Cryptography acceleration (SEC 5.2) at up to 10 Gbps
  - RegEx Pattern Matching Acceleration (PME 2.1) at up to 10 Gbps
  - Decompression/Compression Acceleration (DCE) at up to 17.5 Gbps
  - DPAA chip-to-chip interconnect via RapidIO Message Manager (RMAN)
- 16 SerDes lanes at up to 10.3125 GHz
- Eight Ethernet interfaces, supporting combinations of the following:
  - Up to four 10 Gbps Ethernet MACs
  - Up to eight 1 Gbps Ethernet MACs
  - Up to four 2.5 Gbps Ethernet MACs
- High-speed peripheral interfaces
  - Four PCI Express controllers (two support PCIe 2.0 and two support PCIe 3.0)
  - Two Serial RapidIO 2.0 controllers/ports running at up to 5 GHz with Type 11 messaging and Type 9 data streaming support
- Additional peripheral interfaces
  - Two serial ATA (SATA 2.0) controllers
  - Two high-speed USB 2.0 controllers with integrated PHY
  - Enhanced secure digital host controller (SD/SDXC/eMMC)
  - Enhanced serial peripheral interface (eSPI)
  - Four I2C controllers
  - Four 2-pin UARTs or two 4-pin UARTs
  - Integrated Flash controller supporting NAND and NOR flash
- Three eight-channel DMA engines
- Support for hardware virtualization and partitioning enforcement
- QorIQ Platform's Trust Architecture 2.0

1.3.3 Core and CPU clusters

This chip offers four, high-performance 64-bit Power Architecture Book E-compliant cores. Each CPU core supports two hardware threads, which software views as a virtual CPU. The core CPUs are arranged in a cluster of four with a shared 2 MB L2 cache.

The core subsystem includes the following features:

- Up to 1.8 GHz
- Dual-thread with simultaneous multi-threading (SMT)

    - Threading can be disabled on a per CPU basis
- 40-bit physical addressing
- L2 MMU
    - Supporting 4 KB pages
    - TLB0; 8-way set-associative, 1024-entries (4 KB pages)
    - TLB1; fully associative, 64-entry, supporting variable size pages and indirect page table entries
- Hardware page table walk
- 64-byte cache line size
- L1 caches, running at core frequency
    - 32 KB instruction, 8-way set-associative
    - 32 KB data, 8-way set-associative
    - Each with data and tag parity protection
- Hardware support for memory coherency
- Five integer units: 4 simple (2 per thread), 1 complex (integer multiply and divide)
- Two load-store units: one per thread
- Classic double-precision floating-point unit
    - Uses 32 64-bit floating-point registers (FPRs) for scalar single- and double-precision floating-point arithmetic
    - Designed to comply with IEEE Std. 754™-1985 FPU for both single and double-precision operations
- AltiVec unit
    - 128-bit Vector SIMD engine
    - 32 128-bit VR registers
    - Operates on a vector of
        - Four 32-bit integers
        - Four 32-bit single precision floating-point units
        - Eight 16-bit integers
        - Sixteen 8-bit integers
    - Powerful permute unit
    - Enhancements include: Move from GPRs to VR, sum of absolute differences operation, extended support for misaligned vectors, handling head and tails of vectors
- Supports Data Path Acceleration Architecture (DPAA) data and context "stashing" into L1 and L2 caches
- User, supervisor, and hypervisor instruction level privileges
- Addition of Elemental Barriers and "wait on reservation" instructions
- New power-saving modes including "drowsy core" with state retention and nap
    - State retention power-saving mode allows core to quickly wake up and respond to service requests
- Processor facilities

- Hypervisor APU
- "Decorated Storage" APU for improved statistics support
  - Provides additional atomic operations, including a "fire-and-forget" atomic update of up to two 64-bit quantities by a single access
- Addition of Logical to Real Address translation mechanism (LRAT) to accelerate hypervisor performance
- Expanded interrupt model
  - Improved Programmable Interrupt Controller (PIC) automatically ACKs interrupts
  - Implements message send and receive functions for interprocessor communication, including receive filtering
- External PID load and store facility
  - Provides system software with an efficient means to move data and perform cache operations between two disjoint address spaces
  - Eliminates the need to copy data from a source context into a kernel context, change to destination address space, then copy the data to the destination address space or alternatively to map the user space into the kernel address space

The arrangement of cores into clusters with shared L2 caches is part of a major re-architecture of the QorIQ cache hierarchy. Details of the banked L2 are provided below.

- 2 MB cache with ECC protection (data, tag, & status)
  - Operates at same frequency as the CPUs in the cluster
- 64-byte cache line size
- 16 way, set associative
  - 4 banks, supporting up to four concurrent accesses.
  - Each bank can be configured as exclusive to a vCPU or shared
    - A vCPU can exclusively own one or more banks
  - Ways in each bank can be configured in one of several modes
  - Flexible way partitioning per vCPU
    - I-only, D-only, or unified
- Supports direct stashing of datapath architecture data into L2

1.3.4 Inverted cache hierarchy

From the perspective of software running on an core vCPU, the SoC incorporates a 2-level cache hierarchy. These levels are as follows:

- Level 1: Individual core 32 KB Instruction and Data caches
- Level 2: Locally banked 2 MB cache (configurably shared by other vCPUs in the cluster)

Therefore, the CPC is not intended to act as backing store for the L2s. This allows the CPCs to be dedicated to the non-CPU masters in the SoC, storing DPAA data structures and IO data that the CPUs and accelerators will most likely need.

Although the SoC supports allocation policies that would result in CPU instructions and in data being held in the CPC (CPC acting as vCPU L3), this is not the default. Because the CPC serves fewer masters, it serves those masters better, by reducing the DDR bandwidth consumed by the DPAA and improving the average latency.

1.3.5 CoreNet fabric and address map

The CoreNet fabric provides the following:

- A highly concurrent, fully cache coherent, multi-ported fabric
- Point-to-point connectivity with flexible protocol architecture allows for pipelined interconnection between CPUs, platform caches, memory controllers, and I/O and accelerators at up to 700 MHz
- Eliminates address retries, triggered by CPUs being unable to snoop within the narrow snooping window of a shared bus. This results in the chip having lower average memory latency.

This flexible chip's 40-bit, physical address map consists of local space and external address space. Inbound and outbound translation windows can map the chip into a larger system address space such as the RapidIO or PCIe 64-bit address environment. This functionality is included in the address translation and mapping units (ATMUs).

1.3.6 DDR memory controller

The chip offers a single DDR controller supporting ECC protected memories. This DDR controller operates at up to 2133 MHz for DDR3, and, in more power-sensitive applications, up to 1866.667 MHz for DDR3L. Some key DDR controller features are as follows:

- Support x8 and x16 memory widths
  - Programmable support for single-, dual-, and quad-ranked devices and modules
  - Support for both unbuffered and registered DIMMs
  - 4 chip-selects
  - 40-bit address support, up to 1 TB memory

- The SoC can be configured to retain the currently active SDRAM page for pipelined burst accesses. Page mode support of up to 64 simultaneously open pages can dramatically reduce access latencies for page hits. Depending on the memory system design and timing parameters, page mode can save up to ten memory clock cycles for subsequent burst accesses that hit in an active page.
- Using ECC, the SoC detects and corrects all single-bit errors and detects all double-bit errors and all errors within a nibble.
- Upon detection of a loss of power signal from external logic, the DDR controller can put compliant DDR SDRAM DIMMs into self-refresh mode, allowing systems to implement battery-backed main memory protection.
- In addition, the DDR controller offers an initialization bypass feature for use by system designers to prevent re-initialization of main memory during system power-on after an abnormal shutdown.
- Support active zeroization of system memory upon detection of a user-defined security violation.

1.3.6.1 DDR bandwidth optimizations

Multicore SoCs are able to increase CPU and network interface bandwidths faster than commodity DRAM technologies are improving. As a result, it becomes increasingly important to maximize utilization of main memory interfaces to avoid a memory bottleneck. The SoC's DDR controller is Freescale-developed IP, optimized for the QorIQ SoC architecture, with the goal of improving DDR bandwidth utilization by fifty percent when compared to first generation QorIQ SoCs.

Most of the WRITE bandwidth improvement and approximately half of the READ bandwidth improvement is met through target queue enhancements; in specific, changes to the scheduling algorithm, improvements in the bank hashing scheme, support for more transaction re-ordering, and additional proprietary techniques.

The remainder of the READ bandwidth improvement is due to the addition of an intelligent data prefetcher in the memory subsystem.

1.3.7 Universal serial bus (USB) 2.0

The two USB 2.0 controllers with integrated PHY provide point-to-point connectivity that complies with the USB specification, Rev. 2.0. Each of the USB controllers with integrated PHY can be configured to operate as a stand-alone host, and one of the controllers (USB #2) can be configured as a stand-alone device, or with both host and device functions operating simultaneously.

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
- Both controllers support operation as a stand-alone USB host controller
  - Supports USB root hub with one downstream-facing port
  - Enhanced host controller interface (EHCI)-compatible
- Both controllers supports operation as a stand-alone USB device
  - Support one upstream-facing port
  - Support six programmable USB endpoints

The host and device functions are both configured to support all four USB transfer types:

- Bulk
- Control
- Interrupt
- Isochronous

1.3.8 High-speed peripheral interface complex (HSSI)

The SoC offers a variety of high-speed serial interfaces, sharing a set of 16 SerDes lanes. Each interface is backed by a high speed serial interface controller. The SoC has the following types and quantities of controllers:

- Four PCI Express controllers, one Gen 3.0 PCI Express controller with SRIOV, one Gen 3.0 PCI Express controller without SRIOV and two PCI Express Gen 3.0 controllers
- Two Serial RapidIO 2.0
- Two SATA 2.0
- Up to eight Ethernet controllers with various protocols
- Aurora

The features of each high-speed serial controller are described in the subsequent sections. Debug functionality is described in Debug support."

1.3.8.1 PCI Express

This chip instantiates four PCI Express controllers, each with the following key features:

- One PCI Express controller supports end-point SR-IOV.
  - Two physical functions

  - 64 virtual functions per physical function
  - Eight MSI-X per either physical function or virtual function
- Two PCI Express controllers support 2.0 (maximum lane width off x8).
- Two PCI Express controllers support 3.0 (maximum lane width of x4).
- Power-on reset configuration options allow root complex or endpoint functionality.
- x8, x4, x2, and x1 link widths support
- Both 32- and 64-bit addressing and 256-byte maximum payload size
- Inbound INTx transactions
- Message signaled interrupt (MSI) transactions

1.3.8.2 Serial RapidIO

The Serial RapidIO interface is based on the *RapidIO Interconnect Specification, Revision 2.1* . RapidIO is a high-performance, point-to-point, low-pin-count, packet-switched system-level interconnect that can be used in a variety of applications as an open standard. The rich feature set includes high data bandwidth, low-latency capability, and support for high-performance I/O devices as well as message-passing and software-managed programming models. Receive and transmit ports operate independently, and with 2 x 4 Serial RapidIO controllers, the aggregate theoretical bandwidth is 32 Gbps.

The chip offers two Serial RapidIO controllers Receive and transmit ports operate independently and with 2 x 4 Serial RapidIO controllers; the aggregate theoretical bandwidth is 32 Gbps. The Serial RapidIO controllers can be used in conjunction with "Rapid IO Message Manager (RMAN), as described in RapidIO Message Manager (RMan 1.0)."

Key features of the Serial RapidIO interface unit include the following:

- Support for *RapidIO Interconnect Specification, Revision 2.1* (All transaction flows and priorities.)
- 2x, and 4x LP-serial link interfaces, with transmission rates of 2.5, 3.125, or 5.0 Gbaud (data rates of 1.0, 2.0, 2.5, or 4.0 Gbps) per lane
- Auto-detection of 1x, 2x, or 4x mode operation during port initialization
- 34-bit addressing and up to 256-byte data payload
- Support for SWRITE, NWRITE, NWRITE_R and Atomic transactions
- Receiver-controlled flow control
- RapidIO error injection
- Internal LP-serial and application interface-level loopback modes

The Serial RapidIO controller also supports the following capabilities, many of which are leveraged by the RMan to efficient chip-to-chip communication through the DPAA:

- Support for RapidIO Interconnect Specification 2.1, "Part 2: Message Passing Logical Specification"
- Supports RapidIO Interconnect Specification 2.1, "Part 10: Data Streaming Logical Specification"
- Supports RapidIO Interconnect Specification 2.1, "Annex 2: Session Management Protocol"
  - Supports basic stream management flow control (XON/XOFF) using extended header message format
- Up to 16 concurrent inbound reassembly operations
  - One additional reassembly context is reservable to a specific transaction type
- Support for outbound Type 11 messaging
- Support for outbound Type 5 NWRITE and Type 6 SWRITE transactions
- Support for inbound Type 11 messaging
- Support for inbound Type 9 data streaming transactions
- Support for outbound Type 9 data streaming transactions
  - Up to 64 KB total payload
- Support for inbound Type 10 doorbell transactions
  - Transaction steering through doorbell header classification
- Support for outbound Type 10 doorbell transactions
  - Ordering can be maintained with respect to other types of traffic.
- Support for inbound and outbound port-write transactions
  - Data payloads of 4 to 64 bytes

1.3.8.3 SATA

Each of the SoC's two SATA controllers is compliant with the *Serial ATA 2.6 Specification*. Each of the SATA controllers has the following features:

- Supports speeds: 1.5 Gbps (first-generation SATA), and 3Gbps (second-generation SATA)
- Supports advanced technology attachment packet interface (ATAPI) devices
- Contains high-speed descriptor-based DMA controller
- Supports native command queuing (NCQ) commands
- Supports port multiplier operation
- Supports hot plug including asynchronous signal recovery

1.3.9 Data Path Acceleration Architecture (DPAA)

This chip includes an enhanced implementation of the QorIQ Datapath Acceleration Architecture (DPAA). This architecture provides the infrastructure to support simplified sharing of networking interfaces and accelerators by multiple CPUs. These resources are abstracted as enqueue/dequeue operations by CPU 'portals' into the datapath. Beyond enabling multicore sharing of resources, the DPAA significantly reduces software overheads associated with high-touch packet-processing operations.

Examples of the types of packet-processing services that this architecture is optimized to support are as follows:

- Traditional routing and bridging
- Firewall
- Security protocol encapsulation and encryption
- Intrusion detection/prevention (IDS/IPS)
- Network anti-virus (AV)

The functions off-loaded by the DPAA fall into two broad categories:

- Packet distribution and queue-congestion management
- Accelerating content processing

1.3.9.1 Packet distribution and queue/congestion management

This table lists some packet distribution and queue/congestion management offload functions.

Table 1-1. Offload functions

Function type	Definition
Data buffer management	Supports allocation and deallocation of buffers belonging to pools originally created by software with configurable depletion thresholds. Implemented in a module called the Buffer Manager (BMan).
Queue management	Supports queuing and quality-of-service scheduling of frames to CPUs, network interfaces and DPAA logic blocks, maintains packet ordering within flows. Implemented in a module called the Queue Manager (QMan). The QMan, besides providing flow-level queuing, is also responsible for congestion management functions such as RED/WRED, congestion notifications and tail discards.
Packet distribution	Supports in-line packet parsing and general classification to enable policing and QoS-based packet distribution to the CPUs for further processing of the packets. This function is implemented in the block called the Frame Manager (FMan).
Policing	Supports in-line rate-limiting by means of two-rate, three-color marking (RFC 2698). Up to 256 policing profiles are supported. This function is also implemented in the FMan.
Egress Scheduling	Supports hierarchical scheduling and shaping, with committed and excess rates. This function is supported in the QMan, although the FMan performs the actual transmissions.

1.3.9.2 Accelerating content processing

Properly implemented acceleration logic can provide significant performance advantages over most optimized software with acceleration factors on the order of 10-100x. Accelerators in this category typically touch most of the bytes of a packet (not just headers). To avoid consuming CPU cycles in order to move data to the accelerators, these engines include well-pipelined DMAs. This table lists some specific content-processing accelerators on the chip.

Table 1-2. Content-processing accelerators

Interface	Definition
SEC	Crypto-acceleration for protocols such as IPsec, SSL, and 3G RLL
PME	Regex style pattern matching for unanchored searches, including cross-packet stateful patterns
DCE	Compression/Decompression acceleration for ZLib and deflate

1.3.9.3 DPAA terms and definitions

The QorIQ Platform's Data Path Acceleration Architecture (henceforth DPAA) assumes the existence of network flows, where a flow is defined as a series of network datagrams, which have the same processing and ordering requirements. The DPAA prescribes data structures to be initialized for each flow. These data structures define how the datagrams associated with that flow move through the DPAA. Software is provided a consistent interface (the software portal) for interacting with hardware accelerators and network interfaces.

All DPAA entities produce data onto frame queues (a process called enqueuing) and consume data from frame queues (dequeuing). Software enqueues and dequeues through a software portal (each vCPU has two software portals), and the FMan, RMan, and DPAA accelerators enqueue/dequeue through hardware portals. This figure illustrates this key DPAA concept.

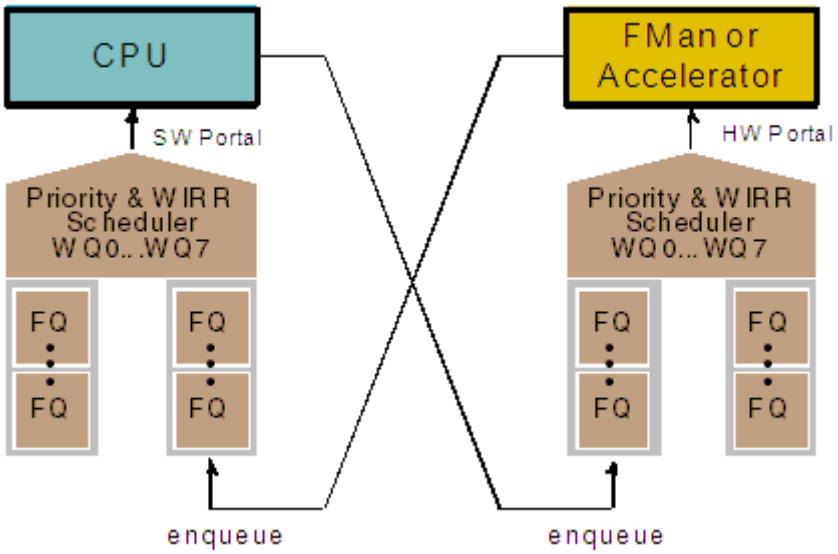


Figure 1-2. DPAA enqueuing and dequeuing

This table lists common DPAA terms and their definitions.

Table 1-3. DPAA terms and definitions

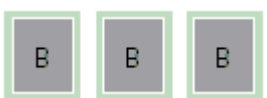
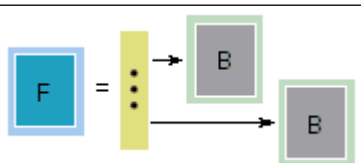
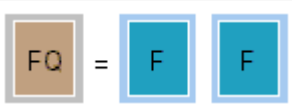
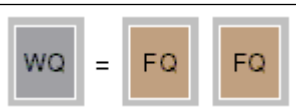
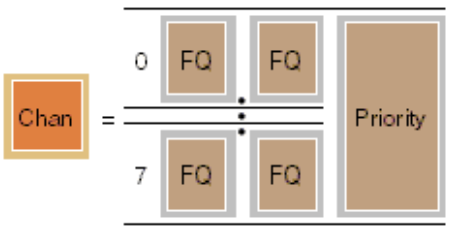
Term	Definition	Graphic representation
Buffer	Region of contiguous memory, allocated by software, managed by the DPAA BMan	
Buffer pool	Set of buffers with common characteristics (mainly size, alignment, access control)	
Frame	Single buffer or list of buffers that hold data, for example, packet payload, header, and other control information	
Frame queue (FQ)	FIFO of frames	
Work queue (WQ)	FIFO of FQs	

Table continues on the next page...

Table 1-3. DPAA terms and definitions (continued)

Term	Definition	Graphic representation
Channel	Set of eight WQs with hardware provided prioritized access	
Dedicated channel	Channel statically assigned to a particular end point, from which that end point can dequeue frames. End point may be a CPU, FMan, PME,DCE,RMan or SEC.	-
Pool channel	A channel statically assigned to a group of end points, from which any of the end points may dequeue frames.	

1.3.9.4 Major DPAA components

The SoC's Datapath Acceleration Architecture, shown in the figure below, includes the following major components:

- Frame Manager (FMan)
- Queue Manager (QMan)
- Buffer Manager (BMan)
- RapidIO Message Manager (RMan 1.0)
- Security Engine (SEC 5.2)
- Pattern Matching Engine (PME 2.1)
- Decompression and Compression Engine (DCE 1.0)

The QMan and BMan are infrastructure components, which are used by both software and hardware for queuing and memory allocation/deallocation. The Frame Managers and RMan are interfaces between the external world and the DPAA. These components receive datagrams via Ethernet or Serial RapidIO and queue them to other DPAA entities, as well as dequeue datagrams from other DPAA entities for transmission. The SEC, PME, and DCE are content accelerators that dequeue processing requests (typically from software) and enqueue results to the configured next consumer. Each component is described in more detail in the following sections.

This figure is a logical view of the DPAA.

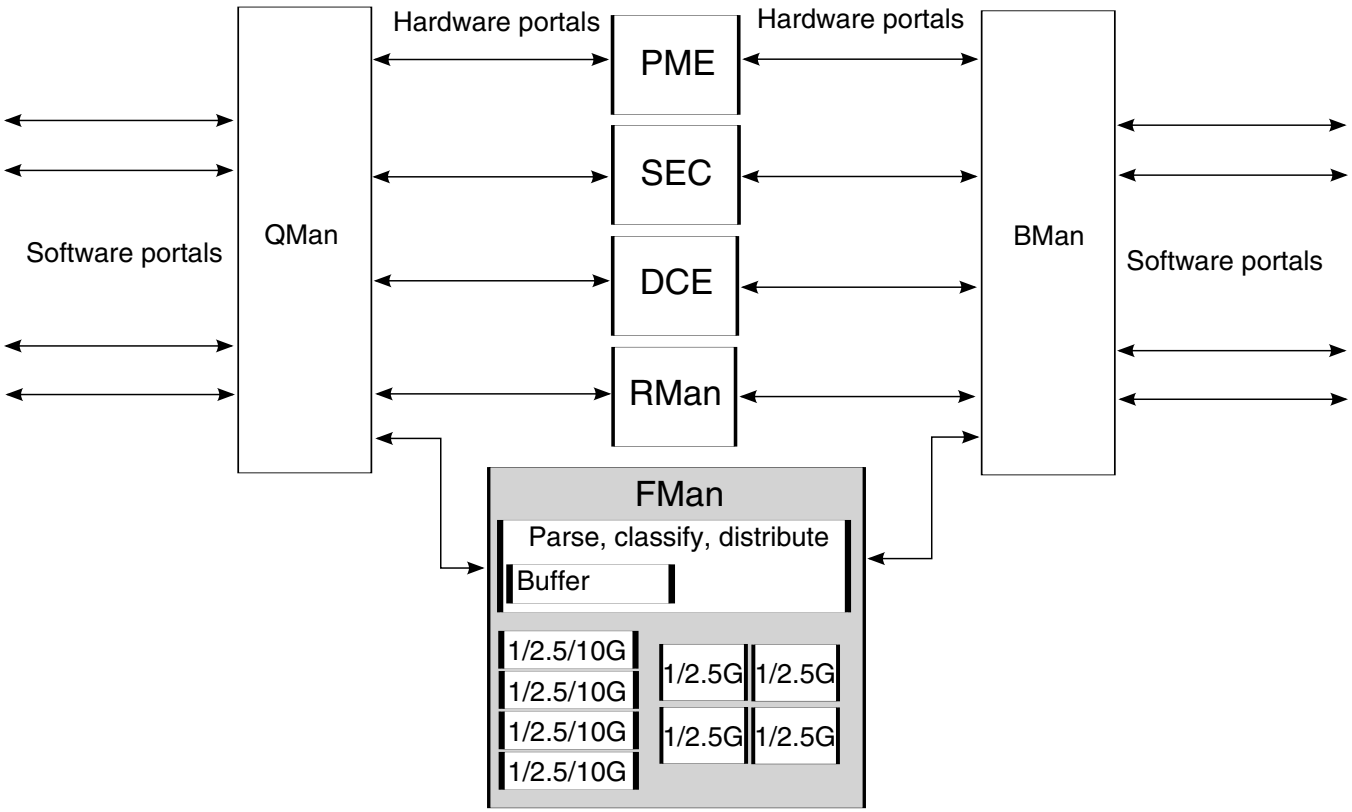


Figure 1-3. Logical representation of DPAA

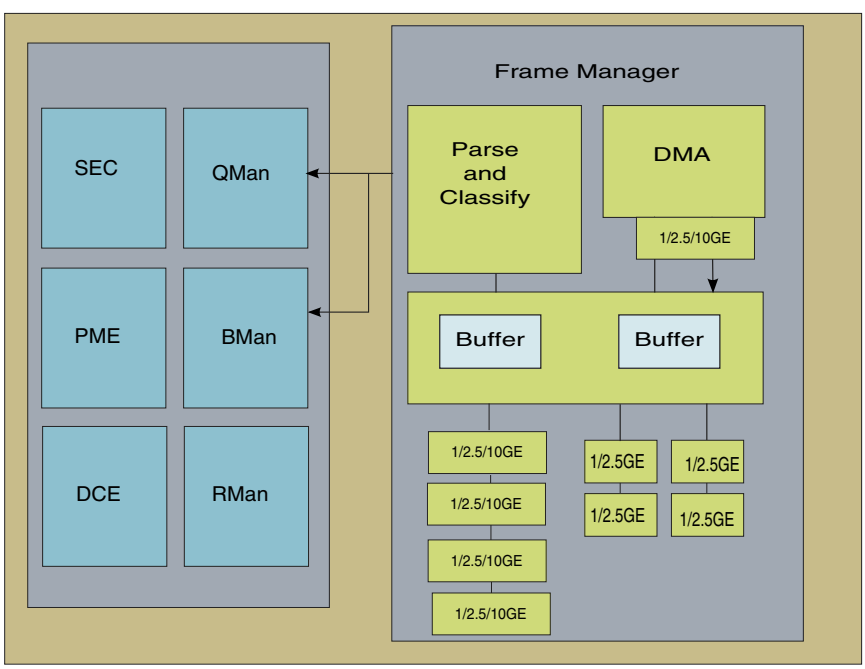


Figure 1-4. T2080 DPAA Components

1.3.9.4.1 Frame Manager and network interfaces

The Frame Manager, or FMan, combines Ethernet MACs with packet parsing and classification logic to provide intelligent distribution and queuing decisions for incoming traffic. The FMan supports PCD at 37.2 Mpps, supporting line rate 25 Gbps at minimum frame size.

The FMan is capable of connecting up to 44 Gbps (4 x 10G + 4 x 1 G). The total guaranteed bandwidth is 25 Gbps. In case more than 25 Gbps is connected, the 10G bit links support lossless flow control using the pause-frame mechanism. Only two of the 10G MACs can reach the 10G link rate.

These Ethernet combinations are supported with the following SerDes options:

- 12 Gbps Ethernet MACs are supported with Higig2 (four lanes at 3.75 GHz)
- 10 Gbps Ethernet MACs are supported with XAUI (four lanes at 3.125 GHz) or HiGig (four lanes at 3.125 GHz), XFI or 10Gbase-KR (one lane at 10.3125 GHz).
- 1 Gbps Ethernet MACs are supported with SGMII (one lane at 1.25 GHz with 3.125 GHz option for 2.5 Gbps Ethernet).
    - Two MACs can be used with RGMII.

The Frame Manager's Ethernet functionality also supports the following:

- 1588v2 hardware timestamping mechanism in conjunction with IEEE Std. 802.3bf (Ethernet support for time synchronization protocol)
- Energy Efficient Ethernet (IEEE Std. 802.3az)
- IEEE Std. 802.3bd (MAC control frame support for priority based flow control)
- IEEE Std. 802.1Qbb (Priority-based flow control) for up to eight queues/priorities
- IEEE Std. 802.1Qaz (Enhanced transmission selection) for three or more traffic classes

1.3.9.4.1.1 Receiver functionality: parsing, classification, and distribution

The Frame Manager matches its Ethernet support with 25 Gbps (37.2 Mpps) of Parsing, Classification, and Distribution (PCD) performance. PCD is the process by which the Frame Manager identifies the frame queue on which received packets should be enqueued. The consumer of the data on the frame queues is determined by Queue Manager configuration; however, these activities are closely linked and managed by the FMan Driver and FMan Configuration Tool, as in previous QorIQ SoCs.

This figure provides a logical view of the FMan's processing flow, illustrating the PCD features.

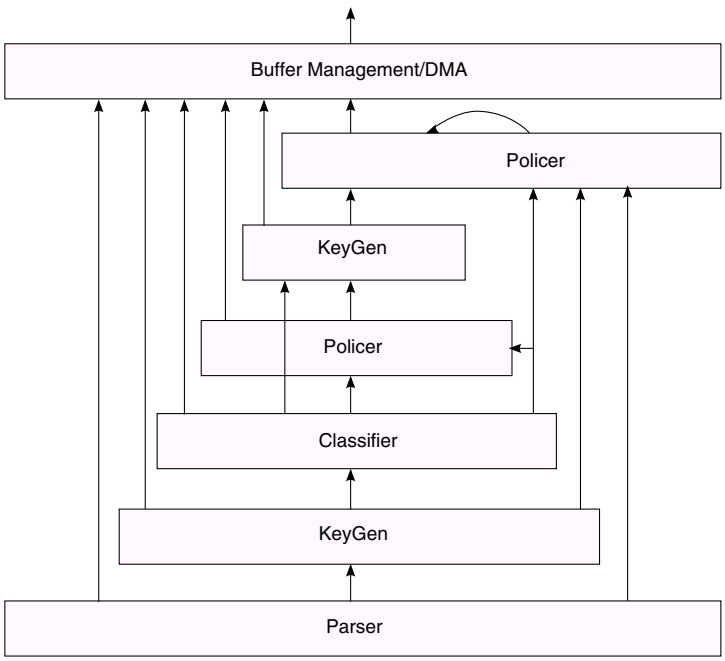


Figure 1-5. Logical view of FMan processing

Each frame received by the FMan is buffered internally while the Parser, KeyGen, and Classification functions operate.

The parse function can parse many standard protocols, including options and tunnels, and it supports a generic configurable capability to allow proprietary or future protocols to be parsed. Hard parsing of the standard protocol headers can be augmented with user-defined soft parsing rules to handle proprietary header fields. Hard and soft parsing occurs at wire speed.

This table defines several types of parser headers.

Table 1-4. Parser header types

Header type	Definition
Self-describing	Announced by proprietary values of Ethertype, protocol identifier, next header, and other standard fields. They are self-describing in that the frame contains information that describes the presence of the proprietary header.
Non-self-describing	Does not contain any information that indicates the presence of the header. For example, a frame that always contains a proprietary header before the Ethernet header would be non-self-describing. Both self-describing and non-self-describing headers are supported by means of parsing rules in the FMan.
Proprietary	Can be defined as being self-describing or non-self-describing

The underlying notion is that different frames may require different treatment, and only through detailed parsing of the frame can proper treatment be determined.

Parse results can (optionally) be passed to software.

1.3.9.4.1.2 FMan distribution and policing

After parsing is complete, there are two options for treatment, as shown in this table.

Table 1-5. Post-parsing treatment options

Treatment	Function	Benefits
Hash	• Hashes select fields in the frame as part of a spreading mechanism. • The result is a specific frame queue identifier. • To support added control, this FQID can be indexed by values found in the frame, such as TOS or p-bits, or any other desired field(s).	Useful when spreading traffic while obeying QoS constraints is required
Classification look-up	• Looks up certain fields in the frame to determine subsequent action to take, including policing. • The FMan contains internal memory that holds small tables for this purpose. • The user configures the sets of lookups to perform, and the parse results dictate which one of those sets to use. • Lookups can be chained together such that a successful look-up can provide key information for a subsequent look-up. After all the look-ups are complete, the final classification result provides either a hash key to use for spreading, or a FQ ID directly.	• Useful when hash distribution is insufficient and a more detailed examination of the frame is required • Can determine whether policing is required and the policing context to use

Key benefits of the FMan policing function are as follows:

- Because the FMan has up to 256 policing profiles, any frame queue or group of frame queues can be policed to either drop or mark packets if the flow exceeds a preconfigured rate.
- Policing and classification can be used in conjunction to mitigate Distributed Denial of Service Attack (DDOS).
- The policing is based on the two-rate-three-color marking algorithm (RFC2698). The sustained and peak rates, as well as the burst sizes, are user-configurable. Therefore, the policing function can rate-limit traffic to conform to the rate that the flow is mapped to at flow set-up time. By prioritizing and policing traffic prior to software processing, CPU cycles can focus on important and urgent traffic ahead of other traffic.

Each FMan also supports PCD on traffic arriving from within the chip. This is referred to as off-line parsing, and it is useful for reclassification following decapsulation of encrypted or compressed packets.

FMan PCD supports virtualization and strong partitioning by delaying buffer pool selection until after classification. In addition to determining the FQ ID for the classified packet, the FMan also determines the 'storage profile.' Configuration of storage profiles (up to 32 per physical port) allows the FMan to store received packets using buffer pools owned by a single software partition, and enqueue the associated Frame Descriptor to a frame queue serviced by only that software partition. This capability includes copying from one buffer pool to another if the traffic is received via the FMan's off-line parsing port. Packets can be copied to multiple buffer pools and enqueued to multiple frame queues to support broadcast and multicast requirements.

1.3.9.4.2 Queue Manager

The Queue Manager (QMan) is the primary infrastructure component in the DPAA, allowing for simplified sharing of network interfaces and hardware accelerators by multiple CPU cores. It also provides a simple and consistent message and data passing mechanism for dividing processing tasks amongst multiple vCPUs.

The Queue Manager offers the following features:

- Common interface between software and all hardware
  - Controls the prioritized queuing of data between multiple processor cores, network interfaces, and hardware accelerators.
  - Supports both dedicated and pool channels, allowing both push and pull models of multicore load spreading.
- Atomic access to common queues without software locking overhead
- Mechanisms to guarantee order preservation with atomicity and order restoration following parallel processing on multiple CPUs
- Egress queuing for Ethernet interfaces
  - Hierarchical (2-level) scheduling and dual-rate shaping
  - Dual-rate shaping to meet service-level agreements (SLAs) parameters (1 Kbps... 10 Gbps range, 1 Kbps granularity across the entire range)
  - Configurable combinations of strict priority and fair scheduling (weighted queuing) between the queues
  - Algorithms for shaping and fair scheduling are based on bytes
- Queuing to cores and accelerators
  - Two level queuing hierarchy with one or more Channels per Endpoint, eight work queues per Channel, and numerous frame queues per work queue
  - Priority and work conserving fair scheduling between the work queues and the frame queues
- Loss-less flow control for ingress network interfaces
- Congestion avoidance (RED/WRED) and congestion management with tail discard

1.3.9.4.3 Buffer Manager

The Buffer Manager (BMan) manages pools of buffers on behalf of software for both hardware (accelerators and network interfaces) and software use.

The Buffer Manager offers the following features:

- Common interface for software and hardware
- Guarantees atomic access to shared buffer pools
- Supports 64 buffer pools
  - Software, hardware buffer consumers can request different size buffers and buffers in different memory partitions
- Supports depletion thresholds with congestion notifications
- On-chip per pool buffer stockpile to minimize access to memory for buffer pool management
- LIFO (last in first out) buffer allocation policy
  - Optimizes cache usage and allocation
  - A released buffer is immediately used for receiving new data

1.3.9.4.4 Pattern Matching Engine (PME 2.1)

The PME 2.1 is Freescale's second generation of extended NFA style pattern matching engine. Unchanged from the first generation QorIQ products, it supports ~10 Gbps data scanning.

Key benefits of a NFA pattern matching engine:

- No pattern "explosion" to support "wildcarding" or case-insensitivity
  - Comparative compilations have shown 300,000 DFA pattern equivalents can be achieved with ~8000 extended NFA patterns
- Pattern density much higher than DFA engines.
  - Patterns can be stored in on-chip tables and main DDR memory
  - Most work performed solely with on-chip tables (external memory access required only to confirm a match)
  - No need for specialty memories; for example, QDR SRAM, RLDRAM, and so on.
- Fast compilation of pattern database, with fast incremental additions
  - Pattern database can be updated without halting processing
  - Only affected pattern records are downloaded
  - DFA style engines can require minutes to hours to recompile and compress database

Freescale's basic NFA capabilities for byte pattern scanning are as follows:

- The PME's regex compiler accepts search patterns using syntax similar to that in software-based regex engines, such as Perl-Compatible Regular Expression (PCRE).
  - Supports Perl meta-characters including wildcards, repeats, ranges, anchors, and so on.
  - Byte patterns are simple matches, such as gabcd123h, existing in both the data being scanned and in the pattern specification database.
- Up to 32 KB patterns of length 1-128 bytes

Freescale's extensions to NFA style pattern matching are principally related to event pattern scanning. Event patterns are sequences of byte patterns linked by 'stateful rules.' Freescale uses event pattern scanning and stateful rule processing synonymously. Stateful rules are hardware instructions by which users define reactions to pattern match events, such as state changes, assignments, bitwise operations, addition, subtraction, and comparisons.

Some key characteristics and benefits of the Stateful Rule extensions include:

- Ability to match patterns across data "work units" or packet boundaries
  - Can be used to correlate patterns, qualify matches (for example, contextual match), or to track protocol state change
- Easily support "greedy" wildcards
  - For example, ABC.*DEF == two patterns tied together by a stateful rule
- Delays the need for software post-processing. Software is alerted after all byte patterns are detected in the proper sequence, rather than any time a byte pattern is detected.
- Implements a significant subset of the regex pattern definition syntax as well as many constructs which cannot be expressed in standard PCRE
- PME 2.1 supports up to 32K stateful rules, linking multiple byte patterns

The PME 2.1 dequeues data from its QMan hardware portal and, based on FQ configuration, scans the data against one of 256 pattern sets, 16 subsets per pattern set.

When the PME finds a byte pattern match, or a final pattern in a stateful rule, it generates a report.

1.3.9.4.5 RapidIO Message Manager (RMan 1.0)

The RapidIO message manager (RMan) produces and consumes Type 8 Port-write, Type 9 Data Streaming, Type 10 Doorbells and Type 11 Messaging traffic and is capable of producing Type 5 NWRITE and Type 6 SWRITE transactions. For inbound traffic, the RMan supports up to 17 open reassembly contexts as a arbitrary mix of Type 9, and Type 11 traffic.

As ingress packets arrive at the RMan, they are compared against up to 64 classification rules to determine the target queue. These rules support Type 8, 9, 10 and 11 transaction types. They may be wild-carded and are configured as masks over selected header fields. This table lists the fields that are maskable as part of each classification rule.

Table 1-6. Maskable fields in each classification rule

Classification rule	Field
Transaction types	RapidIO port
	Source device ID
	Destination device ID
	Flow level
Type 9 messaging-specific	Class-of-service (CoS)
	StreamID
Type 11 messaging-specific	Mailbox
	Extended mailbox
	Letter

Should the packet remain unclassified, the traffic is retried with an error in the case of Type 10 and 11 traffic and dropped in the case of Type 9 traffic. Dropped traffic is logged and upon a threshold can assert an error interrupt.

Classification allows Type 9, 10 and 11 traffic to be distributed across 64 possible Frame queues. A single dedicated inbound Type 8 Port-write Frame queue is provided. For all outbound traffic types (Type 8, 9, 10 and 11), the Data Path Acceleration Architecture allows a very large number of outbound Frame queues effectively limited by system, software and performance constraints.

The RMan is DPAA entity designed to work in conjunction with the chip's Serial RapidIO controllers. This figure illustrates RMan use cases.

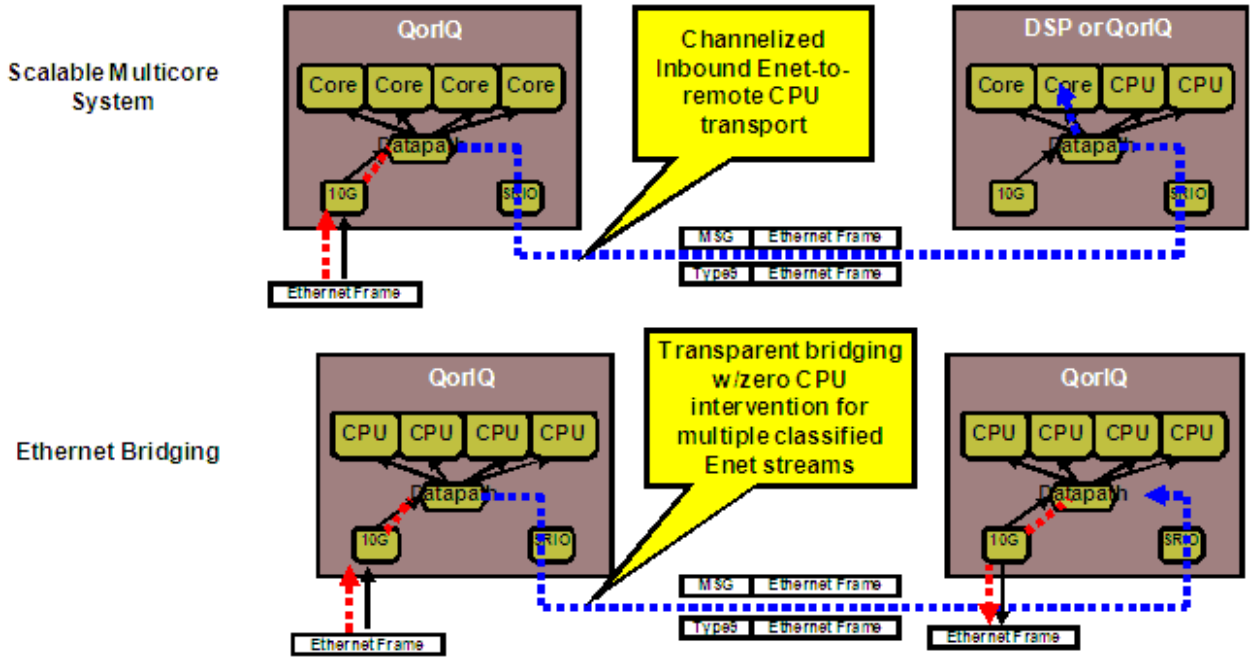


Figure 1-6. RMan use cases

Inbound Serial RapidIO traffic, including messages (Type 11), doorbells (Type 10), and data (Type 9) are classified by the RMan and enqueued to a configured FQ, allowing the DPAA to deliver the "data" to any DPAA consumer, including vCPUs, accelerators, or Ethernet ports (FMan). Outbound traffic enqueued to the RMan for transmission is given a configured ID, allowing the target FQ on the receiving device to be identified. The RMan/Serial RapidIO combination is particularly useful for chip-to-chip communication, with an x4 Serial RapidIO interface providing up to 16 Gbps of data/message bandwidth between RMan enabled QorIQ chips.

1.3.9.4.6 SEC 5.2

The SEC 5.2 can perform full protocol processing for the following security protocols:

- IPsec
- SSL/TLS
- 3GPP RLC encryption/decryption
- LTE PDCP
- SRTP
- IEEE 802.1AE MACSec
- IEEE 802.16e WiMax MAC layer

The SEC 5.2 supports the following algorithms, modes, and key lengths as raw modes, or in combination with the security protocol processing described above.

- Public-key hardware accelerators (PKHA)

    - RSA and Diffie-Hellman (to 4096b)
    - Elliptic curve cryptography (1023b)
- Data-encryption standard accelerators (DESA)
    - DES, 3DES (2-key, 3-key)
    - ECB, CBC, OFB, and CFB modes
- Advanced-encryption standard accelerators (AESA)
    - Key lengths of 128-bit, 192-bit, and 256-bit
    - Confidentiality modes
        - ECB, CBC, OFB, CFB, CTR and XTS
    - Authenticated encryption modes
        - CCM and GCM
- ARC four hardware accelerators (AFHA)
    - Compatible with RC4 algorithm
- Message digest hardware accelerators (MDHA)
    - SHA-1, SHA-256, 384, 512-bit digests
    - MD5 128-bit digest
    - HMAC with all algorithms
- Kasumi/F8 hardware accelerators (KFHA)
    - F8, F9 as required for 3GPP
    - A5/3 for GSM and EDGE, GEA-3 for GPRS
- Snow 3G hardware accelerators (STHA)
    - Implements Snow 3.0, F8 and F9 modes
- ZUC Hardware Accelerators (ZHA)
    - Implements 128-EEA3 & 128-EIA3
- CRC Unit
    - Standard and user-defined polynomials
- Random-number generator (RNG)
    - Incorporates TRNG entropy generator for seeding and deterministic engine (SHA-256)
    - Supports random IV generation
- DTLS
- IEEE Std 802.11 WiFi

Protocol	Cipher suite	Performance (aggregate encap and decap)
IPsec	AES-CBC/AES-XCBC-MAC	4.4 Gbps
LTE PDCP U-plane	128-EEA2 (AES)	8.8 Gbps
LTE PDCP C-plane	128-EEA3 and 128-EIA3 (ZUC)	3.5 Gbps

The SEC dequeues data from its QMan hardware portal and, based on FQ configuration, also dequeues associated instructions and operands in the Shared Descriptor. The SEC processes the data then enqueues it to the configured output FQ. The SEC uses the Status/ CMD word in the output Frame Descriptor to inform the next consumer of any errors encountered during processing (for example, received packet outside the anti-replay window.)

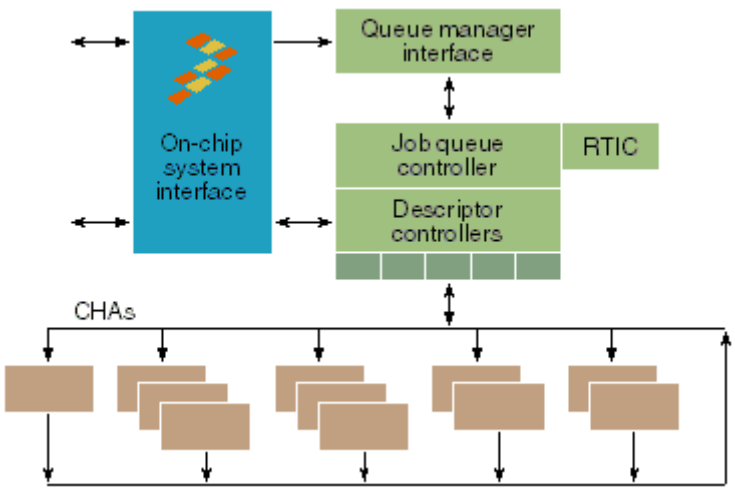


Figure 1-7. SEC 5.2 block diagram

The SEC 5.2 is also part of the QorIQ Platform's Trust Architecture, which gives the SoC the ability to perform secure boot, runtime code integrity protection, and session key protection. The Trust Architecture is described in Resource partitioning and QorIQ Trust Architecture.

1.3.9.4.7 Decompression and Compression Engine (DCE 1.0)

The Decompression and Compression Engine (DCE 1.0) is an accelerator compatible with Datapath Architecture providing lossless data decompression and compression for the QorIQ family of SoCs. The DCE supports the raw DEFLATE algorithm (RFC1951), GZIP format (RFC1952) and ZLIB format (RFC1950). The DCE also supports Base 64 encoding and decoding (RFC4648).

The DEFLATE algorithm is a basic building block for data compression in most modern communication systems. It is used by HTTP to compress web pages, by SSL to compress records, by gzip to compress files and email attachments, and by many other applications.

Deflate involves searching for repeated patterns previously seen in a Frame, computing the length and the distance of the pattern with respect to the current location in the Frame, and encoding the resulting information into a bitstream.

The decompression algorithm involves decoding the bitstream and replaying past data. The Decompression and Compression Engine is architected to minimize the system memory bandwidth required to do decompression and compression of Frames while providing multi-gigabits per second of performance.

Detailed features include the following:

- Deflate; as specified as in RFC1951
- GZIP; as specified in RFC1952
- Zlib; as specified in RFC1950
    - Interoperable with the zlib 1.2.5 compression library
- Compression
    - ZLIB, GZIP and DEFLATE header insertion
    - ZLIB and GZIP CRC computation and insertion
    - Zlib sync flush and partial flush for chunked compression (for example, for HTTP1.1)
    - Four modes of compression
        - No compression (just add DEFLATE header)
        - Encode only using static/dynamic Huffman codes
        - Compress and encode using static Huffman codes
        - Compress and encode using dynamic Huffman codes
    - Uses a 4KB sliding history window
    - Supports Base 64 encoding (RFC4648) after compression
    - Provides at least 2.5:1 compression ratio on the Calgary Corpus
- Decompression supports:
    - ZLIB, GZIP and DEFLATE header removal
    - ZLIB and GZIP CRC validation
    - 32KB history
    - Zlib flush for chunked decompression (for HTTP1.1 for example)
        - All standard modes of decompression
        - No compression
        - Static Huffman codes
        - Dynamic Huffman codes
    - Provides option to return original compressed Frame along with the uncompressed Frame or release the buffers to BMan
    - Does not support use of ZLIB preset dictionaries (FDICT flag = 1 is treated as an error).
    - Base 64 decoding (RFC4648) prior to decompression

The DCE 1.0 is designed to support up to 8.8 Gbps for either compression or decompression, or 17.5 Gbps aggregate at ~4 KB data sizes.

1.3.9.5 DPAA capabilities

Some DPAA features and capabilities have been described in the sections covering individual DPAA components. This section describes some capabilities enabled by DPAA components working together.

1.3.9.5.1 Ingress policing and congestion management

In addition to selecting FQ ID and storage profile, classification can determine whether policing is required for a received packet, along with the specific policing context to be used.

FMan policing capabilities include the following:

- RFC2698: two-rate, three-color marking algorithm
- RFC4115: Differentiated service two-rate, three-color marker with efficient handling of in-profile traffic
- Up to 256 internal profiles

The sustained and peak rates, and burst size for each policing profile are user-configurable.

Congestion management also supports WRED and flow control based on congestion groups.

1.3.9.5.2 Customer-edge egress-traffic management (CEETM)

Customer-edge egress-traffic management (CEETM) is a DPAA enhancement first appearing in the T4240. T2080 continues to support the work queue and frame queue scheduling functionality available in the P4080 and other first generation QorIQ chips, but introduces alternative functionary, CEETM, that can be mode selected on a network interface basis to support the shaping and scheduling requirements of carrier Ethernet connected systems.

1.3.9.5.2.1 CEETM features

Each instance of CEETM (one per FMan) provides the following features:

- Supports hierarchical multi-level scheduling and shaping, which:

    - is performed in an atomic manner; all context at all levels is examined and updated synchronously.
    - employs no intermediate buffering between class queues and the direct connect portal to the FMan.
- Supports dual-rate shaping (paired committed rate (CR) shaper and excess rate (ER) shaper) at all shaping points.
    - Shapers are token bucket based with configurable rate and burst limit.
    - Paired CR/ER shapers may be configured as independent or coupled on a per pair basis; coupled means that credits to the CR shaper in excess of its token bucket limit is credited to the ER bucket
- Supports eight logical network interfaces (LNI)
    - Each LNI:
        - aggregates frames from one or more channels.
        - priority schedules unshaped frames (aggregated from unshaped channels), CR frames, and ER frames (aggregated from shaped channels)
        - applies a dual-rate shaper to the aggregate of CR/ER frames from shaped channels
        - can be configured (or reconfigured for lossless interface failover) to deliver frames to any network interface.
- Supports 32 channels available for allocation across the eight LNIs
- Each channel:
    - can be configured to deliver frames to any LNI.
    - can be configure to be unshaped or shaped; when shaped, a dual rate shaper applies to the aggregate of CR/ER frames from the channel.
    - has eight independent classes and eight grouped classes; grouped classes can be configured as one class group of eight or as two class groups of four.
    - supports weighted bandwidth fairness within grouped class groups with weights configured on a channel and class basis.
    - strict priority scheduling of the eight independent classes and the aggregate(s) of the grouped classe(s); the priority of each of the two class groups can be independently configured to be immediately below any of the independent classes.
    - is configurable such that each of the eight independent classes and two class groups can supply CR frames, ER frames or both when channel is configured to be shaped.
    - is configured independently.
- Each class:
    - has a dedicated class queue (CQ) with equivalent congestion management functionality available to FQs.
    - can have a dedicated or shared Congestion Management Record supports sufficient number of CMRs for all CQs to have a dedicated CMR, if desired.

- can be flow-controlled by traffic-class flow control messages via portal; achieves backward compatibility with by allowing each of these 16 classes to be configured (per LNI) to respect one or none of the 8 on/off control bits within existing message format (as was defined for 8-class non-CEETM channels).
- is identified via a "logical frame queue identifier" to maintain semantic compatibility with enqueue commands to frame queues (non-CEETM queues).
- supports the identification of intra-class flows (logically equivalent to FQs but not queued separately) in order to apply static context (Context_A and Context_B) to frames as they are dequeued from CQs; this provides functionality equivalent to that available when a frame is dequeue from a frame queue (non-CEETM queues).

1.3.9.5.2.2 CEETM configuration

The CEETM configuration, shown in Figure 1-8, is very asymmetrical and is intended to demonstrate the degrees of configurability rather than an envisioned use case.

NOTE

The color green denotes logic units and signal paths that relate to the request and fulfillment of committed rate (CR) packet transmission opportunities. The color yellow denotes the same for excess rate (ER). The color black denotes logic units and signal paths that are used for unshaped opportunities or that operate consistently whether used for CR or ER opportunities.

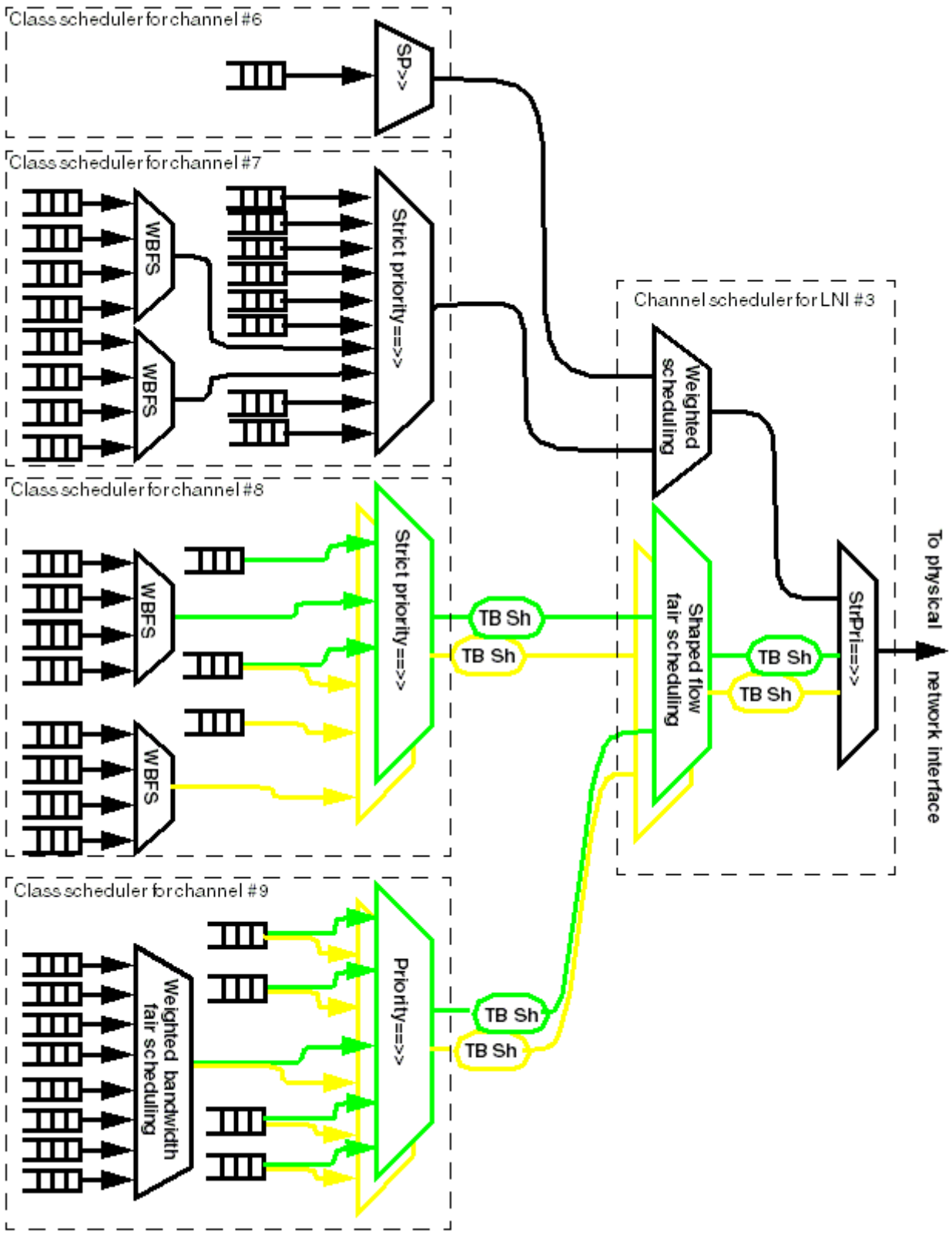


Figure 1-8. CEETM scheduler: illustrative configuration scenario

Figure 1-8 illustrates the following scenario:

- Channels #6, #7, #8 and #9 have been configured to be scheduled by the channel scheduler for LNI#3 (for example, all the packets from these channels are directed to the physical network interface configurably coupled to LNI#3).
- Channels #6 and #7 have been configured to be "unshaped." Packets from these channels will not be subjected to shaping at the channel level and will feed the top priority level within the LNI, which is also not subjected to shaping. Their class schedulers will not distinguish between CR and ER opportunities.
- Channels #8 and #9 have been configured to be "shaped." Their class schedulers will distinguish between CR and ER opportunities. The CR/ER packets to be sent from each channel shall be subjected to a pair of CR/ER token bucket shapers specific to that channel. The aggregate of CR/ER packets from these channels are subject to a pair of CR/ER token bucket shapers specific to LNI#3.
- Channel #6 has only one class in use. That class queue behaves as if it were a channel queue and as a peer to Channel #7. Unused classes do not have to be configured as such; they are simply not used.
- Channel #7 has all 16 classes in use.
  - The group classes have been configured as two groups (A and B) of four classes.
  - The priority of the groups A and B have both been set to be immediately below independent class 5. In a case of similar configuration group A has higher priority than group B.
- Channel #8 has three independent classes and two groups of four grouped classes in use.
  - The priorities of the class groups A and B have been set to be immediately below independent class 0 and class 2 respectively.
  - Independent class 0 and class group A have been configured to request and fulfill only CR packet opportunities.
  - Independent class 1 has been configured to request and fulfill both CR and ER packet opportunities.
  - Independent class 2 and class group B have been configured to request and fulfill only ER packet opportunities.
- Channels #9 has four independent classes and one group of eight grouped classes in use.
  - The group classes have been configured as one group (A) of eight classes.
  - All independent classes and the class group (A) have been configured to request and fulfill both CR and ER packet opportunities.

Benefits of the CEETM include the following:

- Provides "virtual" ports for multiple applications or users with different QoS/CoS requirements which are sharing an egress interface
- Supports DSCP capable scheduling for the following virtual link with configurable combinations of strict priority and weighted scheduling

    - Weighted scheduling closely approximating WFQ
- Supports traffic shaping
    - dual rate shaping of the virtual links
- Supports aggregating traffic from multiple virtual links and shaping this aggregate
- Hierarchical scheduling and shaping
- Class-based scheduling and dual rate shaping
- Supports a subset of the IEEE Data Center Bridging (DCB) standards

1.3.9.5.3 Data Center Bridging (DCB)

Data Center Bridging (DCB) refers to a series of inter-related IEEE specifications collectively designed to enhance Ethernet LAN traffic prioritization and congestion management. Although the primary objective is the data center environment (consisting of servers and storage arrays), some aspects of DCB are applicable to more general uses of Ethernet, within and between network nodes.

The SoC DPAA is compliant with the following DCB specifications on the 10 Gbps ports:

- IEEE Std. 802.1Qbb: Priority-based flow control (PFC)
    - PAUSE frame per Ethernet priority code point (8)
    - Prevents single traffic class from throttling entire port
- IEEE Std. 802.1Qaz: Enhanced transmission selection (ETS)
    - Up to three Traffic Class Groups (TCG), where a TCG is composed of one or more priority code points
    - Bandwidth allocation and transmit scheduling (1% granularity) by traffic class group
    - If one of the TCGs does not consume its allocated bandwidth, unused bandwidth is available to other TCGs

1.3.10 OCeaN DMA

The OCeaN fabric is used to:

- Transfer general data between two memory locations

- Eight high-speed/high-bandwidth channels
- Basic DMA operation modes (direct, simple chaining)
- Extended DMA operation modes (advanced chaining and stride capability)
- Programmable bandwidth control between channels
- Three priority levels supported for source and destination transactions
- Can be activated using DREQ pin

- Optimized to work with the high speed interfaces
- Address translation and mapping unit (ATMU) which allows to define packet attributes as address/device/flow level/transaction type. ATMU Bypass that allows the descriptor to specify the attributes.

1.3.11 Serial memory controllers

In addition to the parallel NAND and NOR flash supported by the IFC, the SoC supports serial flash using eSPI and SDXC/eMMC card and device interfaces. The SDXC/eMMC controller includes a DMA engine, allowing it to move data from serial flash to external or internal memory following straightforward initiation by software.

Detailed features of the eSDXC include the following:

- Conforms to the SD Host Controller Standard Specification version 3.0, including Test Event Register support
- Compatible with the MMC System Specification version 4.5
- Compatible with the SD Memory Card Physical Layer Specification version 3.0.1, and supports the high-capacity SD memory card
- Compatible with the SDIO Card Specification version 3
- Designed to work with SD memory, SDIO, SD combo, MMC, and their variants like mini, micro, embedded, etc..
- Card bus clock frequency up to 52 MHz
- Supports 1-/4-bit SD and SDIO modes, 1-/4-/8-bit MMC 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 pin and SDHC_DAT3 pin depending on the Protocol Control Register
- Supports both synchronous and asynchronous abort
- Supports pause during the data transfer at block gap
- Supports SDIO read wait and suspend resume operations
- Supports Auto CMD12 for multi-block transfer
- Host can initiate command that do 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
- Supports external SD bus voltage selection by register configuration

- Host sends 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 external voltage translator control signals for 3.3V operation
- 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

1.3.11.1 PreBoot Loader and nonvolatile memory interfaces

The PreBoot Loader (PBL) operates on behalf of a large number of interfaces.

1.3.11.1.1 PreBoot Loader (PBL)

The PBL's functions include the following:

- Simplifies boot operations, replacing pin strapping resistors with configuration data loaded from nonvolatile memory
- Uses the configuration data to initialize other system logic and to copy data from low speed memory interfaces (I^2C, IFC, eSPI, SD/SDXC/eMMC) into fully initialized DDR or other access targets in the chip
- Releases CPU 0 from reset, allowing the boot processes to begin from fast system memory

1.3.11.1.2 Integrated Flash Controller

The SoC incorporates an Integrated Flash Controller similar to the one used in some previous generation QorIQ SoCs. The IFC supports both NAND and NOR flash, as well as a general purpose memory mapped interface for connecting low speed ASICs and FPGAs.

1.3.11.1.2.1 NAND Flash features

- x8/x16 NAND Flash interface
- Optional ECC generation/checking
- Flexible timing control to allow interfacing with proprietary NAND devices
- SLC and MLC Flash devices support with configurable page sizes of up to 8 KB
- Support advance NAND commands like cache, copy-back, and multiplane programming

- Boot chip-select (CS0) available after system reset, with boot block size of 8 KB, for execute-in-place boot loading from NAND Flash
- Up to terabyte Flash devices supported

1.3.11.1.2.2 NOR Flash features

- Data bus width of 8/16
- Compatible with asynchronous NOR Flash
- Directly memory mapped
- Supports address data multiplexed (ADM) NOR device
- Flexible timing control allows interfacing with proprietary NOR devices
- Boot chip-select (CS0) available at system reset

1.3.11.1.2.3 General-purpose chip-select machine (GPCM)

The IFC's GPCM supports the following features:

- Normal GPCM
  - Support for x8/16-bit device
  - Compatible with general purpose addressable device, for example, SRAM and ROM
  - External clock is supported with programmable division ratio (2, 3, 4, and so on, up to 16)
- Generic ASIC Interface
  - Support for x8/16-bit device
  - Address and Data are shared on I/O bus
  - Following address and data sequences are supported on I/O bus:
    - 16-bit I/O: AADD
    - 8-bit I/O: AAAADDDD

1.3.12 Resource partitioning and QorIQ Trust Architecture

Consolidation of discrete CPUs into a single, multicore chip introduces many opportunities for unintended resource contentions to arise, particularly when multiple, independent software entities reside on a single chip. A system may exhibit erratic behavior if multiple software partitions cannot effectively partition resources. Device consolidation, combined with a trend toward embedded systems becoming more open (or more likely to run third-party or open-source software on at least one of the cores), creates opportunities for malicious code to enter a system.

This chip offers a new level of hardware partitioning support, allowing system developers to ensure software running on any CPU only accesses the resources (memory, peripherals, and so on) that it is explicitly authorized to access. This section provides an overview of the features implemented in the chip that help ensure that only trusted software executes on the CPUs, and that the trusted software remains in control of the system with intended isolation.

1.3.12.1 Core MMU, UX/SX bits, and embedded hypervisor

The chip's first line of defense against unintended interactions amongst the multiple CPUs/OSes is each core vCPU's MMU. A vCPU's MMU is configured to determine which addresses in the global address map the CPU is able to read or write. If a particular resource (memory region, peripheral device, and so on) is dedicated to a single vCPU, that vCPU's MMU is configured to allow access to those addresses (on 4 KB granularity); other vCPU MMUs are not configured for access to those addresses, which makes them private. When two vCPUs need to share resources, their MMUs are both configured so that they have access to the shared address range.

This level of hardware support for partitioning is common today; however, it is not sufficient for many core systems running diverse software. When the functions of multiple discrete CPUs are consolidated onto a single multicore chip, achieving strong partitioning should not require the developer to map functions onto vCPUs that are the exclusive owners of specific platform resources. The alternative, a fully open system with no private resources, is also unacceptable. For this reason, the core's MMU also includes three levels of access permissions: user, supervisor (OS), and hypervisor. An embedded hypervisor (for example, KVM, XEN, QorIQ ecosystem partner hypervisor) runs unobtrusively beneath the various OSes running on the vCPUs, consuming CPU cycles only when an access attempt is made to an embedded hypervisor-managed shared resource.

The embedded hypervisor determines whether the access should be allowed and, if so, proxies the access on behalf of the original requestor. If malicious or poorly tested software on any vCPU attempts to overwrite important device configuration registers (including vCPU's MMU), the embedded hypervisor blocks the write. High and low-speed peripheral interfaces (PCI Express, UART), when not dedicated to a single vCPU/partition, are other examples of embedded hypervisor managed resources. The degree of security policy enforcement by the embedded hypervisor is implementation-dependent.

In addition to defining regions of memory as being controlled by the user, supervisor, or hypervisor, the core MMU can also configure memory regions as being non-executable. Preventing CPUs from executing instructions from regions of memory used as data buffers is a powerful defense against buffer overflows and other runtime attacks. In

previous generations of Power Architecture, this feature was controlled by the NX (no execute) attribute. In new Power Architecture cores such as the e6500 core, there are separate bits controlling execution for user (UX) and supervisor (SX).

1.3.12.2 Peripheral access management unit (PAMU)

MMU-based access control works for software running on CPUs; however, these are not the only bus masters in the SoC. Internal components with bus mastering capability (FMan, RMan, PCI Express controller, PME, SEC, and so on) also need to be prevented from reading and writing to certain memory regions. These components do not spontaneously generate access attempts; however, if programmed to do so by buggy or malicious software, any of them could read or write sensitive data registers and crash the system. For this reason, the SoC also includes a distributed function referred to as the peripheral access management unit (PAMU).

PAMUs provide address translation and access control for all non-CPU initiators in the system. PAMU access control is based on the logical I/O device number (LIODN) advertised by a bus master for a given transaction. LIODNs can be static (for example, PCI Express controller #1 always uses LIODN 123) or they can be dynamic, based on the ID of the CPU that programmed the initiator (for example, the SEC uses LIODN 456 because it was given a descriptor by vCPU #2). In the dynamic example, the SoC architecture provides positive identification of the vCPU programming the SEC, preventing LIODN spoofing.

1.3.12.3 IO partitioning

The simplest IO configuration in chips running multiple independent software partitions is to dedicate specific IO controllers (PCI Express, SATA, Serial RapidIO controllers) to specific vCPUs. The core MMUs and PAMUs can enforce these access permissions to insure that only the software partition owning the IO is able to use it. The obvious problem with this approach is that there are likely to be more software partitions wanting IO access than there are IO controllers to dedicate to each.

Safe IO sharing can be accomplished through the use of a hypervisor; however, there is a performance penalty associated with virtual IO, as the hypervisor must consume CPU cycles to schedule the IO requests and get the results back to the right software partition.

The DPAA (described in Data Path Acceleration Architecture (DPAA)") was designed to allow multiple partitions to efficiently share accelerators and IOs, with its major capabilities centered around sharing Ethernet ports. These capabilities were enhanced in the chip with the addition of FMan storage profiles. The chip's FMans perform

classification prior to buffer pool selection, allowing Ethernet frames arriving on a single port to be written to the dedicated memory of a single software partition. This capability is fully described in Receiver functionality: parsing, classification, and distribution."

The addition of the RMan extends the chip's IO virtualization by allowing many types of traffic arriving on Serial RapidIO to enter the DPAA and take advantage of its inherent virtualization and partitioning capabilities.

The PCI Express protocol lacks the PDU semantics found in Serial RapidIO, making it difficult to interwork between PCI Express controllers and the DPAA; however, PCI Express has made progress in other areas of partition. The Single Root IO Virtualization specification, which the chip supports as an endpoint, allows external hosts to view the chip as multiple four physical functions (PFs), where each PF supports up to 64 virtual functions (VFs). Having multiple VFs on a PCI Express port effectively channelizes it, so that each transaction through the port is identified as belonging to a specific PF/VF combination (with associated and potentially dedicated memory regions). Message signalled interrupts (MSIs) allow the external Host to generate interrupts associated with a specific VF.

1.3.12.4 Secure boot and sensitive data protection

The core MMUs and PAMU allow the SoC to enforce a consistent set of memory access permissions on a per-partition basis. When combined with an embedded hypervisor for safe sharing of resources, the SoC becomes highly resilient to poorly tested or malicious code. For system developers building high reliability/high security platforms, rigorous testing of code of known origin is the norm.

For this reason, the SoC offers a secure boot option, in which the system developer digitally signs the code to be executed by the CPUs, and the SoC insures that only an unaltered version of that code runs on the platform. The SoC offers both boot time and run time code authenticity checking, with configurable consequences when the authenticity check fails. The SoC also supports protected internal and external storage of developer-provisioned sensitive instructions and data. For example, a system developer may provision each system with a number of RSA private keys to be used in mutual authentication and key exchange. These values would initially be stored as encrypted blobs in external non-volatile memory; but, following secure boot, these values can be decrypted into on-chip protected memory (portion of platform cache dedicated as SRAM). Session keys, which may number in the thousands to tens of thousands, are not good candidates for on-chip storage, so the SoC offers session key encryption. Session keys are stored in main memory, and are decrypted (transparently to software and without impacting SEC throughput) as they are brought into the for decryption of session traffic.

1.3.13 Advanced power management

Power dissipation is always a major design consideration in embedded applications; system designers need to balance the desire for maximum compute and IO density against single-chip and board-level thermal limits.

The chip implements a number of software transparent and performance transparent power management features. Non-transparent power management features are also available, allowing for significant reductions in power consumption when the chip is under lighter loads; however, non-transparent power savings are not assumed in chip power specifications.

1.3.13.1 Transparent power management

This chip's commitment to low power begins with the decision to fabricate the chip in 28 nm bulk CMOS. This process technology offers low leakage, reducing both static and dynamic power. While 28 nm offers inherent power savings, transistor leakage varies from lot to lot and device to device. Leakier parts are capable of faster transistor switching, but they also consume more power. By running devices from the leakier end of the process spectrum at less than nominal voltage and devices from the slower end of the process spectrum at higher nominal voltage, T2080-based systems can achieve the required operating frequency within the specified max power. During manufacturing, Freescale will determine the voltage required to achieve the target frequency bin and program this Voltage ID into each device, so that initialization software can program the system's voltage regulator to the appropriate value.

Dynamic power is further reduced through fine-grained clock control. Many components and subcomponents in the chip automatically sleep (turn off their clocks) when they are not actively processing data. Such blocks can return to full operating frequency on the clock cycle after work is dispatched to them. A portion of these dynamic power savings are built into the chip max power specification on the basis of impossibility of all processing elements and interfaces in the chip switching concurrently. The percent switching factors are considered quite conservative, and measured typical power consumption on QorIQ chips is well below the maximum in the data sheet.

As noted in Frame Manager and network interfaces, the chip supports Energy-Efficient Ethernet. During periods of extended inactivity on the transmit side, the chip transparently sends a low power idle (LPI) signal to the external PHY, effectively telling it to sleep.

Additional power savings can be achieved by users statically disabling unused components. Developers can turn off the clocks to individual logic blocks (including CPUs) within the chip that the system is not using. Based on a finite number of SerDes, it is expected that any given application will have some inactive Ethernet MACs, PCI Express, or serial RapidIO controllers. Re-enabling clocks to a logic block generally requires an chip reset, which makes this type of power management infrequent (effectively static) and transparent to runtime software.

1.3.13.2 Non-transparent power management

Many load-based power savings are use-case specific static configurations (thereby software transparent), and were described in the previous section. This section focuses on SoC power management mechanisms, which software can dynamically leverage to reduce power when the system is lightly loaded. The most important of these mechanisms involves the cores.

A full description of core low-power states with proper names is provided in the chip reference manual. At a high level, the most important of these states can be viewed as PH10, PH15, and PH20 described as follows:

- A PH10 CPU stops instruction fetches but still performs L1 snoops. The CPU retains all state, and instruction fetching can be restarted instantly.
- A PH15 CPU stops instruction fetches and L1 snooping, and turns off all clocks. In the PH15 state, a CPU can still return to full operation quickly (~100 platform clocks).
- A PH20 CPU is the same as PH15 plus partial-power gating.
- A PH30 CPU is same as PH20 plus full power gating

The core offers several ways to enter these (and other) low power states: registers and instructions and events.

As the name implies, register-based power management means that software writes to registers to select the CPU and its low power state. Any CPU with write access to power management registers can put itself, or another CPU, into a low power state; however, a CPU put into a low power state by way of register write cannot wake itself up.

Instruction-based power management means that software executes special WAIT instructions to enter a low power state. CPUs exit the low power state in response to external triggers, interrupts, doorbells, stashes into L1-D cache, or clear reservation on snoop. Each vCPU can independently execute WAIT instructions; however, the physical CPU only enters a specific power mode after the second vCPU executes its weight. The instruction-based power mode state is particularly well-suited for use in conjunction with Freescale's patented Cascade Power Management, which is described in the next section.

Event-based power management means that upon specific events such as a debug event, a device pin event, and so on, can be programmed to enter one or more cores into a specific power mode.

While significant power savings can be achieved through individual CPU low power states, the chip also supports a register-based cluster level low power state. After software puts all CPUs in a cluster in a PH20 state, it can additionally flush the L2 cache and have the entire cluster enter PCL10 mode. Because the L2 arrays have relatively low static power dissipation, this state provides incremental additional savings over having four PH20 CPUs with the L2 on.

1.3.13.3 Cascade power management

CPU cores can choose during run-time to dequeue from a dedicated channel (one per portal) or pool channel(s) (shared by multiple CPU cores). Within a pool channel, the QMan distributes packets to CPU cores using either a round robin or priority scheduling scheme (configurable). The priority scheduling scheme is used to support cascade power management. When priority scheduling is configured on a pool channel, the QMan distributes packets to the lower numbered portal within the pool channel until its DQRR becomes full. Once full, it distributes packets to next lower numbered portals until its DQRR becomes full, and so on.

Software is responsible for putting CPU cores into their drowsy mode. A CPU core is placed into its drowsy mode by having software execute a WAIT instruction. Software detects that a core is ready to be placed into its drowsy mode after a timeout of its DQRR being emptied.

1.3.14 Debug support

The reduced number of external buses enabled by the move to multicore chips greatly simplifies board level lay-out and eliminates many concerns over signal integrity. While the board designer may embrace multicore CPUs, software engineers have real concerns over the potential to lose debug visibility.

Processing on a multicore SoC with shared caches and peripherals also leads to greater concurrency and an increased potential for unintended interactions between device components. To ensure that software developers have the same or better visibility into the device as they would with multiple discrete communications processors, Freescale developed an Advanced Multicore Debug Architecture.

The debugging and performance monitoring capability enabled by the device hardware coexists within a debug ecosystem that offers a rich variety of tools at different levels of the hardware/software stack. Software development and debug tools from Freescale (Codewarrior), as well as third-party vendors, provide a rich set of options for configuring, controlling, and analyzing debug and performance related events.

1.3.15 JTAG

The JTAG Test Access Port (TAP) and Boundary Scan Architecture is designed to comply with IEEE Std. 1149.1 and 1149.6.

0x018E_701D

Chapter 2
Memory Map

2.1 Memory Map Overview

This chapter describes the mechanisms that define the device memory map-the local access windows (LAWs), the address translation and mapping units (ATMUs), and the configuration, control, and status registers (CCSRs).

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

- Logical, virtual, and physical (real) address spaces within the Power Architecture core(s)
- Internal local address space
  - Internal configuration, control, and status register (CCSR) address space, which is a special-purpose subset of the internal local address space
  - 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 serial RapidIO link
- External memory, I/O, and configuration address spaces of the PCI Express links

The MMU in the core handles translation of logical (effective) addresses, into virtual addresses, and ultimately to the physical addresses for the local address space. The MMU is described in the *e6500 Core Reference Manual*.

The local address map refers to the physical address space seen by the core as it accesses memory and I/O space. The DMA engines also see this same local address map. All memory controlled by the DDR and other modules exists in this address map, as do all memory-mapped configuration, control, and status registers (CCSRs). The local address map is defined by a set of 32 local access windows (LAWs). Each of these windows maps a region of the local address space to a specified target interface, such as the DDR controller, PCI Express controller, or other targets. The internal configuration, control,

and status registers (CCSRs) for all the functional blocks are located in the local memory space at a specific CCSR window. There is also a fixed (8 Mbytes of local memory space) default boot window from 00_FF80_0000h to 00_FFFF_FFFFh.

The internal debug control and status registers (DCSRs) control the hardware debug and performance monitoring capabilities of many functional blocks. The DCSRs are mapped into the local memory space by specifying the DCSR space as a LAW target.

If the target mapping performed by the local access windows directs the transaction to one of the external peripheral interfaces (as an outbound read or write), the transaction is then mapped into that interface's external address space by the address translation and mapping unit (ATMU) windows associated with the external interface. Outbound ATMUs perform the mapping from the local address space to the address space of the external peripheral interface; inbound ATMU windows perform the address translation from the external address space to the local address space.

2.2 Global Source and Target IDs

In many instances throughout the system, certain register or packet fields need to indicate or specify a transaction source or target. Table 2-1 provides the encodings used for these fields.

Table 2-1. T2080 Global Source/Target ID Assignments

Source/Target ID Value	Transaction source	Transaction target	Notes
00h	PCI Express 1	PCI Express 1	-
01h	PCI Express 2	PCI Express 2	-
02h	PCI Express 3	PCI Express 3	-
03h	PCI Express 4	PCI Express 4	-
...	Reserved	Reserved	
08h	Rapid I/O port 1	Rapid I/O port 1	-
09h	Rapid I/O port 2	Rapid I/O port 2	-
10h	CPC castout/flush	Memory Complex	Includes targeting segments of CPC configured as SRAM
11h	CPC M-INT writes	Memory Complex	
...	Reserved	Reserved	-
14h	Prefetch manager	Memory Complex	All PMANs use the same Source ID when making memory accesses.
17h	Reserved		
18h	Buffer Manager (control)	Buffer Manager Software Portal	-
...	Reserved	Reserved	-

Table continues on the next page...

Table 2-1. T2080 Global Source/Target ID Assignments (continued)

Source/Target ID Value	Transaction source	Transaction target	Notes
1Ch	PAMU	Reserved	All PAMUs use the same source ID when performing memory accesses.
1Dh	Reserved	DCSR	The debug facilities are accessed through a DCSR mapped LAW. Whenever a Local Access Window targets DCSR space, the size of this space must always be set to 4 Mbytes.
1Eh	Reserved	CCSR	-
1Fh	Reserved	Integrated Flash controller	-
...	Reserved	Reserved	
20h	Pattern matching engine	Reserved	-
21h	Security engine (SEC)	Reserved	
22h	Data compression engine	Reserved	-
...	Reserved	Reserved	
30h	Cluster 1 L2 cache castout/flush	Reserved	
31h	Reserved	Reserved	
32h	Reserved	Reserved	
...	Reserved	Reserved	-
3Ch	Queue Manager (control)	Queue Manager Software Portal	-
...	Reserved	Reserved	-
40h	USB 1	Reserved	-
41h	USB 2	Reserved	-
...	-	-	-
44h	eSDHC	Reserved	-
45h	Reserved		
...	Reserved	Reserved	-
48h	Pre-boot loader	Reserved	-
...	Reserved	Reserved	
4Bh	Nexus port controller	Reserved	-
...	Reserved	Reserved	
5Dh	Rapid I/O message manager		
...	Reserved	Reserved	
60h	SATA 1		
61h	SATA 2		
...	Reserved	Reserved	
70h	DMA 1	Reserved	-
71h	DMA 2	Reserved	-
72h	DMA 3	Reserved	-

Table continues on the next page...

Table 2-1. T2080 Global Source/Target ID Assignments (continued)

Source/Target ID Value	Transaction source	Transaction target	Notes
...	Reserved	Reserved	-
80h	Thread 0 (instruction)	Reserved	Physical core 0
81h	Thread 0 (data)	Reserved	Physical core 0
82h	Thread 1 (instruction)	Reserved	Physical core 0
83h	Thread 1 (data)	Reserved	Physical core 0
84h	Thread 2 (instruction)	Reserved	Physical core 1
85h	Thread 2 (data)	Reserved	Physical core 1
86h	Thread 3 (instruction)	Reserved	Physical core 1
87h	Thread 3 (data)	Reserved	Physical core 1
88h	Thread 4 (instruction)	Reserved	Physical core 2
89h	Thread 4 (data)	Reserved	Physical core 2
8Ah	Thread 5 (instruction)	Reserved	Physical core 2
8Bh	Thread 5 (data)	Reserved	Physical core 2
8Ch	Thread 6 (instruction)	Reserved	Physical core 3
8Dh	Thread 6 (data)	Reserved	Physical core 3
8Eh	Thread 7 (instruction)	Reserved	Physical core 3
8Fh	Thread 7 (data)	Reserved	Physical core 3
...	Reserved	Reserved	-
C0h	Frame Manager XAUI 1	Reserved	-
C1h	Frame Manager XAUI 2	Reserved	-
C2h	Frame Manager offline/host 2	Reserved	-
C3h	Frame Manager offline/host 3	Reserved	-
C4h	Frame Manager offline/host 4	Reserved	-
C5h	Frame Manager offline/host 5	Reserved	-
C6h	Frame Manager offline/host 6	Reserved	-
C7h	Frame Manager offline/host 7	Reserved	-
C8h	Frame Manager GE 1	Reserved	-
C9h	Frame Manager GE 2	Reserved	-
CAh	Frame Manager GE 3	Reserved	-
CBh	Frame Manager GE 4	Reserved	-
CCh	Frame Manager GE 5	Reserved	-
CDh	Frame Manager GE 6	Reserved	-

2.3 Local Access Windows (LAWs)

The local address map is defined by a set of 32 local access windows (LAWs).

Each of these windows maps a programmable region of the local address space to a specified target interface, such as the DDR controller, 512 KB platform SRAM, or other targets. This allows the internal interconnections of the device to route a transaction from its source to the proper target.

Each LAW is defined by a pair of base address registers that specify the starting address for the window, and an attribute register that specifies whether the mapping is enabled, the size of the window, a coherency subdomain, and the target interface for that window. Note that the LAWs do not perform any address translation, and therefore there are no corresponding translation address registers. The local access window registers exist as part of the local access control block in the CCSR space.

With the exception of configuration space (mapped by CCSRBAR) and the boot window, all addresses used by the system must be mapped by a LAW. This includes addresses that are mapped by inbound ATMU windows. Thus, target mappings of the LAWs and the inbound ATMU windows must be consistent.

2.3.1 Precedence of Local Access Windows

If two or more LAWs overlap, the lower-numbered window takes precedence.

For example, consider two LAWs, set up as shown in Table 2-2.

Table 2-2. Overlapping Local Access Windows

LAW	Base Address	Size	Target Interface
1	00_7FF0_0000h	1 Mbyte	Integrated Flash controller (IFC)
2	00_0000_0000h	2 Gbytes	Memory complex (DDR controller)

In this case, LAW 1 governs the mapping of the 1-Mbyte region from 00_7FF0_0000h to 00_7FFF_FFFFh, even though the window described in LAW 2 also encompasses that memory region.

NOTE

The CCSR mapping, defined by the CCSRBARs, supersedes all local access window mappings.

NOTE

The boot window is another special area. In unsecured boot, the boot window has lower priority than the LAWs. However, secure boot, will preempt accesses to the boot window and send them to the internal secure boot code (ISBC). In this case, the

secure boot effectively makes the boot window higher priority than the LAWs.

2.3.2 Configuring Local Access Windows

Once a local access window is enabled, it should not be modified while any device in the system may be using the window.

Neither should a new window be used until the effect of the write to the window is visible to all blocks that use the window. This can be guaranteed by completing a read of the last LAW configuration register before enabling any other devices to use the window. For example, if LAWs 0-3 are being configured in order during the initialization process, the last write (to LAWAR3) should be followed by a read of LAWAR3 before any devices try to use any of these windows. If the configuration is being performed by the core, the read of LAWAR3 should be followed by an **isync** instruction.

2.3.3 Distinguishing Local Access Windows from Other Mapping Functions

It is important to distinguish between the mapping function performed by the LAWs and the additional mapping functions that occur at the target interfaces.

The LAWs define how a transaction is routed through the device's internal interconnects from the transaction's source to its target. After the transaction has arrived at its target interface, that interface controller may perform additional mapping. For instance, the DDR controller has chip select registers that map a memory request to a particular external device. Similarly, the integrated flash controller has base registers that perform a similar function. The external peripheral interface controllers (for example, serial RapidIO and PCI Express) have outbound address translation and mapping units (ATMUs) that map the local address into an external address space.

These other mapping functions are configured by programming the CCSRs of the individual interfaces. Note that there is no need to have a one-to-one correspondence between LAWs and chip select regions or outbound ATMU windows. A single LAW can be further decoded to any number of chip selects or to any number of outbound ATMU windows at the target interface.

2.3.4 SRAM Windows

The CoreNet platform cache (CPC) can be configured as a memory-mapped SRAM. The CPC SRAM control registers (CPCSRCRs) in the CPC cache controller set the base addresses and sizes for these windows. See SRAM Mode Registers for information about configuring SRAM windows.

In addition to defining the SRAM regions in the CPCSRCRs, the CPC SRAM regions need to be covered by LAWs using their associated memory complex as the target ID (TRGT_ID).

2.3.5 Local Address Map Example

The figure below shows what a typical local address map might look like.

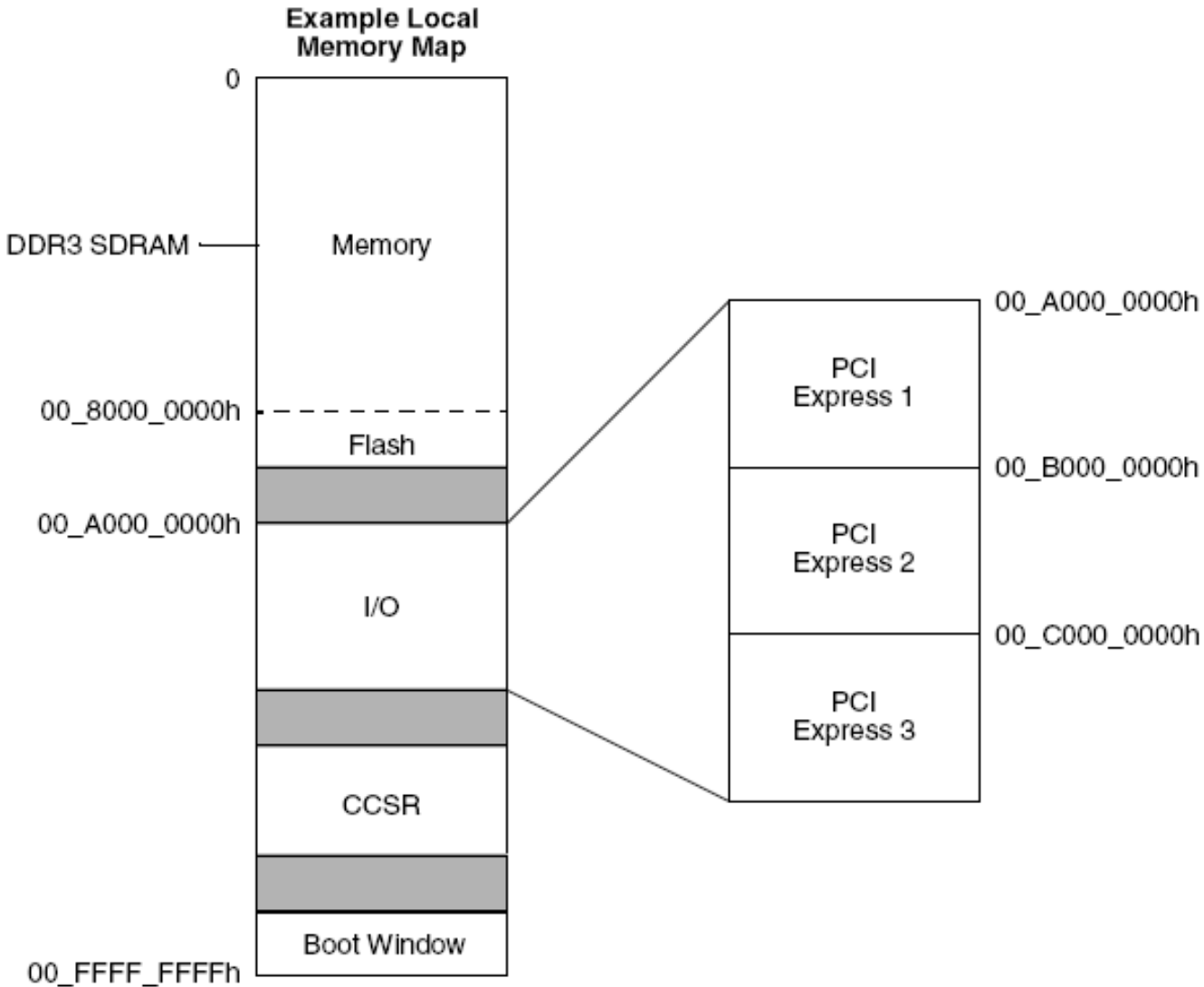


Figure 2-1. Local Address Map Example

The table shows the corresponding set of LAW settings for the example shown in the figure.

Table 2-3. Local Access Window Settings Example

Window	Base Address	SIZE	Target Interface
0	00_0000_0000h	2 Gbytes	DDR
1	00_8000_0000h	1 Mbyte	Integrated Flash controller (IFC) - Flash
2	00_A000_0000h	256 Mbytes	PCI Express 1
3	00_B000_0000h	256 Mbytes	PCI Express 2
4	00_C000_0000h	256 Mbytes	PCI Express 3
5-9	Unused		

In this example, note that it is not required to define a LAW to describe the range of memory used for memory-mapped configuration, control, and status registers because these occupy a fixed 16-Mbyte space pointed to by the CCSRBARs. Also note that the boot window is always enabled and covers local addresses from 00_FF80_0000h to 00_FFFF_FFFFh.

2.4 Local Access Window (LAW) Memory Map

The table below shows the memory map for the LAW registers. The local access window registers are accessed by reading and writing to an address comprised of the base address (specified in the CCSRBAR), plus the block base address, plus the offset of the specific register to be accessed. For the LAWs, the block base address is local access control block at 0x00_0000.

Note that all LAW registers should only be accessed a word (4 bytes) at a time.

LAW memory map

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
C00	LAWn base address register high (LAW_LAWBARH0)	32	R/W	0000_0000h	2.4.1/114
C04	LAWn base address register low (LAW_LAWBARL0)	32	R/W	0000_0000h	2.4.2/114
C08	LAWn attribute register (LAW_LAWAR0)	32	R/W	0000_0000h	2.4.3/115
C10	LAWn base address register high (LAW_LAWBARH1)	32	R/W	0000_0000h	2.4.1/114
C14	LAWn base address register low (LAW_LAWBARL1)	32	R/W	0000_0000h	2.4.2/114
C18	LAWn attribute register (LAW_LAWAR1)	32	R/W	0000_0000h	2.4.3/115
C20	LAWn base address register high (LAW_LAWBARH2)	32	R/W	0000_0000h	2.4.1/114
C24	LAWn base address register low (LAW_LAWBARL2)	32	R/W	0000_0000h	2.4.2/114
C28	LAWn attribute register (LAW_LAWAR2)	32	R/W	0000_0000h	2.4.3/115
C30	LAWn base address register high (LAW_LAWBARH3)	32	R/W	0000_0000h	2.4.1/114
C34	LAWn base address register low (LAW_LAWBARL3)	32	R/W	0000_0000h	2.4.2/114
C38	LAWn attribute register (LAW_LAWAR3)	32	R/W	0000_0000h	2.4.3/115
C40	LAWn base address register high (LAW_LAWBARH4)	32	R/W	0000_0000h	2.4.1/114
C44	LAWn base address register low (LAW_LAWBARL4)	32	R/W	0000_0000h	2.4.2/114
C48	LAWn attribute register (LAW_LAWAR4)	32	R/W	0000_0000h	2.4.3/115
C50	LAWn base address register high (LAW_LAWBARH5)	32	R/W	0000_0000h	2.4.1/114
C54	LAWn base address register low (LAW_LAWBARL5)	32	R/W	0000_0000h	2.4.2/114
C58	LAWn attribute register (LAW_LAWAR5)	32	R/W	0000_0000h	2.4.3/115
C60	LAWn base address register high (LAW_LAWBARH6)	32	R/W	0000_0000h	2.4.1/114

Table continues on the next page...

LAW memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
C64	LAWn base address register low (LAW_LAWBARL6)	32	R/W	0000_0000h	2.4.2/114
C68	LAWn attribute register (LAW_LAWAR6)	32	R/W	0000_0000h	2.4.3/115
C70	LAWn base address register high (LAW_LAWBARH7)	32	R/W	0000_0000h	2.4.1/114
C74	LAWn base address register low (LAW_LAWBARL7)	32	R/W	0000_0000h	2.4.2/114
C78	LAWn attribute register (LAW_LAWAR7)	32	R/W	0000_0000h	2.4.3/115
C80	LAWn base address register high (LAW_LAWBARH8)	32	R/W	0000_0000h	2.4.1/114
C84	LAWn base address register low (LAW_LAWBARL8)	32	R/W	0000_0000h	2.4.2/114
C88	LAWn attribute register (LAW_LAWAR8)	32	R/W	0000_0000h	2.4.3/115
C90	LAWn base address register high (LAW_LAWBARH9)	32	R/W	0000_0000h	2.4.1/114
C94	LAWn base address register low (LAW_LAWBARL9)	32	R/W	0000_0000h	2.4.2/114
C98	LAWn attribute register (LAW_LAWAR9)	32	R/W	0000_0000h	2.4.3/115
CA0	LAWn base address register high (LAW_LAWBARH10)	32	R/W	0000_0000h	2.4.1/114
CA4	LAWn base address register low (LAW_LAWBARL10)	32	R/W	0000_0000h	2.4.2/114
CA8	LAWn attribute register (LAW_LAWAR10)	32	R/W	0000_0000h	2.4.3/115
CB0	LAWn base address register high (LAW_LAWBARH11)	32	R/W	0000_0000h	2.4.1/114
CB4	LAWn base address register low (LAW_LAWBARL11)	32	R/W	0000_0000h	2.4.2/114
CB8	LAWn attribute register (LAW_LAWAR11)	32	R/W	0000_0000h	2.4.3/115
CC0	LAWn base address register high (LAW_LAWBARH12)	32	R/W	0000_0000h	2.4.1/114
CC4	LAWn base address register low (LAW_LAWBARL12)	32	R/W	0000_0000h	2.4.2/114
CC8	LAWn attribute register (LAW_LAWAR12)	32	R/W	0000_0000h	2.4.3/115
CD0	LAWn base address register high (LAW_LAWBARH13)	32	R/W	0000_0000h	2.4.1/114
CD4	LAWn base address register low (LAW_LAWBARL13)	32	R/W	0000_0000h	2.4.2/114
CD8	LAWn attribute register (LAW_LAWAR13)	32	R/W	0000_0000h	2.4.3/115
CE0	LAWn base address register high (LAW_LAWBARH14)	32	R/W	0000_0000h	2.4.1/114
CE4	LAWn base address register low (LAW_LAWBARL14)	32	R/W	0000_0000h	2.4.2/114
CE8	LAWn attribute register (LAW_LAWAR14)	32	R/W	0000_0000h	2.4.3/115
CF0	LAWn base address register high (LAW_LAWBARH15)	32	R/W	0000_0000h	2.4.1/114
CF4	LAWn base address register low (LAW_LAWBARL15)	32	R/W	0000_0000h	2.4.2/114
CF8	LAWn attribute register (LAW_LAWAR15)	32	R/W	0000_0000h	2.4.3/115
D00	LAWn base address register high (LAW_LAWBARH16)	32	R/W	0000_0000h	2.4.1/114
D04	LAWn base address register low (LAW_LAWBARL16)	32	R/W	0000_0000h	2.4.2/114
D08	LAWn attribute register (LAW_LAWAR16)	32	R/W	0000_0000h	2.4.3/115
D10	LAWn base address register high (LAW_LAWBARH17)	32	R/W	0000_0000h	2.4.1/114
D14	LAWn base address register low (LAW_LAWBARL17)	32	R/W	0000_0000h	2.4.2/114
D18	LAWn attribute register (LAW_LAWAR17)	32	R/W	0000_0000h	2.4.3/115
D20	LAWn base address register high (LAW_LAWBARH18)	32	R/W	0000_0000h	2.4.1/114
D24	LAWn base address register low (LAW_LAWBARL18)	32	R/W	0000_0000h	2.4.2/114
D28	LAWn attribute register (LAW_LAWAR18)	32	R/W	0000_0000h	2.4.3/115

Table continues on the next page...

LAW memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
D30	LAWn base address register high (LAW_LAWBARH19)	32	R/W	0000_0000h	2.4.1/114
D34	LAWn base address register low (LAW_LAWBARL19)	32	R/W	0000_0000h	2.4.2/114
D38	LAWn attribute register (LAW_LAWAR19)	32	R/W	0000_0000h	2.4.3/115
D40	LAWn base address register high (LAW_LAWBARH20)	32	R/W	0000_0000h	2.4.1/114
D44	LAWn base address register low (LAW_LAWBARL20)	32	R/W	0000_0000h	2.4.2/114
D48	LAWn attribute register (LAW_LAWAR20)	32	R/W	0000_0000h	2.4.3/115
D50	LAWn base address register high (LAW_LAWBARH21)	32	R/W	0000_0000h	2.4.1/114
D54	LAWn base address register low (LAW_LAWBARL21)	32	R/W	0000_0000h	2.4.2/114
D58	LAWn attribute register (LAW_LAWAR21)	32	R/W	0000_0000h	2.4.3/115
D60	LAWn base address register high (LAW_LAWBARH22)	32	R/W	0000_0000h	2.4.1/114
D64	LAWn base address register low (LAW_LAWBARL22)	32	R/W	0000_0000h	2.4.2/114
D68	LAWn attribute register (LAW_LAWAR22)	32	R/W	0000_0000h	2.4.3/115
D70	LAWn base address register high (LAW_LAWBARH23)	32	R/W	0000_0000h	2.4.1/114
D74	LAWn base address register low (LAW_LAWBARL23)	32	R/W	0000_0000h	2.4.2/114
D78	LAWn attribute register (LAW_LAWAR23)	32	R/W	0000_0000h	2.4.3/115
D80	LAWn base address register high (LAW_LAWBARH24)	32	R/W	0000_0000h	2.4.1/114
D84	LAWn base address register low (LAW_LAWBARL24)	32	R/W	0000_0000h	2.4.2/114
D88	LAWn attribute register (LAW_LAWAR24)	32	R/W	0000_0000h	2.4.3/115
D90	LAWn base address register high (LAW_LAWBARH25)	32	R/W	0000_0000h	2.4.1/114
D94	LAWn base address register low (LAW_LAWBARL25)	32	R/W	0000_0000h	2.4.2/114
D98	LAWn attribute register (LAW_LAWAR25)	32	R/W	0000_0000h	2.4.3/115
DA0	LAWn base address register high (LAW_LAWBARH26)	32	R/W	0000_0000h	2.4.1/114
DA4	LAWn base address register low (LAW_LAWBARL26)	32	R/W	0000_0000h	2.4.2/114
DA8	LAWn attribute register (LAW_LAWAR26)	32	R/W	0000_0000h	2.4.3/115
DB0	LAWn base address register high (LAW_LAWBARH27)	32	R/W	0000_0000h	2.4.1/114
DB4	LAWn base address register low (LAW_LAWBARL27)	32	R/W	0000_0000h	2.4.2/114
DB8	LAWn attribute register (LAW_LAWAR27)	32	R/W	0000_0000h	2.4.3/115
DC0	LAWn base address register high (LAW_LAWBARH28)	32	R/W	0000_0000h	2.4.1/114
DC4	LAWn base address register low (LAW_LAWBARL28)	32	R/W	0000_0000h	2.4.2/114
DC8	LAWn attribute register (LAW_LAWAR28)	32	R/W	0000_0000h	2.4.3/115
DD0	LAWn base address register high (LAW_LAWBARH29)	32	R/W	0000_0000h	2.4.1/114
DD4	LAWn base address register low (LAW_LAWBARL29)	32	R/W	0000_0000h	2.4.2/114
DD8	LAWn attribute register (LAW_LAWAR29)	32	R/W	0000_0000h	2.4.3/115
DE0	LAWn base address register high (LAW_LAWBARH30)	32	R/W	0000_0000h	2.4.1/114
DE4	LAWn base address register low (LAW_LAWBARL30)	32	R/W	0000_0000h	2.4.2/114
DE8	LAWn attribute register (LAW_LAWAR30)	32	R/W	0000_0000h	2.4.3/115
DF0	LAWn base address register high (LAW_LAWBARH31)	32	R/W	0000_0000h	2.4.1/114
DF4	LAWn base address register low (LAW_LAWBARL31)	32	R/W	0000_0000h	2.4.2/114

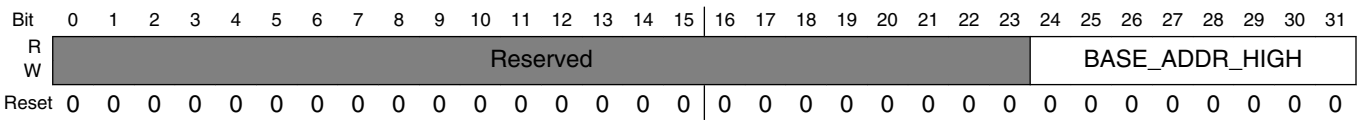
Table continues on the next page...

LAW memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
DF8	LAWn attribute register (LAW_LAWAR31)	32	R/W	0000_0000h	2.4.3/115

2.4.1 LAWn base address register high (LAW_LAWBARH*n*)

Address: 0h base + C00h offset + (16d × i), where i=0d to 31d

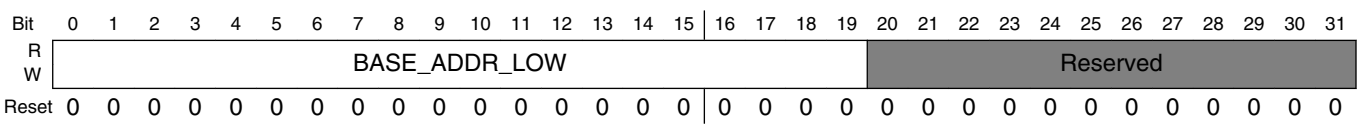


LAW_LAWBARH*n* field descriptions

Field	Description
0–23 -	This field is reserved. Reserved
24–31 BASE_ADDR_ HIGH	Identifies bits 0-7 of the 40-bit base address of local access window*n*. The specified base address should be aligned to the window size, as defined by LAWAR*n*[SIZE].

2.4.2 LAWn base address register low (LAW_LAWBARL*n*)

Address: 0h base + C04h offset + (16d × i), where i=0d to 31d



LAW_LAWBARL*n* field descriptions

Field	Description
0–19 BASE_ADDR_ LOW	Identifies bits 8-27 of the 40-bit base address of local access window *n* . The specified base address should be aligned to the window size, as defined by LAWAR *n* [SIZE].
20–31 -	This field is reserved. Write reserved, read = 0. Because the minimum window size is 4 Kbytes, the 12 least significant bits are treated as zeros.

2.4.3 LAWn attribute register (LAW_LAWAR*n*)

Address: 0h base + C08h offset + (16d × i), where i=0d to 31d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	EN	Reserved			TRGT_ID								CSD_ID			
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	CSD_ID				Reserved						SIZE					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LAW_LAWAR*n* field descriptions

Field	Description
0 EN	Enable 0 The local access window *n* (and all other LAWAR *n* and LAWBAR *n* fields) are disabled. 1 The local access window *n* is enabled and other LAWAR *n* and LAWBAR *n* fields combine to identify an address range for this window.
1–3 -	This field is reserved. Write reserved, read = 0
4–11 TRGT_ID	Target identifier. Identifies the target interface when a transaction hits in the address range defined by this window. See Global Source and Target IDs for the defined encodings. **NOTE:** CPC SRAM regions are generally mapped by LAWs using their associated memory complex target ID. However, these regions also need to be defined by the CPC SRAM control registers (CPCSRCRs) in the CPC block. **NOTE:** Configuration registers are mapped by the windows defined by the CCSRBARs. These mappings supersede local access window mappings, so configuration registers do not appear as a target for local access windows. **NOTE:** Whenever a Local Access Window targets DCSR space, the size of this space must always be set to 4 Mbytes. **NOTE:** TRGT_ID = 0x0F (local space) is an illegal encoding in the LAWARn registers and causes a local access error (CEDR[LAE]) if detected.
12–19 CSD_ID	Coherency subdomain identifier. Identifies a group of one or more partitions that have access to the address space defined by this access window. Each such group may contain one or more processors and/or other caching devices representing a coherency subdomain within the system. For a coherent access via this window, coherency is enforced over this subdomain.
20–25 -	This field is reserved. Write reserved, read = 0
26–31 SIZE	Identifies the size of the window from the starting address. Window size is 2 $^{(SIZE+1)}$ bytes. Example values are as follows: 000000-001010 Reserved 001011 4 Kbytes 001100 8 Kbytes 100010 32 Gbytes 100011 64 Gbytes

Table continues on the next page...

LAW_LAWAR*n* field descriptions (continued)

Field	Description
	...
	100111 1 Tbyte
	101000-111111 Reserved

2.5 Address Translation and Mapping Units

.To facilitate flexibility in defining the address maps for the external interfaces (serial RapidIO and PCI Express), the device provides address translation and mapping units (ATMUs).

The following types of translation and mapping operations are performed by the ATMUs:

- Translating the local address to an external address space
- Translating external addresses to the local address space
- Assigning attributes to transactions
- Mapping a local address to a target interface

Outbound address translation and mapping refers to the translation of addresses from the local address space to the external address space and attributes of a particular I/O interface.

Inbound address translation and mapping refers to the translation of an address from the external address space of an I/O interface to the local address space understood by the internal interfaces of this device. It also refers to the mapping of transactions to a particular target interface and the assignment of transaction attributes. Note that in mapping the transaction to the target interface, an inbound ATMU window performs a function similar to that of the local access windows. The target mappings created by an inbound ATMU must be consistent with those of the LAWs. That is, if an inbound ATMU maps a transaction to a given local address and a given target, a LAW must also map that same local address to the same target.

2.5.1 Address Translation

All of the configuration registers that define an ATMU window's translation and mapping functions follow the same general register format.

The register format is summarized in Table 2-103.

Table 2-103. Format of ATMU Window Definitions

Register	Function
Translation address (TAR/TEAR)	High-order address bits defining location of the window in the target address space
Base address (BAR/BEAR)	High-order address bits defining location of the window in the initiator address space
Window attributes (WAR)	Window enable, window size, target interface, and transaction attributes

The size of the windows must be a power-of-two. To perform a translation or mapping function, the address of the transaction is compared with the base address register of each window. The number of bits used in the comparison is dictated by each window's size attribute. When an address hits a window, if address translation is being performed, the new translated address is created by concatenating the window offset to the translation address. Again, the window's size attribute dictates how many bits are translated.

2.5.2 Outbound ATMUs

If the target mapping performed by the local access windows directs the transaction to one of the external peripheral interfaces (as an outbound read or write), the transaction is then mapped into that interface's external address space by outbound ATMUs associated with the external interface.

The outbound ATMUs perform the mapping from the local address space to the address space of the external peripheral interface, which may be much larger than the local space.

The outbound ATMUs also map attributes such as the transaction type and priority level.

The serial RapidIO controller has eight outbound ATMU windows plus a default window. Each PCI Express controller has four outbound ATMU windows plus a default window. If a transaction's address does not hit any of the outbound ATMU windows, the translation actions defined by the default window are used. The default window is always enabled. The outbound ATMU registers include extended translation address registers so that up to 64 bits of external address space can be supported.

2.5.3 Inbound ATMUs

The inbound ATMUs perform the address translation from the external address spaces to the local address space, attach attributes and transaction types to the transaction, and also map the transaction to its target interface.

The serial RapidIO controller has four inbound ATMU windows plus a default window. If the inbound transaction's address does not hit any of the inbound ATMU windows, the translation actions defined by the default window are used.

Each PCI Express controller has four inbound ATMU windows plus a fixed MSI ATMU window for mapping inbound message-signalled interrupt transactions to the interrupt controller (MPIC).

2.5.3.1 Illegal Interaction Between Inbound ATMUs and LAWs

Since both local access windows and inbound ATMUs map transactions to a target interface, it is essential that they not contradict one another.

For example, it is considered a programming error to have an inbound ATMU map a transaction target to the local memory space if the resulting translated local address is mapped to an external peripheral interface by a local access window. Such programming errors may result in unpredictable system deadlocks.

2.6 Configuration, Control, and Status Register (CCSR) Space

To allow for flexibility, the configuration, control, and status register space is relocatable in the local address space.

The local address map location of this register block is controlled by the configuration, control, and status registers base address register (CCSRBAR), see Accessing Configuration, Control, and Status Registers.

The default value for CCSRBAR is 0.

2.6.1 Accessing CCSR Memory from the Local Processor

When the local processor cores are used to configure CCSR space, the CCSR memory space should typically be marked as cache-inhibited and guarded.

In addition, many configuration registers affect accesses to other memory regions; therefore writes to these registers must be guaranteed to have taken effect before accesses are made to the associated memory regions.

To guarantee that the results of any sequence of writes to configuration registers are in effect, the final configuration register write should be immediately followed by a read of the same register, and that should be followed by a SYNC instruction. Then accesses can safely be made to memory regions affected by the configuration register write.

2.6.2 Accessing CCSR Memory from External Masters

In addition to being accessible by the processor cores, the configuration, control, and status registers are accessible from external interfaces.

This allows external masters on the I/O ports to configure the device.

External masters do not need to know the location of the CCSR memory in the local address map. Rather, they access this region of the local memory map through a window defined by a register in the interface programming model that is accessible to the external master from its external memory map.

The PCI Express controller's base address for accessing the local CCSR memory is selectable through the PCI Express configuration and status register base address register (PEXCSRBAR), at offset 0x10. An external PCI Express root complex host sets this register by running a PCI Express configuration transaction. Subsequent memory accesses by a PCI Express master to the PCI Express address range indicated by PEXCSRBAR are translated to the local address indicated by the current setting of CCSRBAR.

The serial RapidIO base address for accessing the local CCSR memory is selectable through the serial RapidIO LCSBA1CSR, defined in the RapidIO programming model. See Local configuration space base address 1 command and status register. An external serial RapidIO master can set the value of LCSBA1CSR with a maintenance packet. Then subsequent read and write packets whose RapidIO addresses match the window defined by LCSBA1CSR are translated to the local address range indicated by CCSRBAR.

2.6.3 Accessing Reserved Registers and Bits

Reserved registers and bits in the CCSR memory space are not guaranteed to have predictable values.

In general, reads of reserved bits return zeros, but software should not rely on the value of any reserved bit being a zero or remaining consistent.

Unless otherwise specified, when writing registers in CCSR memory space, reserved bits should be written with zeros.

Writing ones to a reserved bit may result in undefined operation.

2.6.4 Organization of CCSR Memory

The configuration, control, and status registers are grouped according to functional units.

Most functional units are allocated a 4-Kbyte address space for registers.

Some functional units have larger address spaces as defined by their programming models. The registers for these blocks are given their own larger regions of CCSR memory.

2.6.5 CCSR Address Map

The full register address of any CCSR is comprised of the CCSR window base address, specified in CCSRBAR (default address 00_FE00_0000h), plus the functional block base address, plus the specific register's offset within that block.

Table 2-104 shows the location of the functional block base addresses for the entire CCSR space. Cross-references are provided to the CCSR maps for each individual block.

Table 2-104. CCSR Block Base Address Map

Block Base Address (Hex)	Block	Section	Comments
0x00_0000 - 0x00_0FFF	Local access control-Local configuration control	Local Configuration Control Memory Map	-
	Local access control-Local access windows	Local Access Window (LAW) Memory Map	-
00_1000h-00_3FFFh	Reserved	-	-
0x00_4000 - 0x00_4FFF	Prefetch manager		-
00_5000h-00_7FFFh	Reserved		-
0x00_8000 - 0x00_8FFF	DDR memory controller	DDR Memory Controller	-
00_9000h-00_AFFFh	Reserved		-

Table continues on the next page...

Table 2-104. CCSR Block Base Address Map (continued)

Block Base Address (Hex)	Block	Section	Comments
00_B000h-00_FFFFh	Reserved	-	-
0x01_0000 - 0x01_0FFF	CoreNet platform cache (CPC)	CoreNet Platform Cache (CPC) Memory Map	-
01_1000h-01_7FFFh	Reserved		-
0x01_8000 - 0x01_8FFF	CoreNet coherency fabric (CCF)	CoreNet Coherency Fabric (CCF) Memory Map	-
	Reserved	-	-
0x02_0000 - 0x02_0FFF	PAMU partition 1 (OCN)	PAMU Memory Map	
0x02_1000 - 0x02_1FFF	PAMU partition 2 (AXI)	PAMU Memory Map	
0x02_2000 - 0x02_2FFF	PAMU partition 3 (QMan, BMan)	PAMU Memory Map	
0x04_0000 - 0x04_FFFF	MPIC-Global registers	MPIC Memory Map	Gbl config: 04_1000h Gbl timers: 04_1100h
0x05_0000 - 0x05_FFFF	MPIC-Interrupt source registers	MPIC Memory Map	External IRQs: 05_0000h Internal IRQs: 05_1200h
0x06_0000 - 0x06_FFFF	MPIC-Processor (core) registers	MPIC Memory Map	-
07_0000h-0B_FFFFh	Reserved	-	-
0x0C_0000 - 0x0D_FFFF	Rapid I/O	Serial RapidIO Memory Map	-
0x0E_0000 - 0x0E_0FFF	Device configuration/ pin control	Device Configuration Memory Map	-
0x0E_1000 - 0x0E_1FFF	Clocking	Clocking Memory Map	-
0x0E_2000 - 0x0E_2FFF	Run control/ power management (RCPM)	RCPM Memory Map/Register Definition	-
0E_3000h-0E_7FFFh	Reserved	-	-

Table continues on the next page...

Table 2-104. CCSR Block Base Address Map (continued)

Block Base Address (Hex)	Block	Section	Comments
0x0E_8000 - 0x0E_8FFF	Security fuse processor (SFP)	Security fuse processor (SFP) memory map	-
0x0E_9000-0x0E_9FFF	Reserved	-	-
0x0E_A000 - 0x0E_AFFF	SerDes control 1	SRDS Memory Map/Register Definition	-
0x0E_B000 - 0x0E_BFFF	SerDes control 2	SRDS2 Memory Map/Register Definition	-
0E_C000h-0E_FFFFh	Reserved		-
0x0F_0000 - 0x0F_0FFF	Thermal monitor unit	TMU Memory Map/Register Definition	-
	Reserved	-	-
0x10_0000 - 0x10_0FFF	DMA controller 1	DMA controller memory map	-
0x10_1000 - 0x10_1FFF	DMA controller 2	DMA controller memory map	-
0x10_2000 - 0x10_2FFF	DMA controller 3	DMA controller memory map	
	Reserved	-	-
0x11_0000 - 0x11_0FFF	Enhanced serial peripheral interface (eSPI)	Enhanced serial peripheral interface (eSPI) memory map	-
0x11_4000 - 0x11_4FFF	Enhanced secure digital high capacity (eSDHC)	eSDHC memory map/register definition	-
	Reserved	-	-
0x11_8000 - 0x11_8FFF	Dual I2C controller 1	I2C Controller Memory Map	I^2C 1: 11_8000h I^2C 2: 11_8100h
0x11_9000 - 0x11_9FFF	Dual I2C controller 2	I2C Controller Memory Map	I^2C 3: 11_9000h I^2C 4: 11_9100h
11_A000h-11_BFFFh	Reserved	-	-
0x11_C000 - 0x11_CFFF	DUART controller 1	DUART Memory Map/Register Definition	UART1: 11_C500h (DUART1) UART2: 11_C600h (DUART1)
0x11_D000 - 0x11_DFFF	DUART controller 2	DUART Memory Map/Register Definition	UART3: 11_D500h (DUART2) UART4: 11_D600h (DUART2)
11_E000h-12_3FFFh	Reserved	-	-

Table continues on the next page...

Table 2-104. CCSR Block Base Address Map (continued)

Block Base Address (Hex)	Block	Section	Comments
0x12_4000 - 0x12_4FFF	Integrated Flash Controller (IFC) - Global registers		-
0x12_5000 - 0x12_5FFF	Integrated Flash Controller (IFC) - Run-time registers		-
12_6000h-12_FFFFh	Reserved	-	-
0x13_0000 - 0x13_0FFF	GPIO 1 controller	GPIO Memory Map/Register Definition	-
0x13_1000 - 0x13_1FFF	GPIO 2 controller	GPIO Memory Map/Register Definition	-
0x13_2000 - 0x13_2FFF	GPIO 3 controller	GPIO Memory Map/Register Definition	-
0x13_3000 - 0x13_3FFF	GPIO 4 controller	GPIO Memory Map/Register Definition	-
13_4000h-13_7FFFh	Reserved	-	-
13_8000h	Pre-boot loader (PBL)	Reserved Address Space Used as Internal PBL Commands	Software cannot write to the PBL CCSR space directly. However, special PBL commands may be leveraged during pre-boot initialization by referencing specific CCSR offsets (unique commands have unique CCSR offsets). See Reserved Address Space Used as Internal PBL Commands," for more information.
13_9000h-1D_FFFFh	Reserved	-	-
0x1E_0000 - 0x1F_FFFF	Rapid I/O message manager / general (RMAN)		See the DPAA reference manual.
1E_1000h-1E_3000h	Reserved		-
	Reserved	-	-
0x21_0000 - 0x21_0FFF	USB 1 (dual role)	USB Memory Map	-
0x21_1000 - 0x21_1FFF	USB 2 (dual role)	USB Memory Map	-
21_2000h-21_3FFFh	Reserved	-	-

Table continues on the next page...

Table 2-104. CCSR Block Base Address Map (continued)

Block Base Address (Hex)	Block	Section	Comments
21_5000h-21_FFFFh	Reserved	-	-
0x22_0000 - 0x22_0FFF	SATA 1	SATA memory map/register definition	-
0x22_1000 - 0x22_1FFF	SATA 2	SATA memory map/register definition	-
22_2000h-22_9FFFh	Reserved	-	-
0x24_0000 - 0x24_3FFF	PCI Express controller 1	PCI Express memory-mapped registers	When used without the SR-IOV feature: • 0000h - 0FFFh PCIe1 registers • 1000h - 7FFFh Reserved When used with the SR-IOV feature: • 0000h - 0FFFh PF0/Shared • 1000h - 1FFFh PF0 VF • 2000h - 2FFFh PF1 • 3000h - 3FFFh PF1 VF • 4000h - 7FFFh Reserved
0x25_0000 - 0x25_0FFF	PCI Express controller 2	PCI Express memory-mapped registers	Offsets (PCI Express 2 does not support SR-IOV): • 0000h - 0FFFh PCIe2 registers • 1000h - 7FFFh Reserved
0x26_0000 - 0x26_0FFF	PCI Express controller 3	PCI Express memory-mapped registers	Offsets (PCI Express 3 does not support SR-IOV): • 0000h - 0FFFh PCIe3 registers • 1000h - 7FFFh Reserved
0x27_0000 - 0x27_0FFF	PCI Express controller 4	PCI Express memory-mapped registers	Offsets (PCI Express 4 does not support SR-IOV): • 0000h - 0FFFh PCIe4 registers • 1000h - 7FFFh Reserved
28_0000h-29_FFFFh	Reserved	-	-
0x30_0000 - 0x30_FFFF	SEC	-	See the SEC reference manual.
31_0000h-31_1FFFh	Reserved	-	-
0x31_2000 - 0x31_2FFF	Data compression engine (DCE)		-
31_3000h-31_3FFFh	Reserved	-	-
0x31_4000 - 0x31_4FFF	Security monitor		

Table continues on the next page...

Table 2-104. CCSR Block Base Address Map (continued)

Block Base Address (Hex)	Block	Section	Comments
31_5000h-31_5FFFh	Reserved	-	-
0x31_6000 - 0x31_6FFF	Pattern matching engine (PME)	-	See the DPAA reference manual.
31_7000h-31_7FFFh	Reserved	-	-
0x31_8000 - 0x31_9FFF	Queue manager (QMAN)	-	See the DPAA reference manual.
31_9000h-31_9FFFh	Reserved	-	-
0x31_A000 - 0x31_AFFF	Buffer manager (BMAN)	-	See the DPAA reference manual.
31_B000h-3F_FFFFh	Reserved	-	-
0x40_0000 - 0x4F_FFFF	Frame manager (FMAN)	-	See the DPAA reference manual.
50_0000h-60_FFFFh	Reserved	-	See the DPAA reference manual.
0xC2_0000 - 0xC2_0FFF	Cluster 0 L2 cache	-	
C6_0000h-CA_FFFFh	Reserved	-	

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 memory controller interface signals
- Integrated Flash controller interface signals
- DUART interface signals
- I^2C interface signals
- Enhanced SPI interface signals
- Enhanced SDHC interface signals
- MPIC interface signals
- Hi-speed serial interface signals (SerDes)
- USB controller interface signals
- IEEE 1588 timestamp signals
- Ethernet management interface signals
- Ethernet controller RGMII interface signals
- DMA controller interface signals
- General-purpose input/output, system control, power management, and debug signals
- Clock, JTAG, and test 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 table provides a summary of the signals grouped by function. Note that the direction of 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 alternate function or reset configuration input signals).

Table 3-1. T2080 Signals Reference by Functional Block

Name	Description	Alternate Function(s)	Number of Signals	I/O
DDR Memory Interface see DDR Controller for more details				
D1_MAPAR_ERR_B	Address Parity Error		1	I
D1_MAPAR_OUT	Address Parity Out	-	1	O
D1_MA[0:15]	Address	-	16	O
D1_MBA[0:2]	Bank Select	-	3	O
D1_MCAS_B	Column Address Strobe	-	1	O
D1_MCKE[0:3]	Clock Enable	-	4	O
D1_MCK[0]	Clock	-	1	O
D1_MCK[1:3]	Clock	-	3	O
D1_MCK[0:3]_B	Clock Complement	-	4	O
D1_MCS[0:3]_B	Chip Select	-	4	O
D1_MDIC[0:1]	Driver Impedence Calibration	-	2	I/O
D1_MDM[0:8]	Data Mask	-	9	O
D1_MDQS[0:8]	Data Strobe	-	9	I/O
D1_MDQS[0:8]_B	Data Strobe	-	9	I/O
D1_MDQ[0:63]	Data	-	64	I/O
D1_MECC[0:7]	Error Correcting Code	-	8	I/O
D1_MODT[0:2]	On Die Termination	-	3	O
D1_MODT[3]	On Die Termination	-	1	O
D1_MRAS_B	Row Address Strobe	-	1	O
D1_MWE_B	Write Enable	-	1	O
Integrated Flash Controller see IFC Controller for more details				
IFC_AD[0]	IFC Address / Data	cfg_gpinput[0]	1	I/O
IFC_AD[1:2]	IFC Address / Data	cfg_gpinput[1:2]	2	I/O
IFC_AD[3:7]	IFC Address / Data	cfg_gpinput[3:7]	5	I/O
IFC_AD[8:15]	IFC Address / Data	cfg_rcw_src[0:7]	8	I/O
IFC_AVD	IFC Address Valid		1	I/O
IFC_A[16]	IFC Address		1	I/O
IFC_A[17]	IFC Address		1	I/O

Table continues on the next page...

Table 3-1. T2080 Signals Reference by Functional Block (continued)

Name	Description	Alternate Function(s)	Number of Signals	I/O
IFC_A[18]	IFC Address		1	I/O
IFC_A[19]	IFC Address		1	I/O
IFC_A[20]	IFC Address		1	I/O
IFC_A[21]	IFC Address	cfg_dram_type	1	I/O
IFC_A[22]	IFC Address	-	1	I/O
IFC_A[23]	IFC Address	-	1	I/O
IFC_A[24]	IFC Address	-	1	I/O
IFC_A[25:27]	IFC Address	GPIO2[25:27] , IFC_WP_B[1:3]	3	I/O
IFC_A[28:31]	IFC Address	GPIO2[28:31]	1	I/O
IFC_CLE	IFC Command Latch Enable / Write Enable	cfg_rcw_src[8]	1	I/O
IFC_CLK[0:1]	IFC Clock	-	2	O
IFC_CS[0]_B	IFC Chip Select	-	1	I/O
IFC_CS[1]_B	IFC Chip Select	GPIO2[10]	1	I/O
IFC_CS[2]_B	IFC Chip Select	GPIO2[11]	1	I/O
IFC_CS[3]_B	IFC Chip Select	GPIO2[12]	1	I/O
IFC_CS[4]_B	IFC Chip Select	GPIO1[9]	1	I/O
IFC_CS[5]_B	IFC Chip Select	GPIO1[10]	1	I/O
IFC_CS[6]_B	IFC Chip Select	GPIO1[11]	1	I/O
IFC_CS[7]_B	IFC Chip Select	GPIO1[12]	1	I/O
IFC_NDDDR_CLK	IFC NAND DDR Clock	-	1	O
IFC_NDDQS	IFC DQS Strobe	-	1	I/O
IFC_OE_B	IFC Output Enable	-	1	I/O
IFC_PAR[0]	IFC Address & Data Parity	GPIO2[13]	1	I/O
IFC_PAR[1]	IFC Address & Data Parity	GPIO2[14]	1	I/O
IFC_PERR_B	IFC Parity Error	GPIO2[15]	1	I/O
IFC_RB[0]_B	IFC Ready / Busy CS0	-	1	I
IFC_RB[1]_B	IFC Ready / Busy CS1	-	1	I
IFC_TE	IFC External Transceiver Enable	-	1	O
IFC_WE0_B	IFC Write Enable	-	1	I/O
IFC_WP[0]_B	IFC Write Protect	-	1	O
DUART See DUART Controller for more details				
UART1_CTS_B	Clear To Send	GPIO1[21]	1	I/O
UART1_RTS_B	Ready to Send	GPIO1[19]	1	I/O
UART1_SIN	Receive Data	GPIO1[17]	1	I/O
UART1_SOUT	Transmit Data	GPIO1[15]	1	I/O
UART2_CTS_B	Clear To Send	GPIO1[22]	1	I/O

Table continues on the next page...

Table 3-1. T2080 Signals Reference by Functional Block (continued)

Name	Description	Alternate Function(s)	Number of Signals	I/O
UART2_RTS_B	Ready to Send	GPIO1[20]	1	I/O
UART2_SIN	Receive Data	GPIO1[18]	1	I/O
UART2_SOUT	Transmit Data	GPIO1[16]	1	I/O
I2C See I2C Detailed Signal Descriptions for more details				
IIC1_SCL	Serial Clock (supports PBL)	-	1	I/O
IIC1_SDA	Serial Data (supports PBL)	-	1	I/O
IIC2_SCL	Serial Clock	-	1	I/O
IIC2_SDA	Serial Data	-	1	I/O
eSPI Interface See eSPI detailed signal descriptions for more details				
SPI_CLK	SPI Clock	-	1	I/O
SPI_CS[0]_B	SPI Chip Select	GPIO2[0]/SDHC_DAT[4]	1	I/O
SPI_CS[1] _B	SPI Chip Select	GPIO2[1]/SDHC_DAT[5]/ SDHC_CMD_DIR		I/O
SPI_CS[2] _B	SPI Chip Select	GPIO2[2]/SDHC_DAT[6]/ SDHC_DAT0_DIR		I/O
SPI_CS[3] _B	SPI Chip Select	GPIO2[3]/SDHC_DAT[7]/ SDHC_DAT123_DIR		I/O
SPI_MISO	Master In Slave Out	-	1	I
SPI_MOSI	Master Out Slave In	-	1	I/O
eSDHC See eSDHC signal descriptions for more details				
SDHC_CLK	Host to Card Clock	GPIO2[9]	1	I/O
SDHC_CMD	Command/Response	GPIO2[4]	1	I/O
SDHC_DAT[0]	Data	GPIO2[5]	1	I/O
SDHC_DAT[1:3]	Data	GPIO2[6:8]	3	I/O
SDHC_VS	Voltage Select	IRQ[3]/GPIO1[23]	1	O
SDHC_DAT0_DIR	DAT0 line direction control	SPI_CS_B[2]	1	O
SD_CMD	CMD line connect to card		1	I/O
SD_CMD_DIR	CMD line direction control	SPI_CS_B[1]	1	I/O
SD_DAT123_DIR	DAT1-DAT3 lines direction control	SPI_CS_B[3]	1	O
SDHC_CD_B	SDHC Card Detect	GPIO4[24]	1	I/O
SDHC_WP	SDHC Write Protect	GPIO4[25]	1	I/O
SDHC_DAT[7:4]	DAT[7:4} line in 8-bit mode Not used in other modes	SPI_CS_B[3:0]	4	I/O

Table continues on the next page...

Table 3-1. T2080 Signals Reference by Functional Block (continued)

Name	Description	Alternate Function(s)	Number of Signals	I/O
SDHC_DAT3	DAT3 line in 4/8-bit mode or configured as card detection pin May be configured as card detection pin in 1-bit mode	GPIO2[8]	1	I/O
SDHC_DAT2	DAT2 line or Read Wait in 4-bit mode Read Wait in 1-bit mode	GPIO2[7]	1	I/O
SDHC_DAT1	DAT1 line in 4/8-bit mode Also used to detect interrupt in 1/4-bit mode	GPIO2[6]	1	I/O
SDHC_DAT0	DAT0 line in all modes Also used to detect busy state	GPIO2[5]	1	I/O
Multicore Programmable Interrupt Controller				
IRQ[0]	External Interrupt		1	I/O
IRQ[1]	External Interrupt		1	I/O
IRQ[2]	External Interrupt		1	I/O
IRQ[3]	External Interrupt	GPIO1[23] , SDHC_VS	1	I/O
IRQ[4:11]	External Interrupt	GPIO1[24:31]	1	I/O
IRQ_OUT_B	Interrupt Output	EVT_B[9]	1	I/O
LP Trust				
LP_TMP_DETECT_B	Low Power Tamper Detect	-	1	I
Trust				
TMP_DETECT_B	Tamper Detect	-	1	I
System Control				
HRESET_B	Hard Reset	-	1	I/O
PORESET_B	Power On Reset	-	1	I
RESET_REQ_B	Reset Request (POR or Hard)	-	1	O
Power Management				
ASLEEP	Asleep	GPIO1[13], cfg_xvdd_sel	1	O
SYSCLK				
SYSCLK	System Clock	-	1	I
DDR Clocking				
D1_DDRCLK	DDR Controller Clock	-	1	I
RTC				
RTC	Real Time Clock	GPIO1[14]	1	I/O
Debug				

Table continues on the next page...

Table 3-1. T2080 Signals Reference by Functional Block (continued)

Name	Description	Alternate Function(s)	Number of Signals	I/O
CKSTP_OUT_B	Checkstop Out	-	1	O
CLK_OUT	Clock Out	-	1	O
EVT_B[0]	Event 0	-	1	I/O
EVT_B[1]	Event 1	-	1	I/O
EVT_B[2]	Event 2	-	1	I/O
EVT_B[3]	Event 3	-	1	I/O
EVT_B[4]	Event 4	-	1	I/O
DFT				
SCAN_MODE_B	Reserved	-	1	I
TEST_SEL_B	Reserved	-	1	I
JTAG				
TCK	Test Clock	-	1	I
TDI	Test Data In	-	1	I
TDO	Test Data Out	-	1	O
TMS	Test Mode Select	-	1	I
TRST_B	Test Reset	-	1	I
Analog Signals				
D1_MVREF	SSTL Reference Voltage	-	1	I/O
D1_TPA	DDR Controller 1 Test Point Analog	-	1	I/O
TD1_ANODE	Thermal diode anode	-	1	I/O
TD1_CATHODE	Thermal diode cathode	-	1	I/O
TH_TPA	Thermal Test Point Analog	-	1	-
Serdes 1				
SD1_IMP_CAL_RX	SerDes Receive Impedence Calibration	-	1	I
SD1_IMP_CAL_TX	SerDes Transmit Impedance Calibration	-	1	I
SD1_REF1_CLK_P	SerDes PLL 1 Reference Clock	-	1	I
SD1_REF1_CLK_N	SerDes PLL 1 Reference Clock Complement	-	1	I
SD1_REF2_CLK_P	SerDes PLL 2 Reference Clock	-	1	I
SD1_REF2_CLK_N	SerDes PLL 2 Reference Clock Complement	-	1	I
SD1_RX[0:7]_P	SerDes Receive Data (positive)	-	8	I

Table continues on the next page...

Table 3-1. T2080 Signals Reference by Functional Block (continued)

Name	Description	Alternate Function(s)	Number of Signals	I/O
SD1_RX[0:7]_N	SerDes Receive Data (negative)	-	8	I
SD1_TX[0:7]_P	SerDes Transmit Data (positive)	-	8	O
SD1_TX[0:7]_N	SerDes Transmit Data (negative)	-	8	O
Serdes 2				
SD2_IMP_CAL_RX	SerDes Receive Impedence Calibration	-	1	I
SD2_IMP_CAL_TX	SerDes Transmit Impedance Calibration	-	1	I
SD2_REF1_CLK_P	SerDes PLL 1 Reference Clock	-	1	I
SD2_REF1_CLK_N	SerDes PLL 1 Reference Clock Complement	-	1	I
SD2_REF2_CLK_P	SerDes PLL 2 Reference Clock	-	1	I
SD2_REF2_CLK_N	SerDes PLL 2 Reference Clock Complement	-	1	I
SD2_RX[0:7]_P	SerDes Receive Data (positive)	-	8	I
SD2_RX[0:7]_N	SerDes Receive Data (negative)	-	8	I
SD2_TX[0:7]_P	SerDes Transmit Data (positive)	-	8	O
SD2_TX[0:7]_N	SerDes Transmit Data (negative)	-	8	O
USB PHY 1 & 2 See USB External Signals for more details				
USB1_DRVVBUS	USB PHY Digital signal - Drive VBUS	-	1	O
USB1_PWRFAULT	USB PHY Digital signal - Power Fault	-	1	I
USB1_UDM	USB PHY Data Minus	-	1	I/O
USB1_UDP	USB PHY Data Plus	-	1	I/O
USB1_UID	USB PHY ID Detect	-	1	I
USB1_VBUSCLMP	USB PHY VBUS	-	1	I
USB2_DRVVBUS	USB PHY Digital signal - Drive VBUS	-	1	O
USB2_PWRFAULT	USB PHY Digital signal - Power Fault	-	1	I
USB2_UDM	USB PHY Data Minus	-	1	I/O
USB2_UDP	USB PHY Data Plus	-	1	I/O

Table continues on the next page...

Table 3-1. T2080 Signals Reference by Functional Block (continued)

Name	Description	Alternate Function(s)	Number of Signals	I/O
USB2_UID	USB PHY ID Detect	-	1	I
USB2_VBUSCLMP	USB PHY VBUS	-	1	I
USB_IBIAS_REXT	USB PHY Impedance Calibration	-	1	I/O
USBCLK	USB PHY Clock In		1	I
IEEE1588 See “QorIQ Data Path Acceleration Architecture (DPAA) Reference Manual” for more details				
TSEC_1588_ALARM_OUT[1]	Alarm Out 1	GPIO3[3]	1	I/O
TSEC_1588_ALARM_OUT[2]	Alarm Out 2	GPIO3[4]	1	I/O
TSEC_1588_CLK_IN	Clock In	GPIO3[0]	1	I/O
TSEC_1588_CLK_OUT	Clock Out	GPIO3[5]	1	I/O
TSEC_1588_PULSE_OUT[1]	Pulse Out 1	GPIO3[6]	1	I/O
TSEC_1588_PULSE_OUT[2]	Pulse Out 2	GPIO3[7]	1	I/O
TSEC_1588_TRIG_IN[1]	Trigger In 1	GPIO3[1]	1	I/O
TSEC_1588_TRIG_IN[2]	Trigger In 2	GPIO3[2]	1	I/O
Ethernet Management Interface 1 See “QorIQ Data Path Acceleration Architecture (DPAA) Reference Manual” for more details				
EMI1_MDC	Management Data Clock	-	1	O
EMI1_MDI/O	Management Data In/ Out	-	1	I/O
Ethernet Management Interface 2 See “QorIQ Data Path Acceleration Architecture (DPAA) Reference Manual” for more details				
EMI2_MDC	Management Data Clock (1.2V open drain)	-	1	O
EMI2_MDI/O	Management Data In/ Out (1.2V open drain)	-	1	I/O
Ethernet Controller (RGMII) 1 See “QorIQ Data Path Acceleration Architecture (DPAA) Reference Manual” for more details				
EC1_GTX_CLK	Transmit Clock Out	GPIO3[16]	1	I/O
EC1_GTX_CLK125	Reference Clock	GPIO3[17]	1	I/O
EC1_RXD[3:0]	Receive Data	GPIO3[18:21]	4	I/O
EC1_RX_CLK	Receive Clock	GPIO3[23]	1	I/O
EC1_RX_CTL	Receive Data Valid	GPIO3[22]	1	I/O
EC1_TXD[0:3]	Transmit Data	GPIO3[14:11]	1	I/O
EC1_TX_CTL	Transmit Enable	GPIO3[15]	1	I/O
Ethernet Controller (RGMII) 2 See “QorIQ Data Path Acceleration Architecture (DPAA) Reference Manual” for more details				

Table continues on the next page...

Table 3-1. T2080 Signals Reference by Functional Block (continued)

Name	Description	Alternate Function(s)	Number of Signals	I/O
EC2_GTX_CLK	Transmit Clock Out	GPIO4[28]	1	I/O
EC2_GTX_CLK125	Reference Clock	GPIO4[29]	1	I/O
EC2_RXD[3:0]	Receive Data	GPIO3[28:31]	4	I/O
EC2_RX_CLK	Receive Clock	GPIO4[31]	1	I/O
EC2_RX_CTL	Receive Data Valid	GPIO4[30]	1	I/O
EC2_TXD[0:3]	Transmit Data	GPIO3[27:24]	1	O
EC2_TX_CTL	Transmit Enable	GPIO4[27]	1	O
I2C 3 & 4 See I2C Detailed Signal Descriptions for more details				
IIC3_SCL	Serial Clock	GPIO4[0]	1	I/O
IIC3_SDA	Serial Data	GPIO4[1]	1	I/O
IIC4_SCL	Serial Clock	GPIO4[2]	1	I/O
IIC4_SDA	Serial Data	GPIO4[3]	1	I/O
DMA See DMA signal descriptions for more details				
DMA1_DACK0_B	DMA1 channel 0 acknowledge	GPIO4[5]	1	I/O
DMA1_DDONE0_B	DMA1 channel 0 done	GPIO4[6]	1	I/O
DMA1_DREQ0_B	DMA1 channel 0 request	GPIO4[4]	1	I/O
DMA2_DACK0_B	DMA2 channel 0 acknowledge	GPIO4[8] / EVT_B[7]	1	I/O
DMA2_DDONE0_B	DMA2 channel 0 done	GPIO4[9] / EVT_B[8]	1	I/O
DMA2_DREQ0_B	DMA2 channel 0 request	GPIO4[7]	1	I/O
General Purpose Input/Output				
GPIO1[9]	General Purpose Input/ Output	ASLEEP	1	I/O
GPIO1[10]	General Purpose Input/ Output	RTC	1	I/O
GPIO1[11]	General Purpose Input/ Output	IFC_CS_B[6]	1	I/O
GPIO1[12]	General Purpose Input/ Output	IFC_CS_B[7]	1	I/O
GPIO1[13]	General Purpose Input/ Output	ASLEEP	1	O
GPIO1[14]	General Purpose Input/ Output	RTC	1	I/O
GPIO1[15]	General Purpose Input/ Output	UART1_SOUT	1	I/O
GPIO1[16]	General Purpose Input/ Output	UART2_SOUT	1	I/O
GPIO1[17]	General Purpose Input/ Output	UART1_SIN	1	I/O

Table continues on the next page...

Table 3-1. T2080 Signals Reference by Functional Block (continued)

Name	Description	Alternate Function(s)	Number of Signals	I/O
GPIO1[18]	General Purpose Input/ Output	UART2_SIN	1	I/O
GPIO1[19]	General Purpose Input/ Output	UART1_RTS_B	1	I/O
GPIO1[20]	General Purpose Input/ Output	UART2_RTS_B	1	I/O
GPIO1[21]	General Purpose Input/ Output	UART1_CTS_B	1	I/O
GPIO1[22]	General Purpose Input/ Output	UART2_CTS_B	1	I/O
GPIO1[23]	General Purpose Input/ Output	IRQ[3] / SDHC_VS	9	I/O
GPIO1[24:31]	General Purpose Input/ Output	IRQ[4:11]	8	I/O
GPIO2[0]	General Purpose Input/ Output	SPI_CS_B[0] / SDHC_DAT[4]	1	I/O
GPIO2[1]	General Purpose Input/ Output	SPI_CS_B[1] / SDHC_DAT[5] / SDHC_CMD_DIR	1	I/O
GPIO2[2]	General Purpose Input/ Output	SPI_CS_B[2] / SDHC_DAT[6] / SDHC_DAT0_DIR	1	I/O
GPIO2[3]	General Purpose Input/ Output	SPI_CS_B[3] / SDHC_DAT[7] / SDHC_DAT123_DIR	1	I/O
GPIO2[4]	General Purpose Input/ Output	SDHC_CMD	1	I/O
GPIO2[5]	General Purpose Input/ Output	SDHC_DAT[0]	1	I/O
GPIO2[6:8]	General Purpose Input/ Output	SDHC_DAT[1:3]	3	I/O
GPIO2[9]	General Purpose Input/ Output	SDHC_CLK	1	I/O
GPIO2[10]	General Purpose Input/ Output	IFC_CS_B[1]	1	I/O
GPIO2[11]	General Purpose Input/ Output	IFC_CS_B[2]	1	I/O
GPIO2[12]	General Purpose Input/ Output	IFC_CS_B[3]	1	I/O
GPIO2[13]	General Purpose Input/ Output	IFC_PAR[0]	1	I/O
GPIO2[14]	General Purpose Input/ Output	IFC_PAR[1]	1	I/O
GPIO2[15]	General Purpose Input/ Output	IFC_PERR_B	1	I/O
GPIO2[25:31]	General Purpose Input/ Output	IFC_A[25:31]	7	I/O

Table continues on the next page...

Table 3-1. T2080 Signals Reference by Functional Block (continued)

Name	Description	Alternate Function(s)	Number of Signals	I/O
GPIO3[0]	General Purpose Input/ Output	TSEC_1588_CLK_IN	1	I/O
GPIO3[1]	General Purpose Input/ Output	TSEC_1588_TRIG_IN1	1	I/O
GPIO3[2]	General Purpose Input/ Output	TSEC_1588_TRIG_IN2	1	I/O
GPIO3[3]	General Purpose Input/ Output	TSEC_1588_ALARM_OUT1	1	I/O
GPIO3[4]	General Purpose Input/ Output	TSEC_1588_ALARM_OUT2	1	I/O
GPIO3[5]	General Purpose Input/ Output	TSEC_1588_CLK_OUT	1	I/O
GPIO3[6]	General Purpose Input/ Output	TSEC_1588_PULSE_OUT1	1	I/O
GPIO3[7]	General Purpose Input/ Output	TSEC_1588_PULSE_OUT2	1	I/O
GPIO3[16]	General Purpose Input/ Output	EC1_GTX_CLK	1	I/O
GPIO3[17]	General Purpose Input/ Output	EC1_GTX_CLK[125]	1	I/O
GPIO3[18:21]	General Purpose Input/ Output	EC1_RXD[3:0]	4	I/O
GPIO3[23]	General Purpose Input/ Output	EC1_RX_CLK	1	I/O
GPIO3[22]	General Purpose Input/ Output	EC1_RX_CTL	1	I/O
GPIO3[14:11]	General Purpose Input/ Output	EC1_TXD[0:3]	4	I/O
GPIO3[15]	General Purpose Input/ Output	EC1_TX_CTL	1	I/O
GPIO3[16]	General Purpose Input/ Output	EC1_GTX_CLK	1	I/O
GPIO3[17]	General Purpose Input/ Output	EC1_GTX_CLK125	1	I/O
GPIO3[18:21]	General Purpose Input/ Output	EC1_RXD[3:0]	4	I/O
GPIO3[22]	General Purpose Input/ Output	EC1_RX_CTL	1	I/O
GPIO3[23]	General Purpose Input/ Output	EC1_RX_CLK	1	I/O
GPIO3[24:27]	General Purpose Input/ Output	EC2_TXD[3:0]	4	I/O
GPIO3[28:31]	General Purpose Input/ Output	EC2_RXD[3:0]	4	I/O

Table continues on the next page...

Table 3-1. T2080 Signals Reference by Functional Block (continued)

Name	Description	Alternate Function(s)	Number of Signals	I/O
GPIO4[30]	General Purpose Input/ Output	EC2_RX_CTL	1	I/O
GPIO4[31]	General Purpose Input/ Output	EC2_RX_CLK	1	I/O
GPIO4[0]	General Purpose Input/ Output	IIC3_SCL	1	I/O
GPIO4[1]	General Purpose Input/ Output	IIC3_SDA	1	I/O
GPIO4[2]	General Purpose Input/ Output	IIC4_SCL	1	I/O
GPIO4[3]	General Purpose Input/ Output	IIC4_SDA	1	I/O
GPIO4[5]	General Purpose Input/ Output	DMA1_DREQ0_B	1	I/O
GPIO4[6]	General Purpose Input/ Output	DMA1_DACK0_B	1	I/O
GPIO4[7]	General Purpose Input/ Output	DMA1_DDONE0_B	1	I/O
GPIO4[8]	General Purpose Input/ Output	DMA2_DACK0_B	1	I/O
GPIO4[9]	General Purpose Input/ Output	DMA2_DDONE0_B	1	I/O
GPIO4[24]	General Purpose Input/ Output	SDHC_CD_B	1	I/O
GPIO4[25]	General Purpose Input/ Output	SDHC_WP	1	I/O
GPIO4[28]	General Purpose Input/ Output	EC2_GTX_CLK	1	I/O
GPIO4[29]	General Purpose Input/ Output	EC2_GTX_CLK125	1	I/O

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. The detailed interpretation of their voltage levels during reset is described in Power-On Reset Configuration .

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 hardware specifications 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 hardware specifications document.

Table 3-2. T2080 reset configuration signals

Reset configuration name	Functional interface	Functional Signal Name	Default
cfg_rcw_src[0:7]	IFC	IFC_AD[8:15]	1111 1111 [1]
cfg_rcw_src[8]	IFC	IFC_CLE	1 [1]
cfg_ifc_te	IFC	IFC_TE	1
cfg_dram_type	IFC	IFC_A[21]	1
cfg_gpinput[0:7]	IFC	IFC_AD[0:7]	1111 1111
cfg_xvdd_sel	Power management	ASLEEP	1

1. cfg_rcw_src[0:8]=1_1111_1111 is not a valid setting; They must be set to one of the valid options defined in Reset configuration word (RCW) source

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 . 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 SPI_CS_B[0:3]/ SDHC_DAT[4:7]/GPIO2[0:3] can be utilized by either the SPI block, the eSDHC block, or by GPIO 2. Because these signals alternatively service three different functional blocks, their function is determined at the device level by RCW[SPI_BASE]. The following sections describe external signal functional selection using the RCW.

3.4.1 UART and GPIO1 signal multiplexing

For the UART signals, the following relationships apply:

- UART1 shares signals with GPIO1 and UART3
- UART2 shares signals with GPIO1 and UART4

The functionality of these signals is determined by the UART_BASE field in the reset configuration word (RCW[UART_BASE]).

Table 3-3. UART signal configuration

Signal name	Signal function	RCW[UART_BASE]or RCW[366:368]
UART1_SOUT	GPIO1[15]	000
	UART1_SOUT	011, 100, 101, 110, 111
UART1_SIN	GPIO1[17]	000
	UART1_SIN	011, 100, 101, 110, 111
UART1_RTS_B	GPIO1[19]	000, 011, 101
	UART1_RTS_B	100, 110
	UART3_SOUT	111
UART1_CTS_B	GPIO1[21]	000, 011, 101
	UART1_CTS_B	100, 110
	UART3_SIN	111
UART2_SOUT	GPIO1[16]	000, 011, 110
	UART2_SOUT	110, 101, 110
UART2_SIN	GPIO1[18]	000, 011, 100
	UART2_SIN	110, 101, 111
UART2_RTS_B	GPIO1[20]	000, 011, 100, 101
	UART2_RTS_B	110
	UART4_SOUT	111
UART2_CTS_B	GPIO1[22]	000, 011, 100, 101
	UART2_CTS_B	110
	UART4_SIN	111

3.4.2 ASLEEP and GPIO1 signal multiplexing

The power management signal, ASLEEP, is shared with GPIO1[13]. The functionality of this signal is determined by the ASLEEP field in the reset configuration word (RCW[ASLEEP]).

Table 3-4. ASLEEP signal configuration

Signal name	Signal function	RCW[ASLEEP]
ASLEEP	ASLEEP	0
	GPIO1[13]	1

3.4.3 RTC and GPIO1 signal multiplexing

The clocking signal, RTC, is shared with GPIO1[14]. The functionality of this signal is determined by the RTC field in the reset configuration word (RCW[RTC]).

Table 3-5. RTC signal configuration

Signal name	Signal function	RCW[RTC]
RTC	RTC	0
	GPIO1[14]	1

3.4.4 eSDHC and GPIO2 signal multiplexing

The eSDHC interface shares signals with GPIO2. The functionality of these signals are determined by the SDHC_BASE field in the reset configuration word (RCW[SDHC_BASE]).

Table 3-6. SDHC_BASE signal configuration

Signal name	Signal function	RCW[SDHC_BASE]
SDHC_CMD	SDHC_CMD	0
	GPIO2[4]	1
SDHC_DAT[0:3]	SDHC_DAT[0:3]	0
	GPIO2[5:8]	1
SDHC_CLK	SDHC_CLK	0
	GPIO2[9]	1

3.4.5 MPIC and debug signal multiplexing

The interrupt controller (MPIC) shares a signal with debug signal EVT_B[9]. The functionality of this signal is determined by the IRQ_OUT field in the reset configuration word (RCW[IRQ_OUT]).

Table 3-7. IRQ_OUT signal configuration

Signal name	Signal function	RCW[IRQ_OUT]
IRQ_OUT_B	IRQ_OUT_B	0
	EVT_B[9]	1

3.4.6 MPIC and GPIO1 signal multiplexing

The interrupt controller (MPIC) shares signals with GPIO1. The functionality of these signals is determined by the IRQ_BASE and IRQ_EXT fields in the reset configuration word (RCW[IRQ_BASE, IRQ_EXT] or RCW[373:381]).

Table 3-8. IRQ*n* signal configuration

Signal name	Signal function	RCW[370+i], 3≤i≤11
IRQ[i] 3≤i≤11	IRQ[i]	0
	GPIO1[i+20]	1
		RCW[IRQ_EXT]
IRQ[3:11]	IRQ[3:11]	000
	GPIO1[24:29]/GPIO1[31]	001

3.4.7 eSPI, eSDHC, and GPIO2 signal multiplexing

The eSPI shares signals with the eSDHC and GPIO2. The functionality of these signals is determined by the SPI_BASE and SPI_EXT fields in the reset configuration word (RCW[SPI_BASE, SPI_EXT]).

Table 3-9. eSPI Signal configuration

Signal name	Signal function	RCW[SPI_BASE]
SPI_CS_B[0:3]	SPI_CS_B[0:3]	00
	SDHC_DAT[4:7]	01
	GPIO2[0:3]	10
		RCW[SPI_EXT]
SPI_CS_B[0:2]	SPI_CS_B[0:2]	000
	SDHC_DAT[4:6]	001

3.4.8 IFC, GPIO1, and GPIO2 signal multiplexing

The integrated Flash controller shares signals with GPIO1 and GPIO2. The functionality of these signals is determined by the IFC_GRP_[*a*]_BASE fields in the reset configuration word as described in the following table.

Table 3-10. IFC signal configuration

Signal name	Signal function	IFC_GRP_[a]_BASE]					
		E1	E2	D	C	B	A
IFC_CS_B[1:3]	IFC_CS_B[1:3]	0	-	-	-	-	-
	GPIO2[10:12]	1	-	-	-	-	-
IFC_CS_B[4:7]	IFC_CS_B[4:7]	-	0	-	-	-	-
	GPIO1[9:12]	-	1	-	-	-	-
IFC_PAR[0:1]	IFC_PAR[0:1]	-	-	0	-	-	-
	GPIO2[13:14]	-	-	1	-	-	-
IFC_PERR_B	IFC_PERR_B	-	-	0	-	-	-
	GPIO2[15]	-	-	1	-	-	-
IFC_PAR[2:3]	IFC_PAR[2:3]	-	-	0	-	-	-
	GPIO2[16:17]	-	-	1	-	-	-
IFC_A[26:31]	IFC_A[26:31]	-	-	-	0	-	-
	GPIO2[18:23]	-	-	-	1	-	-
IFC_AD[28]	IFC_AD[28]	-	-	-	-	00	-
	GPIO2[28]	-	-	-	-	01	-
IFC_AD[29:31]	IFC_AD[29:31]	-	-	-	-	00	-
	GPIO2[29:31]	-	-	-	-	01	-
	IFC_RB_B[2:4]	-	-	-	-	10	-
IFC_AD[25:27]	IFC_AD[25:27]	-	-	-	-	-	00
	GPIO2[25:27]	-	-	-	-	-	01
	IFC_WP_B[1:3]	-	-	-	-	-	10

3.4.9 IEEE 1588 and GPIO3 signal multiplexing

The IEEE 1588 interface shares signals with GPIO3. The functionality of these signals is determined by the 1588 field in the reset configuration word (RCW[1588]).

Table 3-11. IEEE 1588 signal configuration

Signal name	Signal function	RCW[1588]
TSEC_1588_CLK_IN	TSEC_1588_CLK_IN	0
	GPIO3[0]	1
TSEC_1588_TRIG_IN1	TSEC_1588_TRIG_IN1	0
	GPIO3[1]	1
TSEC_1588_TRIG_IN2	TSEC_1588_TRIG_IN2	0
	GPIO3[2]	1
TSEC_1588_ALARM_OUT1	TSEC_1588_ALARM_OUT1	0
	GPIO3[3]	1

Table continues on the next page...

Table 3-11. IEEE 1588 signal configuration (continued)

Signal name	Signal function	RCW[1588]
TSEC_1588_ALARM_OUT2	TSEC_1588_ALARM_OUT2	0
	GPIO3[4]	1
TSEC_1588_CLK_OUT	TSEC_1588_CLK_OUT	0
	GPIO3[5]	1
TSEC_1588_PULSE_OUT1	TSEC_1588_PULSE_OUT1	0
	GPIO3[6]	1
TSEC_1588_PULSE_OUT2	TSEC_1588_PULSE_OUT2	0
	GPIO3[7]	1

3.4.10 Ethernet controller 1 and GPIO3 signal multiplexing

The Ethernet controller 1 (EC1) RGMII interface shares signals with GPIO3. The functionality of these signals is determined by the EC1 field in the reset configuration word (RCW[EC1]). When RGMII is enabled for a particular MAC, it always takes precendence over SerDes-based interfaces.

Table 3-12. EC1 RGMII signal configuration

Signal name	Signal function	RCW[EC1]
EC1_TXD[3:0]	EC1_TXD[3:0]	00
	GPIO3[11:14]	10
EC1_GTX_CLK	EC1_GTX_CLK (FM_MAC3 RGMII)	00
	GPIO3[16]	10
EC1_GTX_CLK125	EC1_GTX_CLK125 (FM_MAC3 RGMII)	00
	GPIO3[17]	10
EC1_RX_CLK	EC1_RX_CLK	00
	GPIO3[23]	10
EC1_RXD[3:0]	EC1_RXD[3:0]	00
	GPIO3[18:21]	10
EC1_TX_CTL	EC1_TX_CTL (FMan-MAC3 RGMII)	00
	GPIO3[15]	10
EC1_RX_CTL	EC1_RX_CTL (FMan-MAC3 RGMII)	00
	GPIO3[22]	10

3.4.11 Ethernet controller 2 and GPIO3 signal multiplexing

The Ethernet controller 2 (EC2) RGMII interface shares signals with GPIO3. In addition, the chip can be configured to use either Frame Manager 1 MAC5 (FMan1-MAC5) or Frame Manager 2 MAC6 (FMan2-MAC6) for EC2. The functionality of these signals is determined by the EC2 field in the reset configuration word (RCW[EC2]). When RGMII is enabled for a particular MAC, it always takes precendence over SerDes-based interfaces.

Table 3-13. EC2 RGMII signal configuration

Signal name	Signal function	RCW[EC2]
EC2_TXD[3:0]	EC2_TXD[3:0] FMan MAC4 RGMII	00
	EC2_TXD[3:0] FMan MAC10 RGMII	01
	GPIO3[20:23]	10
EC2_GTX_CLK	EC2_GTX_CLK FMan MAC4 RGMII	00
	EC2_GTX_CLK FMan MAC10 RGMII	01
	GPIO3[25]	10
EC2_RX_CLK	EC2_RX_CLK FMan MAC4 RGMII	00
	EC2_RX_CLK FMan MAC10 RGMII	01
	GPIO3[27]	10
EC2_RXD[3:0]	EC2_RXD[3:0] FMan MAC4 RGMII	00
	EC2_RXD[3:0] FMan MAC10 RGMII	01
	GPIO3[28:31]	10
EC2_GTX_CLK125	EC2_GTX_CLK125 (FMan MAC4 RGMII)	00
	EC2_GTX_CLK125 (FMan MAC10 RGMII)	01
	GPIO4[29]	10
EC2_TX_CTL	EC2_TX_CTL (FMan-MAC4 RGMII)	00
	EC2_TX_CTL (FMan-MAC10 RGMII)	01
	GPIO4[27]	10
EC2_RX_CTL	EC2_RX_CTL (FMan-MAC4 RGMII)	00
	EC2_RX_CTL (FMan-MAC10 RGMII)	01
	GPIO4[30]	10

3.4.12 I^2C3 and GPIO4 signal multiplexing

The I^2C3 interface shares signals with GPIO4. The functionality of these signals is determined by the I2C3 field in the reset configuration word (RCW[I2C3]).

Table 3-14. I2C3 signal configuration

Signal name	Signal function	RCW[I2C3]
IIC3_SCL	IIC3_SCL	00
	GPIO4[0]	01
IIC3_SDA	IIC3_SDA	00
	GPIO4[1]	01

3.4.13 I^2C4, GPIO4, and Debug Signal Multiplexing

The I^2C4 interface shares signals with GPIO4, and Debug. The functionality of these signals is determined by the I2C4 field in the reset configuration word (RCW[I2C4]).

Table 3-15. I2C4 Signal Configuration

Signal Name	Signal Function	RCW[I2C4]
IIC4_SCL	IIC4_SCL	00
	GPIO4[2]	01
	EVT_B[5]	10
IIC4_SDA	IIC4_SDA	00
	GPIO4[3]	01
	EVT_B[6]	10

3.4.14 DMA1 and GPIO4 signal multiplexing

DMA controller 1, channel 0 (DMA1) shares signals with GPIO4. The functionality of these signals is determined by the DMA1 field in the reset configuration word (RCW[DMA1]).

Table 3-16. DMA1 signal configuration

Signal name	Signal function	RCW[DMA1]
DMA1_DREQ_B[0]	DMA1_DREQ_B[0]	0
	GPIO4[4]	1
DMA1_DACK_B[0]	DMA1_DACK_B[0]	0
	GPIO4[5]	1
DMA1_DDONE_B[0]	DMA1_DDONE_B[0]	0
	GPIO4[6]	1

3.4.15 DMA2, debug, and GPIO4 signal multiplexing

DMA controller 2, channel 0 (DMA2) shares signals with debug and GPIO4. The functionality of these signals is determined by the DMA2 field in the reset configuration word (RCW[DMA2]).

Table 3-17. DMA2 signal configuration

Signal name	Signal function	RCW[DMA2]
DMA2_DREQ_B[0]	DMA2_DREQ_B[0]	00
	GPIO4[7]	01, 10
DMA2_DACK_B[0]	DMA2_DACK_B[0]	00
	GPIO4[8]	01
	EVT_B[7]	10
DMA2_DDONE_B[0]	DMA2_DDONE_B[0]	00
	GPIO4[9]	01
	EVT_B[8]	10

3.5 Output Signal States During Reset

When a system reset is initiated (HRESET_B or PORESET_B sampled asserted by the chip), the chip aborts all current internal and external transactions and releases all bidirectional I/O signals to a high-impedance state. See *Reset, Clocking, and Initialization* for a complete description of the reset functionality.

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

Note that signals associated with the interface selected as RCW source via cfg_rcw_src[0:8] are enabled and active while the chip is in reset (that is, while HRESET_B is being driven asserted by the chip, but 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. Additionally, the configuration, control, and status registers are described. 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 4-2 and Table 4-3 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	References (Section/Page)
PORESET_B	I	Power on reset input.	System Control Signals
HRESET_B	I/O	Hard reset input. Functions as an output during initial steps in the power on reset sequence. See Power-On Reset Sequence for more information.	System Control Signals
RESET_REQ_B	O	Reset request output. An internal block requests that either HRESET_B or PORESET_B be asserted.	System Control Signals
CKSTP_OUT_B	O	Checkstop out.	System Control Signals

Table continues on the next page...

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

Signal	I/O	Description	References (Section/Page)
SYSCLK	I	Primary clock input.	External Clock Signals
RTC	I	Real time clock input.	External Clock Signals
SD1_REF_CLK*n*/ SD1_REF_CLK*n*_B SD2_REF_CLK*n*/ SD2_REF_CLK*n*_B	I	SerDes high-speed interface reference clock *n*.	External Clock Signals
CLK_OUT	O	Diagnostic clock output.	External Clock Signals
DDRCLK	I	Reference clock for DDR controller, when running in asynchronous mode.	External Clock Signals

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 4-2. 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 and Power-On Reset Configuration 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.
HRESET_B	I/O	Hard reset. Causes the chip to abort all current internal and external transactions and set all registers to their default values. HRESET_B may be asserted completely asynchronously with respect to all other signals. HRESET_B is driven as an output during the first part of the power on reset sequence, after which, it becomes an input, allowing external devices to stall/hold the reset sequence. See Power-On Reset Sequence for more information.	
		State Meaning	Asserted/Negated-See for more information on the interpretation of the other chip signals during reset.
		Timing	Assertion/Negation-The chip data sheet gives specific timing information for this signal.
RESET_REQ_B	O	Reset request. Indicates to the board (system in which the chip is embedded) that a condition requiring the assertion of HRESET_B or PORESET_B has been detected.	
		State Meaning	Asserted-An event has triggered a request for either a hard reset or a power on reset. See Reset Request Status Register (DCFG_CCSR_RSTRQSR1) for more information.

Table continues on the next page...

Table 4-2. System Control Signals: Detailed Signal Descriptions (continued)

Signal	I/O	Description	
			Negated-Indicates no reset request.
		Timing	Assertion/Negation-May occur any time. Once asserted, RESET_REQ_B does not negate until either HRESET_B or PORESET_B is asserted.
CKSTP_OUT_B	O	Checkstop out.	
		State Meaning	Asserted-Indicates that the chip is in a checkstop state. Negated-Indicates normal operation. After CKSTP_OUT_B has been asserted, it is negated after the next negation (low-to-high transition) of PORESET_B.
		Timing	Assertion-May occur at any time; may be asserted asynchronously to the input clocks. Negation-Must remain asserted until the chip has been reset with a PORESET_B.
ASLEEP	O	Power Management Signal. See the "Run Control and Power Management" chapter.	
TMP_DETECT_B	I	Tamper Detect.	
LP_TMP_DETECT_B	I	Low Power Tamper Detect.	

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 4-3. Clock External Signals-Detailed Signal Descriptions

Signal	I/O	Description	
SYSCLK	I	System clock (SYSCLK). SYSCLK is the primary clock input to the chip.	
		Timing	Assertion/Negation-See the chip data sheet for specific timing information for this signal.
RTC	I	Real time clock. May be used (optionally) to clock the time base of the cores. The RTC timing specifications are given in the chip data sheet . This signal can also be used (optionally) to clock the global timers in the programmable interrupt controller (PIC).	
		Timing	Assertion/Negation-See the chip data sheet for specific timing information for this signal.
SD*n*_REF_CLK*n*, SD*n*_REF_CLK*n*_B	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 .	
		Timing	Assertion/Negation-See the chip data sheet for specific timing information for these signals.

Table continues on the next page...

Table 4-3. Clock External Signals-Detailed Signal Descriptions (continued)

Signal	I/O	Description
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.
DDRCLK	I	DDR controller complex clock. DDRCLK is the clock source for the DDR memory controller complex except in the case where synchronous mode of operation is selected.

4.3 Local Configuration Control Registers

4.3.1 Accessing Configuration, Control, and Status Registers

The memory-mapped configuration, control, and status registers (CCSRs) occupy a 16-Mbyte region of memory. Their location is programmable using CCSR base address register high (CCSRBARH) and CCSR base address register low (CCSRBARL); together, these registers are referred to as CCSRBAR. The default base address for the CCSRs is 0x0_FE00_0000. CCSRBARH and CCSRBARL are part of the local access block of CCSR memory, which begins at offset 0x0 from CCSRBAR. Because CCSRBAR is at offset 0x0 from the beginning of the local access registers, CCSRBAR always points to itself.

4.3.1.1 Updating CCSRBARs

Updating the CCSRBARs that relocate the entire 16-Mbyte region of CCSR registers requires special treatment. The effect of the update must be guaranteed to be visible by the mapping logic before an access to the new location is seen.

To ensure this happens, use the following guidelines:

- CCSRBARH and CCSRBARL should be updated during initial configuration of the device when only one host or controller has access to the device.
  - If an external host on PCI Express or RapidIO is configuring the device, it should set CCSRBARH and CCSRBARL to the desired final values before the core is released to boot.
  - If the core is initializing the device, it should set CCSRBARH and CCSRBARL to the desired final values before enabling other I/O devices to access the device.

When updating CCSRBARH and CCSRBARL, use the following sequence:

1. Place a temporary LAW over the new planned CCSRBAR space (because the CCSR space has not been moved yet, the address of the LAW register is still relative to the old CCSR space). Program the LAWAR*n*[TRGT_ID] = 0x1E (a reserved setting) for this temporary LAW. When the LAW registers for this temporary LAW are read, it forces the initial write to the CCSR space to complete. Note that this LAW is only temporary.
2. The application that is performing the update must ensure that all other cores and applications are not accessing the configuration space during the procedure.
3. Read the current CCSRBARH and CCSRBARL using load word instructions.
4. Follow this with an **isync** instruction. This forces any outstanding accesses to configuration space to completion.
5. Write the new values for CCSRBARH and CCSRBARL to their old locations.

   The CCSRBARH has a shadow register. When the CCSRBARH has a new value written it loads a CCSRBARH shadow register. When the CCSRBARL is written, the CCSRBARH shadow register contents along with the CCSRBARL value are loaded into the CCSRBARH and CCSRBARL registers, respectively.

6. Follow this with a **sync** instruction.
7. Write a 1 to the commit bit (C) of CCSRAR at the old location.
8. Follow this with a **sync** instruction.
9. The temporary LAW configured in Step 1 above should now be disabled and reused elsewhere because it is no longer needed.

4.3.2 Accessing Alternate Configuration Space

An alternate configuration space can be accessed by configuring the ALTCBARH, ALTCBARL, and ALTCAR registers. These are intended to be used with the pre-boot loader (PBL) to allow the PBL to access a 16-Mbyte region of the memory map. By loading the proper PBL command in the serial ROM, the base address in ALTCBARH and ALTCBARL can be combined with the 24 bits of address offset supplied from the serial ROM to generate a 40-bit address that is mapped to the target specified in ALTCAR. Thus, by configuring these registers, the PBL has access to the entire memory map, one 16-Mbyte block at a time.

NOTE

Updates to the alternate configuration access window must be performed with care. The ALTCBARL and ALTCBARH must be written before setting ALTCAR[EN] or undefined results will occur.

While enabled, the alternate configuration space always takes precedence over the LAWs. The CoreNet Coherency Fabric (CCF) automatically disables the alternate configuration window when the PBL has completed its pre-boot initialization (PBI) process. Windows (including LAWs and the boot space translation window) can be enabled by the PBL during PBI to establish an initial address map that the cores see during their boot process.

The alternative configuration mechanism cannot be used to access external devices on the peripheral interfaces (SRIO or PCI Express). ALTCAR does not support SRIO or PCIe target IDs.

4.3.3 Boot Space Translation

When each core comes out of reset, its MMU has one 4 KB page defined at 0x0_FFFF_F*nnn*. Each core begins execution with the instruction at effective address 0x0_FFFF_FFFC. To get this instruction, the core's first instruction fetch is a burst read of boot code from effective address 0x0_FFFF_FFE0. For systems in which the boot code resides at a different address, the chip provides boot space translation capability. Note that boot space translation affects transactions initiated by all cores in the same manner.

The pre-boot loader can enable boot space translation, or the boot space translation can be set up by an external host when the device is configured to be in boot holdoff mode. If translation is to be performed to a page outside the default boot window (8 Mbytes at 0x0_FF80_0000 to 0x0_FFFF_FFFF), the external host or pre-boot loader must then also set up the target ID defining the target destination.

When an address falls within the range for the boot window space, boot space translation is applied (if enabled) to translate the address per Boot space translation register high (LCC_BSTRH), Boot space translation register low (LCC_BSTRL), and Boot space translation attribute register (LCC_BSTAR). If LCC_BSTAR[EN] bit is set and the access falls within the boot translation window size, a translation of the incoming address occurs. The lower address bits that represent the byte offset within the programmed size remain unchanged. The LCC_BSTRH and LCC_BSTRL are combined to form a translation address to replace the remaining upper address bits of the incoming address. The boot window location starts at address 0x0_FF80_0000 and extends to 0x0_FFFF_FFFF. The address range within the boot window that is translated by the boot space translation registers covers the region from 4 GB-BSTAR[SIZE] to 4 GB-1.

4.4 Local Configuration Control Memory Map

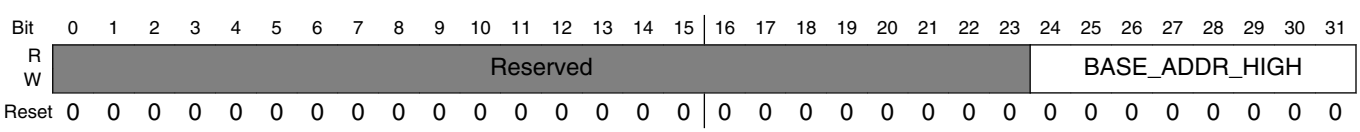
This section describes the CCSRs that control access to the configuration space, the alternate configuration space, and boot space translation as described in the previous sections.

LCC memory map

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
0	Configuration, control, and status registers base address register high (LCC_CCSRBARH)	32	R/W	0000_0000h	4.4.1/155
4	Configuration, control, and status registers base address register low (LCC_CCSRBARL)	32	R/W	FE00_0000h	4.4.2/156
8	Configuration, control, and status registers attribute register (LCC_CCSRAR)	32	R/W	0000_0000h	4.4.3/156
10	Alternate configuration base address register high (LCC_ALTCBARH)	32	R/W	0000_0000h	4.4.4/157
14	Alternate configuration base address register low (LCC_ALTCBARL)	32	R/W	0000_0000h	4.4.5/157
18	Alternate configuration attribute register (LCC_ALTCAR)	32	R/W	0000_0000h	4.4.6/158
20	Boot space translation register high (LCC_BSTRH)	32	R/W	0000_0000h	4.4.7/158
24	Boot space translation register low (LCC_BSTRL)	32	R/W	0000_0000h	4.4.8/159
28	Boot space translation attribute register (LCC_BSTAR)	32	R/W	01F0_000Bh	4.4.9/159

4.4.1 Configuration, control, and status registers base address register high (LCC_CCSRBARH)

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



LCC_CCSRBARH field descriptions

Field	Description
0–23 -	This field is reserved. Reserved
24–31 BASE_ADDR_HIGH	Identifies the upper portion (the 8 most significant bits) of the CCSR window base address.

4.4.2 Configuration, control, and status registers base address register low (LCC_CCSRBARL)

Address: 0h base + 4h offset = 4h

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	BASE_ADDR_LOW																															
Reset	1	1	1	1	1	1	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

LCC_CCSRBARL field descriptions

Field	Description
0–31 BASE_ADDR_ LOW	Identifies the lower portion of the CCSR window base address. Because the CCSR base address must be aligned on a 16-Mbyte boundary, bits 8-31 are ignored by hardware and always return 0x00_0000.

4.4.3 Configuration, control, and status registers attribute register (LCC_CCSRAR)

Address: 0h base + 8h offset = 8h

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

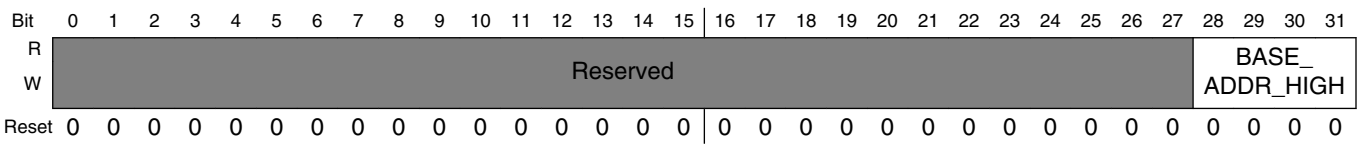
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LCC_CCSRAR field descriptions

Field	Description
0 C	Commit. Commits the CCSR base address configuration. Writing a 1 to this bit causes the CCSR base address programmed in CCSRBARH and CCSRBARL to take effect. When read, this bit always returns zero.
1–31 -	This field is reserved. Reserved

4.4.4 Alternate configuration base address register high (LCC_ALTCBARH)

Address: 0h base + 10h offset = 10h

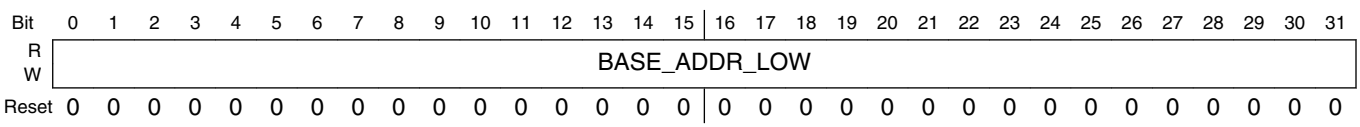


LCC_ALTCBARH field descriptions

Field	Description
0–27 -	This field is reserved. Reserved
28–31 BASE_ADDR_ HIGH	Identifies the upper portion (the 4 most significant bits) of the alternate configuration window base address.

4.4.5 Alternate configuration base address register low (LCC_ALTCBARL)

Address: 0h base + 14h offset = 14h



LCC_ALTCBARL field descriptions

Field	Description
0–31 BASE_ADDR_ LOW	Identifies the lower portion of the alternate configuration window base address. Because the base address must be aligned on a 16-Mbyte boundary, bits 8-31 are ignored by hardware and always return 0x00_0000.

4.4.6 Alternate configuration attribute register (LCC_ALTCAR)

Address: 0h base + 18h offset = 18h

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

LCC_ALTCAR field descriptions

Field	Description
0 EN	Enable for alternate configuration window. The alternate configuration window has a fixed size of 16 Mbytes. 0 Alternate configuration window is disabled. 1 Alternate configuration window is enabled.
1–3 -	Write reserved, read = 0
4–11 TRGT_ID	Identifies the device ID to target when a transaction hits in the 16-Mbyte address range defined by the alternate configuration window. See Global Source and Target IDs for encodings. **NOTE:** SRIO and PEX targets are not supported.
12–31 -	Write reserved, read = 0

4.4.7 Boot space translation register high (LCC_BSTRH)

Address: 0h base + 20h offset = 20h

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																												TRANS_ ADDR_HIGH			
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

LCC_BSTRH field descriptions

Field	Description
0–27 -	This field is reserved. Reserved
28–31 TRANS_ADDR_ HIGH	Identifies the upper portion (the 4 most significant bits) of the boot space translation address.

4.4.8 Boot space translation register low (LCC_BSTRL)

Address: 0h base + 24h offset = 24h

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	TRANS_ADDR_LOW																															
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

LCC_BSTRL field descriptions

Field	Description
0–31 TRANS_ADDR_LOW	Identifies the lower portion of the boot space translation address. **NOTE:** The least significant bits of the address representing offsets within the boot space (as defined by BSTAR[SIZE]) are treated as zero by the hardware. Reading them returns zeros.

4.4.9 Boot space translation attribute register (LCC_BSTAR)

Address: 0h base + 28h offset = 28h

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

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

LCC_BSTAR field descriptions

Field	Description
0 EN	Enable for boot space translation. 0 Boot space translation is disabled. 1 Boot space translation is enabled.
1–3 -	Write reserved, read = 0
4–11 TRGT_ID	Identifies the destination for accesses covered by boot space translation. See Global Source and Target IDs for encodings.
12–25 -	Write reserved, read = 0
26–31 SIZE	Boot space translation window size. The window size is 2 $^{(SIZE+1)}$ bytes. Minimum window size is 4 Kbytes; maximum window size is 8 Mbytes. Example settings: 000000-001010 Reserved 001011 4 Kbytes 001100 8 Kbytes

Table continues on the next page...

LCC_BSTAR field descriptions (continued)

Field	Description
	001101 16 Kbytes 010101 4 Mbytes 010110 8 Mbytes 010111-111111 Reserved

4.5 Clocking Memory Map

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

Clocking memory map

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E_1000	Core cluster n clock control/status register (Clocking_CLKCCSR)	32	R/W	0000_0000h	4.5.1/160
E_1010	Clock generator n hardware accelerator control/status register (Clocking_CLKCG1HWACSR)	32	R/W	0000_0000h	4.5.2/162
E_1030	Clock generator n hardware accelerator control/status register (Clocking_CLKCG2HWACSR)	32	R/W	0000_0000h	4.5.2/162
E_1050	Clock generator n hardware accelerator control/status register (Clocking_CLKCG3HWACSR)	32	R/W	0000_0000h	4.5.2/162
E_1070	Clock generator n hardware accelerator control/status register (Clocking_CLKCGHWACSR)	32	R/W	0000_0000h	4.5.2/162
E_1800	PLL cluster n general status register (Clocking_PLLC1GSR)	32	R	0000_0000h	4.5.3/163
E_1820	PLL cluster n general status register (Clocking_PLLC2GSR)	32	R	0000_0000h	4.5.3/163
E_1A00	Platform clock domain control/status register (Clocking_CLKPCSR)	32	R/W	0000_0000h	4.5.4/165
E_1C00	Platform PLL general status register (Clocking_PLLPGSR)	32	R/W	0000_0000h	4.5.5/167
E_1C20	DDR PLL general status register (Clocking_PLLDGSR)	32	R/W	0000_0000h	4.5.6/169

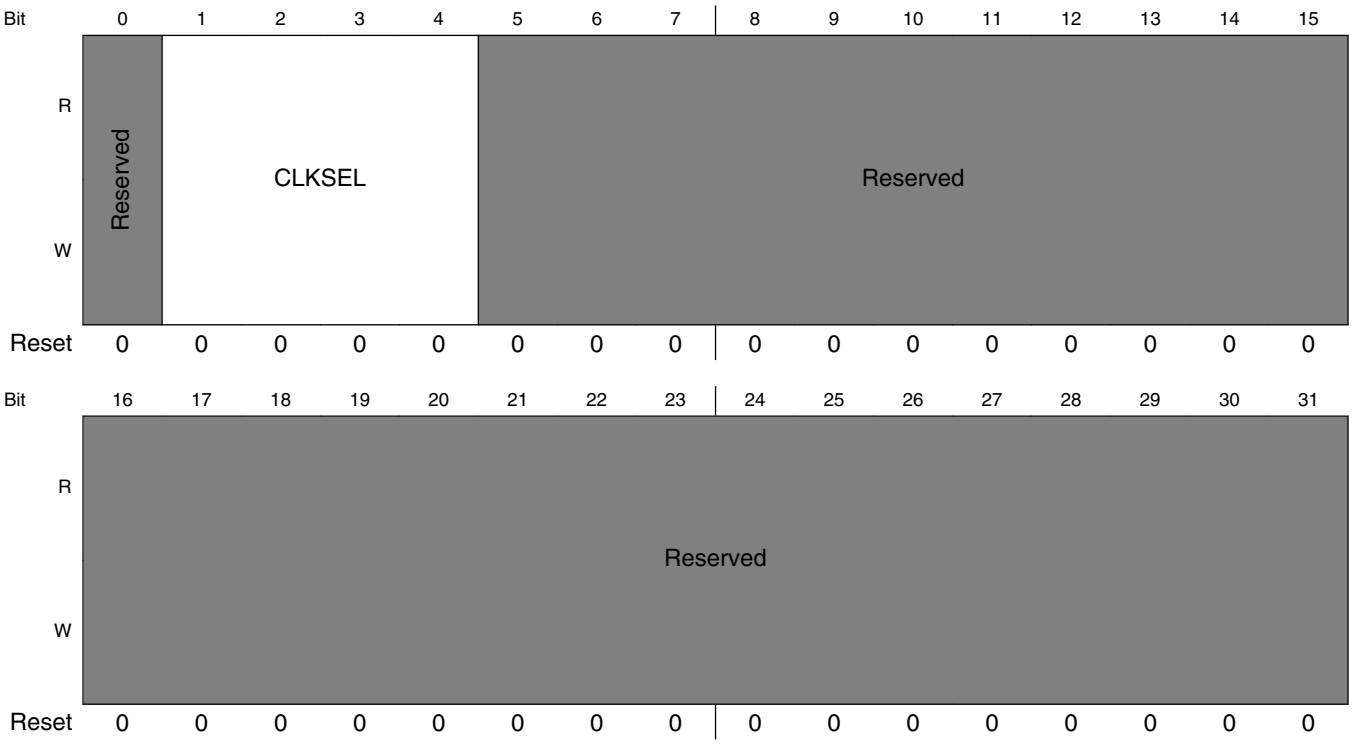
4.5.1 Core cluster n clock control/status register (Clocking_CLKCCSR)

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

Table 4-14. Core cluster assignments to CLKC*n*CSR registers

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

Address: E_1000h base + 0h offset = E_1000h



Clocking_CLKCCSR field descriptions

Field	Description
0 -	This field is reserved. Reserved
1–4 CLKSEL	Clock Select. Selects the clock source for the corresponding core cluster. See Table 4-14 above. 0000 Corresponding cluster group PLL1 output 0001 Corresponding cluster group PLL1 output divide-by-2 0010 Corresponding cluster group PLL1 output divide-by-4 0011 Reserved 0100 Corresponding cluster group PLL2 output 0101 Corresponding cluster group PLL2 output divide-by-2 0110 Corresponding cluster group PLL2 output divide-by-4 0111-1111 Reserved
5–31 -	This field is reserved. Reserved

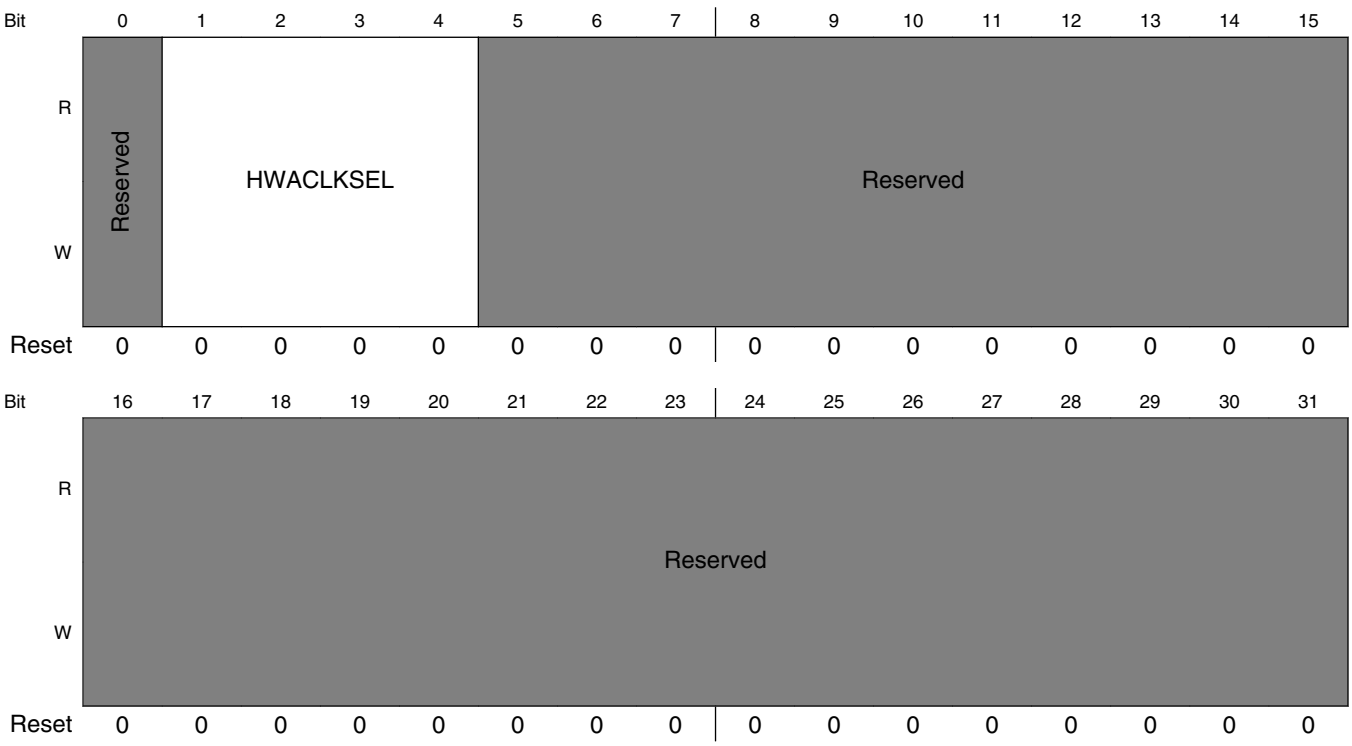
4.5.2 Clock generator n hardware accelerator control/status register (Clocking_CLKCG*n*HWACSR)

The CLKCG*n*HWACSR registers control the clock frequency selection for each hardware accelerator.

Table 4-18. Hardware accelerator assignments to CLKCG*n*HWACSR registers

Register	Cluster Group	Mux	Hardware Accelerator
CLKCG1HWACSR	CGA	M1	FMan
CLKCG2HWACSR	CGA	M1	eSDHC

Address: E_1000h base + 10h offset + (32d × i), where i=0d to 4d



Clocking_CLKCG*n*HWACSR field descriptions

Field	Description
0 -	This field is reserved. Reserved
1–4 HWACLKSEL	Hardware accelerator clock select. Selects the clock source for peripheral hardware accelerators. See Table 4-16 above CLKCG1HWACSR (CGA_M1): 0000 Reserved

Table continues on the next page...

Clocking_CLKCG*n*HWACSR field descriptions (continued)

Field	Description
	0001 CGA PLL1 divide-by-1 0010 CGA PLL1 divide-by-2 0011 CGA PLL1 divide-by-3 0100 CGA PLL1 divide-by-4 101 Platform clock 0110 CGA PLL2 divide-by-2 0111 CGA PLL2 divide-by-3 CLKCG2HWACSR (CGA_M2): 0000 Reserved 0001 CGA PLL2 divide-by-1 0010 CGA PLL2 divide-by-2 0011 CGA PLL2 divide-by-3 0100 CGA PLL2 divide-by-4 0101 Platform clock 0110 CGA PLL1 divide-by-2 0111 CGA PLL1 divide-by-3 1000-1111 Reserved
5–31 -	This field is reserved. Reserved

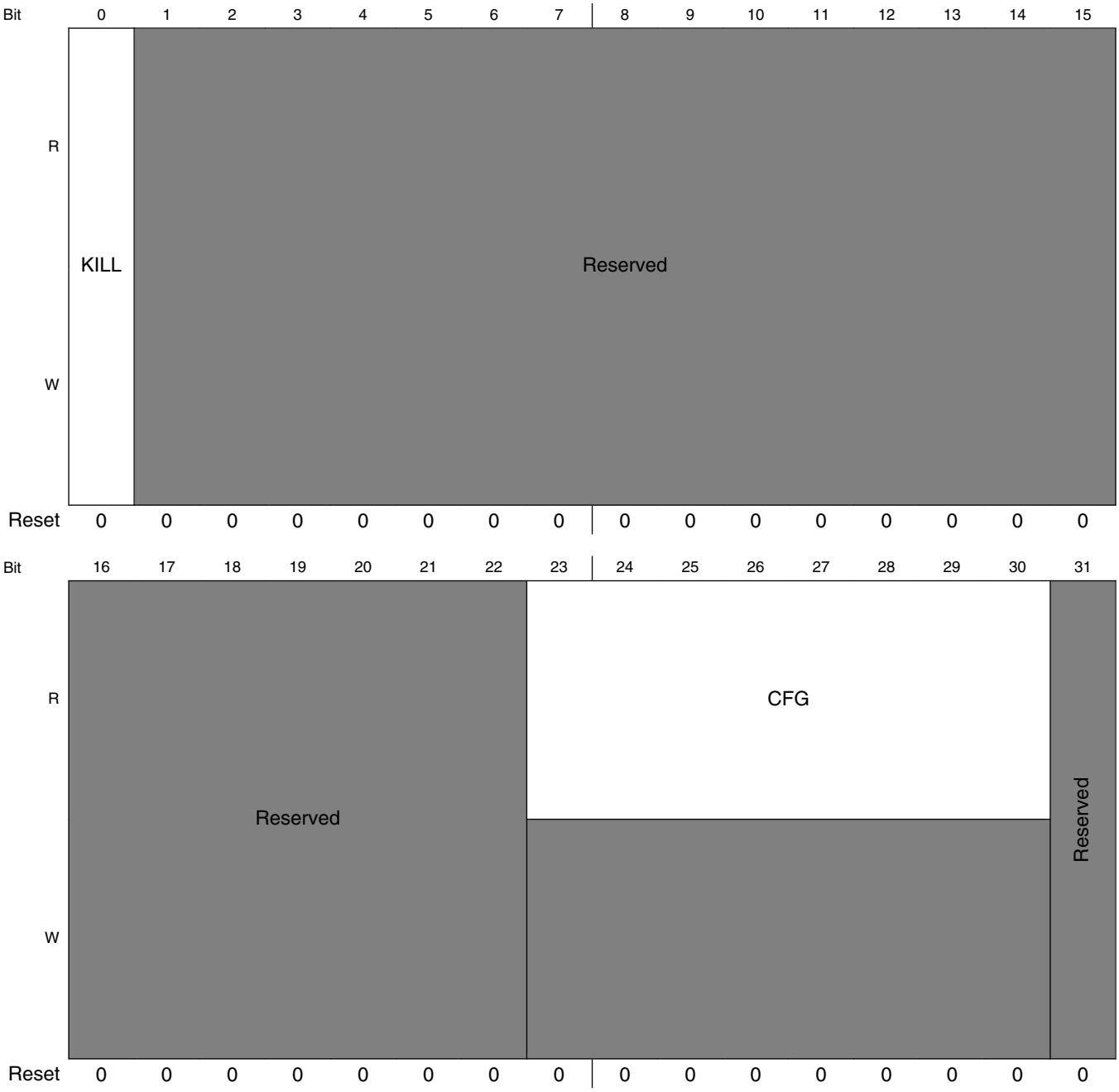
4.5.3 PLL cluster n general status register (Clocking_PLLC*n*GSR)

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

Table 4-28. PLL assignments to PLLC*n*GSR registers

Register	Cluster Group	PLL
PLLC1GSR	CGA	PLL1
PLLC2GSR	CGA	PLL2

Address: E_1000h base + 800h offset + (32d × i), where i=0d to 1d



Clocking_PLLC*n*GSR field descriptions

Field	Description
0 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. 0 PLL is active. 1 PLL is disabled.
1–22 -	This field is reserved. Reserved

Table continues on the next page...

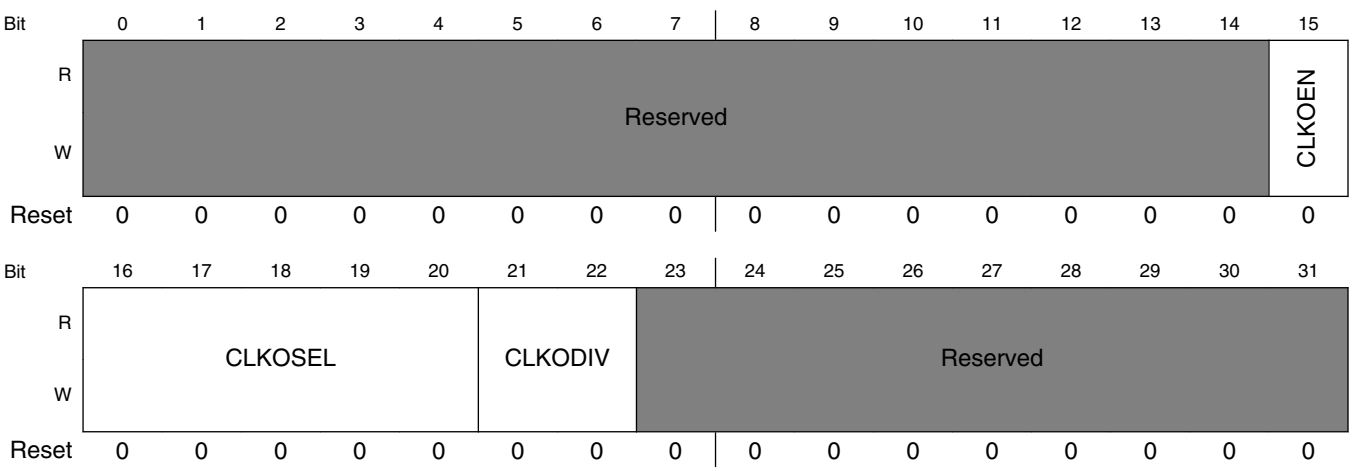
Clocking_PLLC*n*GSR field descriptions (continued)

Field	Description
23–30 CFG	Reflects the current PLL multiplier configuration. Indicates the frequency for this PLL. 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 hardware specifications for the supported frequencies.
31 -	This field is reserved. Reserved

4.5.4 Platform clock domain control/status register (Clocking_CLKPCSR)

The CLKPCSR register selects which signal to observe on the CLK_OUT1 pad.

Address: E_1000h base + A00h offset = E_1A00h



Clocking_CLKPCSR field descriptions

Field	Description
0–14 -	This field is reserved. Reserved
15 CLKOEN	CLK_OUT pad enable 0 Release CLK_OUT pad to high impedance 1 Enable CLK_OUT pad
16–20 CLKOSEL	Selects core related clock signal for observation on CLK_OUT pad 11001 Platform Clock /3 11010 Platform Clock /2

Table continues on the next page...

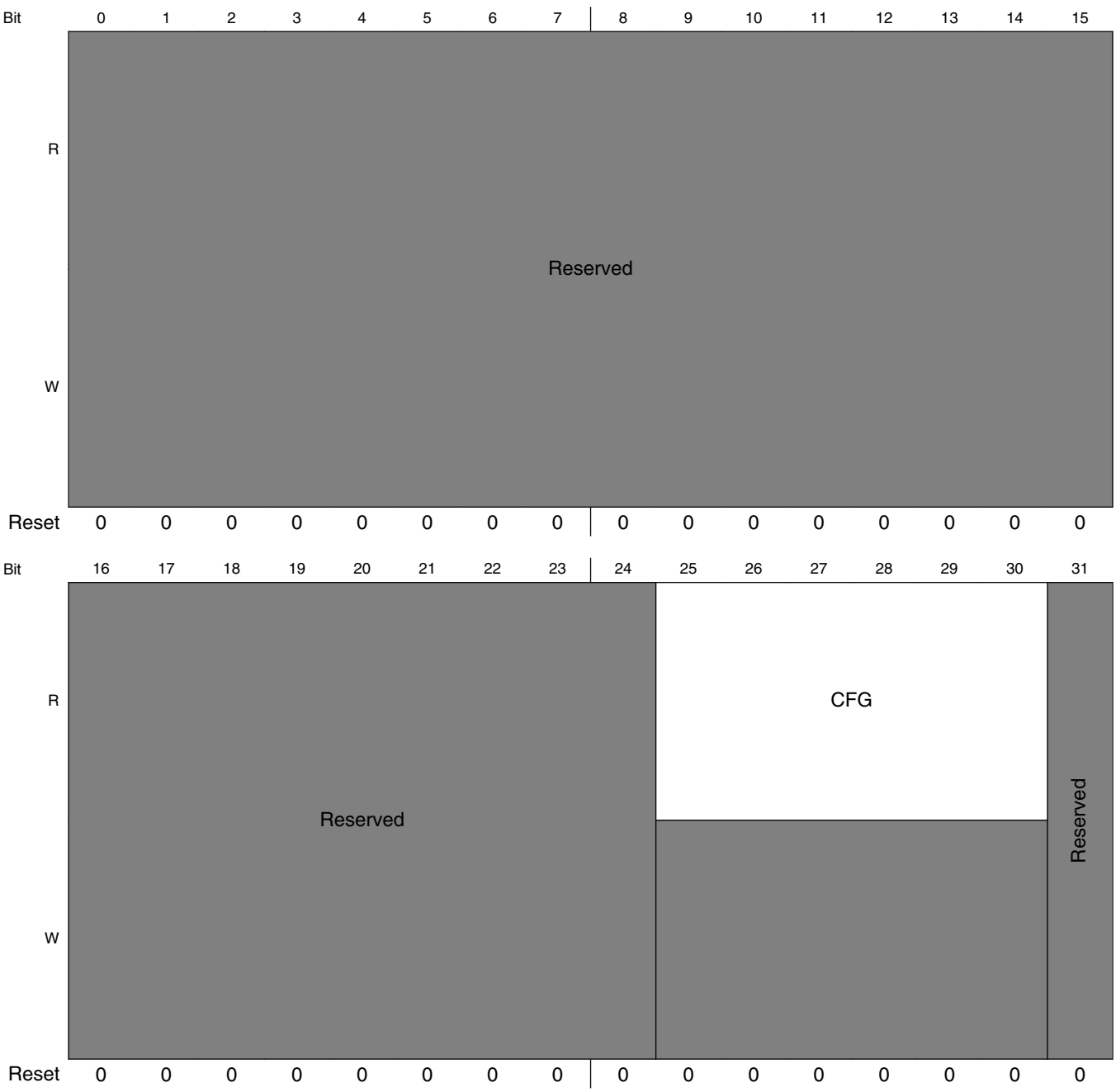
Clocking_CLKPCSR field descriptions (continued)

Field	Description
	11011 Platform Clock 11100-11101 Reserved 11110 Platform feedback clock 11111 SYSCLK All others Reserved
21–22 CLKODIV	Selects division setting for clock-out mux output to reduce the frequency of the signal to a more observable/measurable range. 00 No division (divide-by-1) 01 Divide-by-2 10 Divide-by-4 11 Divide-by-8
23–31 -	This field is reserved. Reserved

4.5.5 Platform PLL general status register (Clocking_PLLPGSR)

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

Address: E_1000h base + C00h offset = E_1C00h



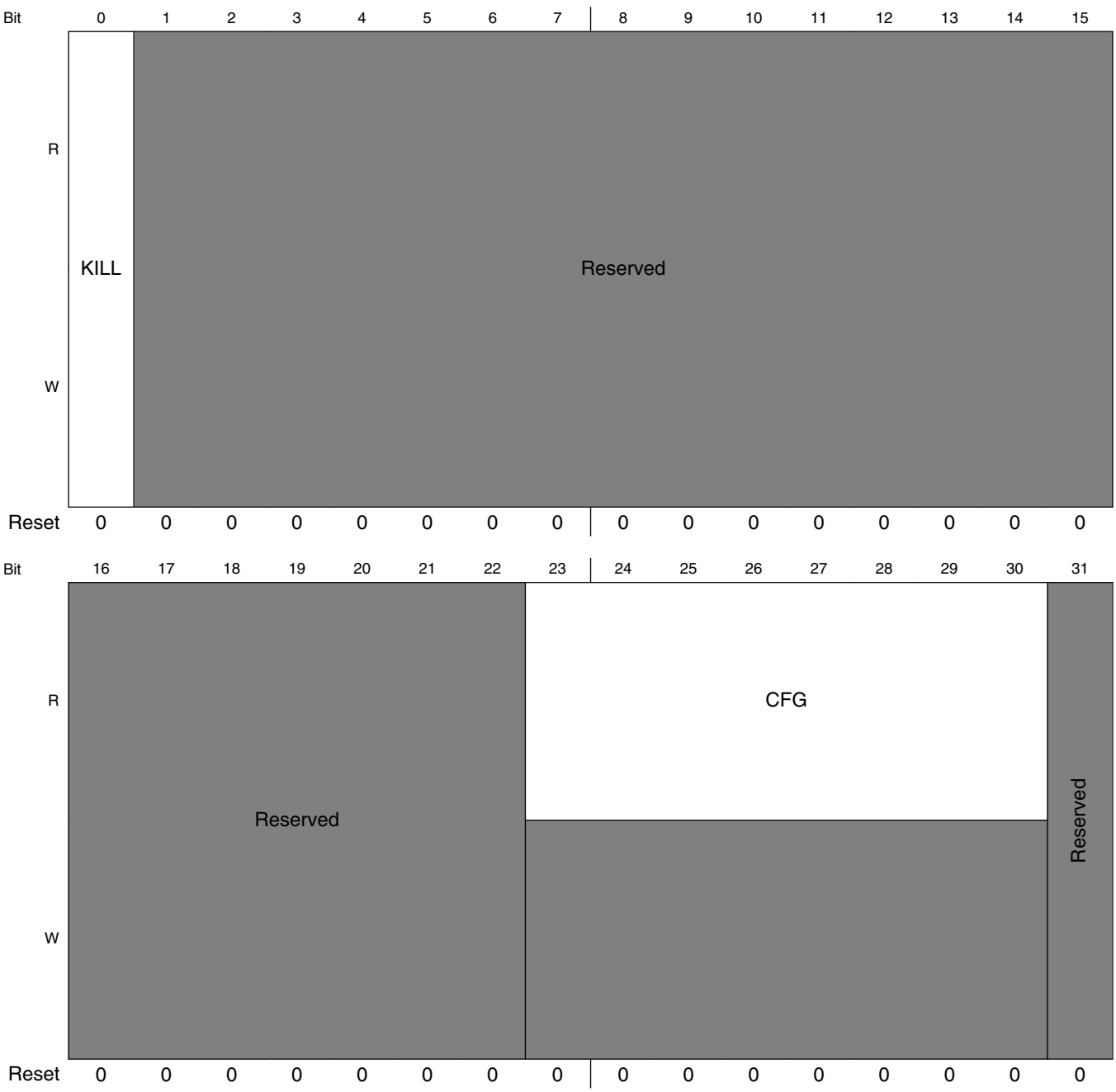
Clocking_PLLPGSR field descriptions

Field	Description
0–24 -	This field is reserved. Reserved
25–30 CFG	Reflects the current PLL multiplier configuration. Indicates the frequency for this PLL. 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 hardware specifications for the supported frequencies.
31 -	This field is reserved. Reserved

4.5.6 DDR PLL general status register (Clocking_PLLDGSR)

The PLLDGSR register provides information regarding the DDR PLL configuration.

Address: E_1000h base + C20h offset = E_1C20h



Clocking_PLLDGSR field descriptions

Field	Description
0 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. 0 PLL is active. 1 PLL is disabled.
1–22 -	This field is reserved. Reserved
23–30 CFG	Reflects the current PLL multiplier configuration. Indicates the frequency for this PLL.
31 -	This field is reserved. Reserved

4.6 Functional Description

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

4.6.1 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. The chip asserts HRESET_B throughout the power-on reset process, including RCW configuration. Reset and RCW configuration time varies depending on the configuration source and SYSCLK frequency. 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, HRESET_B is released by the device and the clocking unit begins distributing PLL outputs throughout the device. Pre-boot initialization is then optionally performed, and the cores are permitted to boot.

The detailed POR sequence for the device is as follows:

1. The external system logic asserts PORESET_B and power is applied to comply with the chip's data sheet .

2. 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, Freescale 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.

3. The system applies a toggling SYSCLK signal and stable POR configuration inputs. At this point, SYSCLK is propagated throughout the device; the platform PLL is running in bypass mode.
4. The device begins driving HRESET_B asserted after sampling the assertion of PORESET_B.
5. External system logic negates PORESET_B after its required hold time and after POR configuration inputs have been valid for their required setup times.
6. The device samples the RCW source POR configuration inputs (cfg_rcw_src[0:*n*]) on deassertion of PORESET_B to determine the RCW source. Note that the POR configuration inputs are sampled only on a PORESET_B.
7. The device initiates and completes reset of the rest of the platform logic. Note that this platform reset step is the point where the device hard reset process (HRESET_B) begins if an external device asserts HRESET_B (assuming the device is not already sequencing through the power-on reset process).
8. 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[0:*n*]. All of the DDR I/Os become enabled at this point (though MCKE, MCK, MODT are enabled from the beginning). The ASLEEP signal is also enabled at this point.
9. If the IFC's NAND Flash interface is configured as the RCW source, the reset block instructs the IFC to load a boot block from Flash into the internal buffer RAM of the IFC. Once complete, the reset block proceeds to instruct the Pre-Boot Loader to begin reading in RCW data. Note that if the IFC NAND Flash interface reports an ECC error, the device reset sequence is halted indefinitely, waiting for another PORESET_B or hard reset.

10. The pre-boot loader (PBL) starts loading the RCW data from the interface specified by cfg_rcw_src[0:*n*] configuration inputs and stores that 64 bytes of data to the RCWSR registers within the device configuration block. Loading time varies and depends on the source of the RCW. for more information).
11. The PLLs begin to lock.
12. 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 or hard reset.
13. The platform clock tree is then switched over and is driven by the output of the platform PLL.
14. The device stops driving HRESET_B at this point. All other I/O drivers are enabled at this point.
15. If the IFC's NAND Flash interface is:
    - configured as the pre-boot initialization source

      OR

    - the boot device target AND not fused as secure boot

      AND

    - the IFC's NAND Flash interface was NOT previously used as the RCW source,

      then the reset block informs the IFC to load a boot block from Flash into the internal buffer RAM of the IFC. Once complete, the IFC signals back to the reset block, and the reset block can proceed. Note that if the IFC reports an ECC error, the device reset sequence is halted indefinitely, waiting for a hard reset or PORESET_B.
16. The PBL performs Pre-Boot Initialization, if enabled by RCW, by reading data from either the eSDHC, SPI, I^2C1, or IFC interface and writing to CCSR space or local memory space (SRAM, DDR). If the PBL reports an error during its pre-boot initialization process, the device reset sequence is halted indefinitely, waiting for a hard reset or PORESET_B. Note that non-CCSR-mapped memory space of a module on OCN must not be the target of any pre-boot initialization.
17. Any external device optionally driving HRESET_B negates it if not done earlier. If other external devices do not release HRESET_B, the device reset sequence stalls at this point.
18. System ready state. The peripheral interfaces (serial RapidIO and PCI Express) are released to accept external requests, and the boot vector fetches by the cores 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 negates synchronized to a rising edge of SYSCLK, 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 SRDSBnRSTCTL[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.

Figure below shows a timing diagram of the POR sequence.

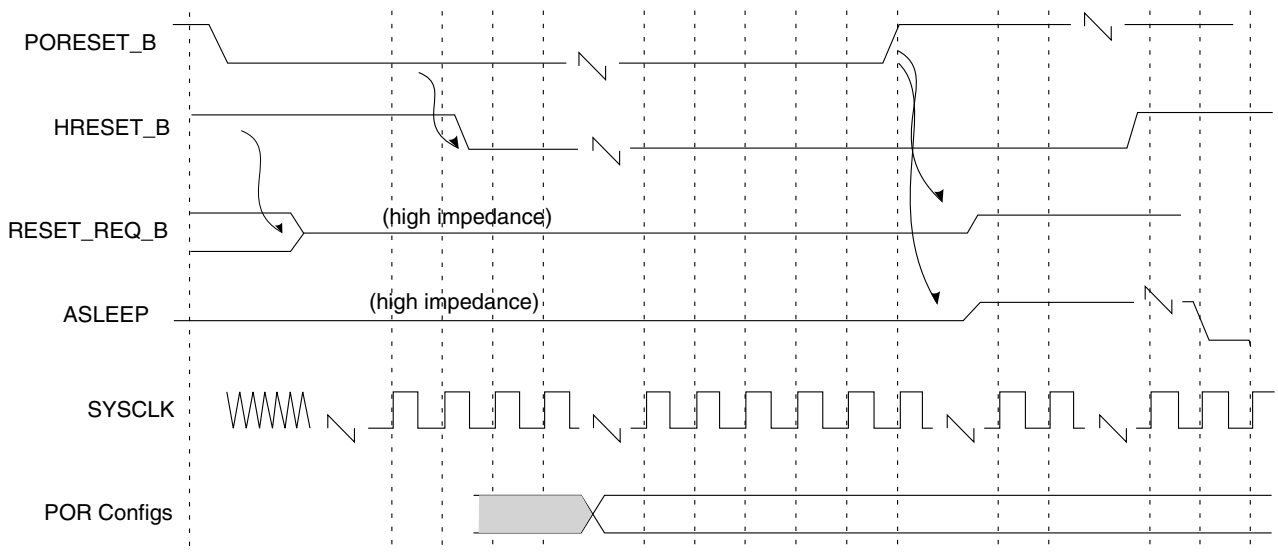


Figure 4-22. Power-On Reset Sequence

4.6.2 Hard Reset Sequence

The hard reset sequence is initiated by the external system asserting HRESET_B. Upon sampling the HRESET_B asserted, the device then begins driving HRESET itself throughout the hard reset state.

Reset and RCW configuration time varies subject to the configuration source and SYSCLK frequency. The reset configuration input signals are not sampled by a hard reset (only a power-on reset), so the device immediately begins loading RCW data and configures the device as explained in Reset Configuration Word (RCW). After the configuration sequence completes, the device releases the HRESET signal and exits the

hard reset state. After negation is detected, a 16-cycle period is taken before testing for the presence of an external reset. The hard reset sequence begins with reset of the device at step 7 in Power-On Reset Sequence.

4.6.3 Power-On Reset Configuration

Various device functions are initialized by sampling certain signals during power on reset. The values of all these signals are sampled into registers when PORESET_B is deasserted. These inputs are to be pulled high or low by external resistors. 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.

4.6.3.1 Reset configuration word (RCW) source

The reset configuration word (RCW) source inputs, cfg_rcw_src[0:8], are multiplexed on {IFC_AD[8:15], IFC_CLE}. These configuration inputs select the source for the RCW data as shown in the following table. The encoded values latched on these signals during POR are accessible in PORSR1[RCW_SRC], as described in PORSR1.

Table 4-34. RCW source encodings

cfg_rcw_src [0]_[1:4]_[5:8] value	RCW source
0_0000_xxxx	Reserved
0_0001_xxxx	8-bit NOR Flash cfg_rcw_src[5]: 0 Address shift “left” (most significant bits are IFC_AD[0:n-1] 1 Address shift “right” (least significant bits are IFC_AD[0:n-1]

Table continues on the next page...

Table 4-34. RCW source encodings (continued)

cfg_rcw_src [0]_[1:4]_[5:8] value	RCW source
	cfg_rcw_src[6:7]: 00 Shift left by 10 OR shift right by 10 (depending on the value of cfg_rcw_src[5]/ IFC_MODE[5] to provide 22b addressability. Note that this option is not valid for pinouts supporting 32b of addressability. 01 Shift left by 7 OR shift right by 7 (depending on the value of cfg_rcw_src[5]/ IFC_MODE[5]) to provide up to 25b addressability. . 10 Shift left by 4 OR shift right by 4 (depending on the value of cfg_rcw_src[5]/ IFC_MODE[5]) to provide up to 28b addressability. 11 Shift left by 0 OR shift right by 0 (depending on the value of cfg_rcw_src[5]/ IFC_MODE[5]]) to provide up to 32b addressability. cfg_rcw_src[8]: 0 CS before ALE (supports internal latch based asynchronous NOR devices) 1 ALE before CS (supports simple asynchronous NOR devices)
0_0010_xxxx	16-bit NOR Flash cfg_rcw_src[5:8] encodings are the same as those described for 0_0001_xxxx (8-bit NOR Flash)
0_0011_xxxx	32-bit NOR Flash cfg_rcw_src[5:8] encodings are the same as those described for 0_0001_xxxx (8-bit NOR Flash)
0_0100_0000[1]	SD/MMC (eSDHC)
0_0100_001-0_0100_0011	Reserved
0_0100_0100[1]	SPI 16-bit addressing
0_0100_0101[1]	SPI 24-bit addressing
0_0100_011x	Reserved
0_0100_1000[1]	I2C1 normal addressing (supports ROMs up to 256 bytes)
0_0100_1001[1]	I2C1 extended addressing
0_0100_1010-0_0111_1111	Reserved
0_100x_xxxx[1]	Reserved
0_1010_0000-0_1111_1111	Reserved
1_0000_00xx	8-bit NAND Flash, 512-byte page, 32 pages/block cfg_rcw_src[7]: 0 Bad Block Indicator in page 0/1 1 Bad Block Indicator in page 0 or last page cfg_rcw_src[8]: 0 ECC disabled 1 4 bits per 520 bytes
1_0000_01xx	8-bit NAND Flash, 2 KB page, 64 pages/block cfg_rcw_src[7:8] encodings are the same as those described for 1_0000_00xx (8-bit NAND Flash, 512-byte page, 32 pages/block)
1_0000_10xx	8-bit NAND Flash, 2 KB page, 128 pages/block

Table continues on the next page...

Table 4-34. RCW source encodings (continued)

cfg_rcw_src [0]_[1:4]_[5:8] value	RCW source
	cfg_rcw_src[7:8] encodings are the same as those described for 1_0000_00xx (8-bit NAND Flash, 512-byte page, 32 pages/block)
1_0000_11xx	Reserved
1_0001_xxxx	8-bit NAND Flash, 4 KB page, 128 pages/block cfg_rcw_src[5]: 0 Bad Block Indicator in page 0/1 1 Bad Block Indicator in page 0 or last page cfg_rcw_src[6:8]: 000 ECC disabled 001 4 bits per 520 byte sector 010-100 Reserved 101 8 bits per 528 byte sector 110 24 bits per 1 KB sector 111 40 bits per 1 KB sector
1_0010_xxxx	8-bit NAND Flash, 4 KB page, 256 pages/block cfg_rcw_src[5:8] encodings are the same as those described for 1_0001_xxxx (8-bit NAND Flash, 4 KB page, 128 pages/block)
1_0011_xxxx	8-bit NAND Flash, 4 KB page, 512 pages/block cfg_rcw_src[5:8] encodings are the same as those described for 1_0001_xxxx (8-bit NAND Flash, 4 KB page, 128 pages/block)
1_0100_xxxx	8-bit NAND Flash, 8 KB page, 128 pages/block cfg_rcw_src[5:8] encodings are the same as those described for 1_0001_xxxx (8-bit NAND Flash, 4 KB page, 128 pages/block)
1_0101_xxxx	8-bit NAND Flash, 8 KB page, 256 pages/block cfg_rcw_src[5:8] encodings are the same as those described for 1_0001_xxxx (8-bit NAND Flash, 4 KB page, 128 pages/block)
1_0110_xxxx	8-bit NAND Flash, 8 KB page, 512 pages/block cfg_rcw_src[5:8] encodings are the same as those described for 1_0001_xxxx (8-bit NAND Flash, 4 KB page, 128 pages/block)
1_0111_xxxx	Reserved
1_1000_00xx	16-bit NAND Flash, 512-byte page, 32 pages/block cfg_rcw_src[7:8] encodings are the same as those described for 1_0000_00xx (8-bit NAND Flash, 512-byte page, 32 pages/block)
1_1000_01xx	16-bit NAND Flash, 2 KB page, 64 pages/block cfg_rcw_src[7:8] encodings are the same as those described for 1_0000_00xx (8-bit NAND Flash, 512-byte page, 32 pages/block)
1_1000_10xx	16-bit NAND Flash, 2 KB page, 128 pages/block cfg_rcw_src[7:8] encodings are the same as those described for 1_0000_00xx (8-bit NAND Flash, 512-byte page, 32 pages/block)
1_1000_11xx	Reserved

Table continues on the next page...

Table 4-34. RCW source encodings (continued)

cfg_rcw_src [0]_[1:4]_[5:8] value	RCW source
1_1001_xxxx	16-bit NAND Flash, 4 KB page, 128 pages/block cfg_rcw_src[5:8] encodings are the same as those described for 1_0001_xxxx (8-bit NAND Flash, 4 KB page, 128 pages/block)
1_1010_xxxx	16-bit NAND Flash, 4 KB page, 256 pages/block cfg_rcw_src[5:8] encodings are the same as those described for 1_0001_xxxx (8-bit NAND Flash, 4 KB page, 128 pages/block)
1_1011_xxxx	16-bit NAND Flash, 4 KB page, 512 pages/block cfg_rcw_src[5:8] encodings are the same as those described for 1_0001_xxxx (8-bit NAND Flash, 4 KB page, 128 pages/block)
1_1100_xxxx	16-bit NAND Flash, 8 KB page, 128 pages/block cfg_rcw_src[5:8] encodings are the same as those described for 1_0001_xxxx (8-bit NAND Flash, 4 KB page, 128 pages/block)
1_1101_xxxx	16-bit NAND Flash, 8 KB page, 256 pages/block cfg_rcw_src[5:8] encodings are the same as those described for 1_0001_xxxx (8-bit NAND Flash, 4 KB page, 128 pages/block)
1_1110_xxxx	16-bit NAND Flash, 8 KB page, 512 pages/block cfg_rcw_src[5:8] encodings are the same as those described for 1_0001_xxxx (8-bit NAND Flash, 4 KB page, 128 pages/block)
1_1111_xxxx	Reserved

1. Not valid as an IFC_MODE encoding

4.6.3.2 General-purpose input

The general-purpose inputs listed in this table are available for application-specific use. The encoded values latched on these signals during POR are accessible in GPPORCR1[POR_CFG_VEC], as described in General-Purpose POR Configuration Register (DCFG_CCSR_GPPORCR1).

Table 4-35. General-purpose input

Functional signals	Reset configuration name	Value (binary)	General-purpose input
IFC_AD[0:7] Default (1111_1111)	cfg_gpinput[0:7]	all	Application-defined

4.6.3.3 DRAM type select

The DRAM type select input, described in this table, specifies the DDR technology and, thus, voltage (GV_{DD}) to be used with the DDR memory controllers. The encoded value latched on this signal during POR is accessible in PORSR2[DRAM_TYPE], as described in POR Status Register 1 (DCFG_CCSR_PORSR1).

Table 4-36. DRAM type select

Functional signals	Reset configuration name	Value (binary)	DRAM type
IFC_A[21]	cfg_dram_type	0	DDR3 technology (1.5 V)
Default (1)		1	DDR3L technology (1.35 V)

4.6.3.4 SerDes XVDD voltage select

The XVDD type select input, described in this table, specifies the XVDD voltage to be used with the chip. The encoded value latched on this signal during POR is accessible in PORSR2[XVDD_SEL], as described in POR Status Register 2 (DCFG_CCSR_PORSR2).

Table 4-37. SerDes XVDD voltage select

Functional signals	Reset configuration name	Value (binary)	XVDD voltage
ASLEEP Default (1)	cfg_xvdd_sel	0	Reserved
		1	1.35 V

4.6.3.5 IFC external transceiver enable polarity select

The IFC external transceiver enable polarity select input, described in this table, specifies the polarity of the IFC_TE output signal to accommodate external transceivers with either active-high or active-low enable inputs. The encoded value latched on this signal during POR is accessible in PORSR1[IFC_TE]. See External transceiver enable (TE) for more information.

Table 4-38. IFC external transceiver enable polarity select

Functional signals	Reset configuration name	Value (binary)	IFC external transmitter polarity
IFC_TE	cfg_ifc_te	0	IFC drives logic 1 for TE assertion
Default (1)		1	IFC drives logic 0 for TE assertion

4.6.4 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[0:8]-see Reset Configuration Word Source for more information). See Pre-Boot Loader 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 logic involved is clocked directly from SYSCLK since RCW importing takes place before on-chip PLLs are configured.

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 (see Reset Control Word Status Register n (DCFG_CCSR_RCWSR*n*)) contained in the Device Configuration and Pin Control module , after which the device is configured as specified in the RCW. Required Format of Data Structure Consumed by PBL provides details of the data structure that is required to reside in non-volatile memory.

4.6.4.1 RCW Field Definitions

Table 1 describes the function of the individual bits of the 512-bit (64-byte) RCW data structure.

NOTE

Unless noted otherwise, any bit ranges in Table 1 listed as reserved must be populated with 0.

Table 4-39. 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 All valid platform PLL frequencies. 01-11 Reserved

Table continues on the next page...

Table 4-39. RCW Field Descriptions (continued)

Bit(s) (of 0-511)	Field Name	Description	Notes/comments
2-6	SYS_PLL_RAT	System PLL multiplier/ratio	This field selects the platform clock:SYSCLK ratio. Options: 0_0000 Reserved 0_0011 3:1 0_0100 4:1 0_0101 5:1 0_0110 6:1 0_0111 7:1 0_1000 8:1 0_1001 9:1 0_1010 10:1 0_1011 11:1 0_1100 12:1 **NOTE:** All ratios may not be supported due to frequency restrictions. Refer to the hardware specifications for the supported frequencies.
7	Reserved		
8-9	MEM_PLL_CFG	Memory controller complex PLL configuration.	This field configures the memory complex PLLs for the frequency of the reference clock applied. Options: 00 All valid DDR PLL frequencies. All others Reserved
10-15	MEM_PLL_RAT	Memory controller complex PLL multiplier/ratio (applies to DDR controller). **NOTE:** Note that only asynchronous mode is supported for the memory controller complex.	This field selects the DDR data rate to DDRCLK ratio The MEM_PLL_CFG field must be configured to use the appropriate reference clock frequency option (higher or lower) for the cutoff frequency of the selected ratio. Refer to the hardware specifications for the specific cutoff reference clock frequency for each ratio. Any reference clock frequency applied at or below the stated cutoff point requires configuring MEM_PLL_CFG for a lower frequency reference clock. Any reference clock frequency applied above the stated cutoff point requires configuring MEM_PLL_CFG for a higher frequency reference clock. See the MEM_PLL_CFG field definition for more details. Options: 00_1000 8:1 00_1001 9:1 00_1010 10:1

Table continues on the next page...

Table 4-39. RCW Field Descriptions (continued)

Bit(s) (of 0-511)	Field Name	Description	Notes/comments
			00_1011 11:1 00_1100 12:1 00_1101 13:1 00_1110 14:1 00_1111 15:1 01_0000 16:1 01_0001 17:1 01_0010 18:1 01_0011 19:1 01_0100 20:1 All other encodings are reserved. **NOTE:** Not all ratios are supported due to frequency restrictions. Refer to the hardware specifications for the supported frequencies.
16-23	Reserved		
24-25	CGA_PLL1_CFG	Cluster group A PLL 1 configuration.	Options: 00 All valid cluster group A PLL frequencies. All others Reserved
26-31	CGA_PLL1_RAT	Cluster group A PLL 1 multiplier/ratio.	Options: 00_0101 through 10_1000 5:1 through 30:1 Async Mode All other encodings are reserved. **NOTE:** Not all ratios are supported due to frequency restrictions. Refer to the hardware specifications for the supported frequencies.
32-33	CGA_PLL2_CFG	Cluster group A PLL 2 configuration.	Options: 00 All valid cluster group A PLL frequencies. All others Reserved
34-39	CGA_PLL2_RAT	Cluster group A PLL 2 multiplier/ratio.	Options: 00_0101 through 10_1000 5:1 through 30:1 Async All other encodings are reserved. **NOTE:** Not all ratios are supported due to frequency restrictions. Refer to the hardware specifications for the supported frequencies.
40-95	Reserved		
96-99	C1_PLL_SEL	Cluster 1 PLL select. Selects one of two available cluster group A PLLs.	Options: 0000 CGA_PLL1 /1

Table continues on the next page...

Table 4-39. RCW Field Descriptions (continued)

Bit(s) (of 0-511)	Field Name	Description	Notes/comments
			0001 CGA_PLL1 /2 0010 CGA_PLL1 /4 0100 CGA_PLL2 /1 0101 CGA_PLL2 /2 0110 CGA_PLL2 /4 All other encodings are reserved.
100-127	Reserved		
SerDes PLL and Protocol Configuration (Bits 128-183)			
128-135	SRDS_PRTCL_S1	SerDes protocol select - SerDes 1	This field must always be used in conjunction with SRDS_PRTCL_S2. See "Networking Protocols (SerDes 1 and SerDes 2)" in the SerDes Module chapter for a complete list of the options and the definitions of this encoded field.
136-143	SRDS_PRTCL_S2	SerDes protocol select - SerDes 2	This field must always be used in conjunction with SRDS_PRTCL_S1. See "Networking Protocols (SerDes 1 and SerDes 2)" in the SerDes Module chapter for a complete list of the options and the definitions of this encoded field. **NOTE:** DMA3 is not available for use in any configuration containing SRIO2.
144-157	Reserved		
158	FM_MAC_RAT	FM-to-MAC Ratio	Controls the ratio between the main portion of FM and its MACs and PCS: 0b0 - FM runs at 2x Frequency of the MAC 0b1 - FM runs at 1x Frequency of the MAC
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. Bit 160 corresponds to SerDes 1, PLL1 (lanes A-H); bit 161 corresponds to SerDes 1, PLL2 (lanes C-H). Options: Selection for protocols operating at 1.25 or 2.5 or 5.0GHz: 0 - 100MHz 1 - 125MHz Selection for protocols operating at 3.0GHz 0 - 100MHz 1 - 125MHz Selection for protocols operating at 3.125 or 6.25GHz 0 - 125MHz 1 - 156.25MHz Selection for protocols operating at 3.75GHz

Table continues on the next page...

Table 4-39. RCW Field Descriptions (continued)

Bit(s) (of 0-511)	Field Name	Description	Notes/comments
			0 - 125MHz 1 - 156.25MHz Selection for protocols operating at 10.3125GHz: 0 - 156.25MHz **NOTE:** The higher or lower reference clock frequency depends on the protocol and its operating frequency.
162-163	SRDS_PLL_REF_CLK_SEL_S2	SerDes PLL reference clock select - SerDes 2	This field selects the PLL reference clock frequency for SerDes 2. For defintion see SRDS_PLL_REF_CLK_SEL_S1 Bit 162 corresponds to SerDes 2, PLL1 (lanes A-H) Bit 163 corresponds to SerDes 2, PLL2 (lanes C-H).
164-167	Reserved		
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 Bit 169 corresponds to SerDes 1, PLL2. Option for each bit 0- PLL is not powered down 1- PLL is powered down If this field is set to 0, it informs RESET_REQ logic that it should assert RESET_REQ if a proper reference clock isn't applied or the SerDes PLL does not lock. If this field is set to 1, it forces any data lanes selected by the protocols defined in SRDS_PRTCL_S1 that would have been used by this PLL to be powered down as well. For example, if SRDS_PRTCL_S1 selects a protocol which assigns lanes A-D to PLL1 and lanes E-H to PLL2, if PLL2's RCW bit is set to powered down, then lanes E-H are powered down. This field is ignored if the respective SRDS_PRTCL_S1 field does not use a given Serdes PLL.
170-171	SRDS_PLL_PD_S2	SerDes PLL power down - SerDes 2	For defintion see SRDS_PLL_PD_S1 This field is used to power down the SerDes 2 PLLs. Bit 170 corresponds to SerDes 2, PLL1 Bit 171 corresponds to SerDes 2, PLL2.
172-175	Reserved		

Table continues on the next page...

Table 4-39. RCW Field Descriptions (continued)

Bit(s) (of 0-511)	Field Name	Description	Notes/comments
176-177	SRDS_DIV_PEX_S1	SerDes frequency divider - SerDes 1, PCI Express	Controls the frequency of PCI-Express protocols in this Serdes that are operating 5/2.5G or 8/5/2.5G.Lanes supporting other frequencies and protocols are unaffected by this field. Options: 00-Can train up to a max rate of 8G (5G if this is the max rate supported by this PCIe for the selected protocol) 01 -Can train up to a max rate of 5G 10-Can train up to a max rate of 2.5G 11 - Reserved
178	SRDS_DIV_SRIO_S2	SerDes frequency divider - SerDes , Serial RapidIO	This field configures the frequency of serial RapidIO protocols on SerDes 2. Lanes that are supporting other protocols and frequencies are unaffected by this field. Options: 0 5G 1 2.5G
179	Reserved		
180	SRDS_DIV_AURORA_S2	SerDes frequency divider - SerDes 2, Aurora	This field configures the frequency of Aurora protocols on SerDes 2. Lanes that are supporting other protocols and frequencies are unaffected by this field. Options: 0 5G 1 2.5G
181-182	SRDS_DIV_PEX_S2	SerDes frequency divider - SerDes 2, PCI Express	This field configures the frequency of PCI Express protocols on SerDes 2. Lanes that are supporting other protocols and frequencies are unaffected by this field. Options: 00 8G (or 5G if that is the maximum for the corresponding protocol selection) 01 5G 10 2.5G 11 Reserved
183	Reserved		
Miscellaneous PLL-Related Configuration (Bits 184-191)			
184-191	Reserved		
Boot Configuration (Bits 192-223)			

Table continues on the next page...

Table 4-39. RCW Field Descriptions (continued)

Bit(s) (of 0-511)	Field Name	Description	Notes/comments
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 restrictions apply: • RCW and pre-boot initialization data must be loaded from the same non-volatile memory device • At most, only one given IFC option can be used for cfg_rcw_src, PBI_SRC, and BOOT_LOC. This does not mean that BOOT_LOC cannot be different from cfg_rcw_src and PBI_SRC (provided the second restriction is observed). For example, the device can be configured to load RCW and pre-boot initialization data from NAND Flash (IFC) and boot from DDR. However, the device cannot load RCW and pre-boot initialization data from NAND Flash (IFC) and boot from NOR Flash (IFC). Also, the hard-coded RCW source options are not considered their own memory interface for this purpose. Options: 0000 I^2C1 normal addressing (up to 256 byte ROMs) 0001 I^2C1 extended addressing 0100 SPI 16-bit addressing 0101 SPI 24-bit addressing 0110 SD/MMC 1110 IFC (The RCW field IFC_MODE configures the IFC, provided the IFC has not already been configured by the cfg_rcw_src configuration input signals, which have precedence.) 1111 disabled All other encodings are reserved.
196-200	BOOT_LOC	Boot location. This is the initial location that the core fetches from.	Note the restrictions in PBI_SRC above. Specifically, the second restriction that at most, only one given IFC option can be selected by cfg_rcw_src, PBI_SRC, and BOOT_LOC. Options: 0_0000 PCIe1 0_0001 PCIe2 0_0010 PCIe3 0_0011 PCIe4 0_1000 SRIO 1 0_1001 SRIO 2 0_1010 - Reserved 0_1011 - Reserved

Table continues on the next page...

Table 4-39. RCW Field Descriptions (continued)

Bit(s) (of 0-511)	Field Name	Description	Notes/comments
			1_0000 Memory complex 1_0001 Reserved 1_0010 Reserved 1_0100 Reserved 1_1000 IFC (The RCW field IFC_MODE configures the IFC, provided the IFC has not already been configured by the cfg_rcw_src configuration input signals, which have precedence.) All other encodings are reserved.
201	BOOT_HO	Boot hold off mode	Options: 0 All cores except core 0 in hold off 1 All cores in hold off
202	SB_EN	Secure boot enable	Note that secure boot is enabled if either this RCW bit is set or the Intent to Secure fuse value is set. Options: 0 Secure boot is not enabled 1 Secure boot is enabled; boot from internal boot ROM
203-211	IFC_MODE	Integrated Flash controller mode	When PBI_SRC and/or BOOT_LOC are configured for IFC, this field selects the IFC mode for either pre-boot initialization or as the initial boot target of the core. Note that cfg_rcw_src has precedence over configuring the IFC. Valid IFC_MODE encodings are a subset of the cfg_rcw_src encodings. See Reset configuration word (RCW) source for the valid encodings for this field.
212-223	Reserved		
Clocking Configuration (Bits 224-255)			
224-226	HWA_CGA_M1_CLK_SEL	Hardware accelerator block (FM), cluster group A, mux 1 clock select.	Allows for the pattern matching engine (PME) frequency to be maximized (using asynchronous clock) by leveraging cluster group A PLL 1 or 2 as the source clock. Options: 000 Reserved 001 Asynchronous mode - Cluster group A PLL 1 / 1 010 Asynchronous mode - Cluster group A PLL 1 / 2 011 Asynchronous mode - Cluster group A PLL 1 / 3 100 Asynchronous mode - Cluster group A PLL 1 / 4

Table continues on the next page...

Table 4-39. RCW Field Descriptions (continued)

Bit(s) (of 0-511)	Field Name	Description	Notes/comments
			101 Platform clock 110 Asynchronous mode - Cluster group A PLL 2 / 2 111 Asynchronous mode - Cluster group A PLL 2 / 3
227-229	Reserved		
230-231	DRAM_LAT	DDR latency.	Options: 00 6-6-6 or 7-7-7 DRAMs 01 8-8-8, 9-9-9, 10-10-10, 11-11-11, or higher latency DRAMs 10 Reserved 11 5-5-5 DRAMs **NOTE:** If DDR latency is not known at the time the RCW is loaded, it is acceptable to conservatively configure this field as 01. This field is used for optimizing the DDR interface. No functional issues result if this field does not match the latency of the actual DRAM's used.
232	DDR_RATE	DDR data rate.	Reserved. Must be 0.
233	Reserved		
234-255	Reserved		
Memory and High-Speed I/O Configuration (Bits 256-287)			
256-259	Reserved		
260-262	RIO_DEVICE_ID	RapidIO device ID	See the description of BDIDCSR[BDID] in the Serial RapidIO controller memory mapped registers for more information.
263	RIO_SYS_SIZE	RapidIO system size	See the description of PEFCAR[CTLS] in the Serial RapidIO controller memory mapped registers for more information. Options: 0 Small system size (256 devices) 1 Large system size (65536 devices)
264-266	HOST_AGT_PEX	Host/agent PEX. Configures Host/Agent mode for all PCIe Interfaces..	000 All PCIe*n* in RC mode 001 All PCIe*n* in EP mode 010 PCIe1 and PCIe3 agent mode, rest in host mode 011 PCIe2 and PCIe3 agent mode, rest in host mode. 100 PCIe1 in agent mode, rest in host mode 101 PCIe2 in agent mode, rest in host mode. 110 PCIe3 in agent mode, rest in host mode

Table continues on the next page...

Table 4-39. RCW Field Descriptions (continued)

Bit(s) (of 0-511)	Field Name	Description	Notes/comments
			111 PCIe4 in agent mode, rest in host mode
267	HOST_AGT_SRIO	Host/agent SRIO. Configures Host (Root Complex) or Agent (Endpoint) modes for all serial RapidIO interfaces.	Options: 0 All host mode 1 All agent mode
268	RIO_RESPOND_ONLY	RapidIO respond-only mode. Configures respond-only mode for all serial RapidIO interfaces	Options: 0 All SRIO interfaces are enabled by default to issue requests into the system. 1 All SRIO interfaces are not enabled by default to issue requests into the system
269-287	Reserved		
General Purpose Information (Bits 288-319)			
288-319	GP_INFO	General purpose information. This field has no affect on functional logic; it may be used by software.	
320-351	Reserved		
Group A Pin Configuration (Bits 352-383)			
352-356	Reserved		
357-359	IRQ_EXT	Together with IRQ_BASE field, this field configures functionality of IRQ[3:11] pins	Options: 0b000 - IRQ[3:11] or GPIO_1[23:31] as defined by IRQ_BASE 0b001 - {SDHC_VS, IRQ[4:9] or GPIO1[24:29] (as defined by IRQ_BASE), IRQ[11] or GPIO1[31] (as defined by IRQ_BASE)}
360-362	SPI_EXT	Together with SPI_BASE field, this field configures functionality of the SPI pins	Options 000 - See SPI_BASE for defintion 001 - SPI_CS_B[0:2] or GPIO2[0:2] or SDHC_DAT[4:6] (as defined by SPI_BASE),
363-365	Reserved		
366-368	UART_BASE	This field configures the functionality of the UART pins: {UART1_SOUT, UART1_SIN, UART1_RTS_B, UART1_CTS_B, UART2_SOUT, UART2_SIN, UART2_RTS_B, UART2_CTS_B}	Options: 000 {GPIO1[15], GPIO1[17], GPIO1[19], GPIO1[21], GPIO1[16], GPIO1[18], GPIO1[20], GPIO1[22]} 011 {UART1_SOUT, UART1_SIN, GPIO1[19], GPIO1[21], GPIO1[16], GPIO1[18], GPIO1[20], GPIO1[22]} 100 {UART1_SOUT, UART1_SIN, UART1_RTS_B, UART1_CTS_B, GPIO1[16], GPIO1[18], GPIO1[20], GPIO1[22]} 101 {UART1_SOUT, UART1_SIN, GPIO1[19], GPIO1[21], UART2_SOUT, UART2_SIN, GPIO1[20], GPIO1[22]} 110 {UART1_SOUT, UART1_SIN, UART1_RTS_B, UART1_CTS_B, UART2_SOUT, UART2_SIN, UART2_RTS_B, UART2_CTS_B}

Table continues on the next page...

Table 4-39. RCW Field Descriptions (continued)

Bit(s) (of 0-511)	Field Name	Description	Notes/comments
			111 {UART1_SOUT, UART1_SIN, UART3_SOUT, UART3_SIN, UART2_SOUT, UART2_SIN, UART4_SOUT, UART4_SIN}
369	ASLEEP	This field configures the functionality of the ASLEEP pin.	Options: 0 ASLEEP 1 GPIO1[13]
370	RTC	This field configures the functionality of the RTC pin.	Options: 0 RTC 1 GPIO1[14]
371	SDHC_BASE	This field configures the functionality of the SDHC pins: {SDHC_CMD, SDHC_DAT[0:3], SDHC_CLK}	Options: 0 {SDHC_CMD, SDHC_DAT[0:3], SDHC_CLK} 1 GPIO2[4:9] **NOTE:** If cfg_rcw_src selects SD/MMC as the RCW source, the SDHC pins are driven with SDHC functionality regardless of the setting of this field.
372	IRQ_OUT	This field configures the functionality of the IRQ_OUT_B pin.	Options: 0 IRQ_OUT_B 1 EVT_B[9] **NOTE:** EVT_B[9] functionality, if selected by this field, defaults to input functionality until software reconfigures to output functionality.
373-381	IRQ_BASE	This field configures the functionality of IRQ[3:11] pins--the corresponding GPIOs for these pins are GPIO1[23:31].	Options for each bit: 0 IRQ 1 GPIO
382-383	SPI_BASE	This field configures the functionality of the SPI_CS_B[0:3] pins.	Options: 00 SPI_CS_B[0:3] 01 SDHC_DAT[4:7] for 8-bit MMC card support 10 GPIO2[0:3] 11 Reserved **NOTE:** If cfg_rcw_src selects an SPI option as the RCW source, the SPI pins used for loading the RCW {SPI_MOSI, SPI_MISO, SPI_CLK, SPI_CS_B[0]} are driven with SPI functionality regardless of the setting of this field.
Group B Pin Configuration (Bits 384-415)			
384-404	Reserved		
405	IFC_GRP_E1_BASE	This field configures the functionality of the Group E1 IFC pins: {IFC_CS_B[1:3]}.	Options: 0 {IFC_CS_B[1:3]}

Table continues on the next page...

Table 4-39. RCW Field Descriptions (continued)

Bit(s) (of 0-511)	Field Name	Description	Notes/comments
			1 {GPIO2[10:12]}
406	IFC_GRP_E2_BASE	This field configures the functionality of the Group E2 IFC pins: {IFC_CS_B[4:7]}.	Options: 0 {IFC_CS_B[4:7]} 1 {GPIO1[9:12]}
407	IFC_GRP_D_BASE	This field configures the functionality of the Group D IFC pins: {IFC_PAR[0:1], IFC_PERR_B, IFC_PAR[2:3]}.	Options: 0 {IFC_PAR[0:1], IFC_PERR_B, IFC_PAR[2:3]} 1 {GPIO2[13:17]}
408	Reserved		
409	Reserved		
410-411	IFC_GRP_B_BASE	This field configures the functionality of the Group B IFC pins: {IFC_AD[28:31]}.	Options: 00 {IFC_AD[28:31]/ IFC_A[28:31]} 01 {GPIO2[28:31]} 10 {unused, IFC_RB_B[2:4]}
412-413	IFC_GRP_A_BASE	This field configures the functionality of the Group A IFC pins: {IFC_AD[25:27]}.	Options: 00 {IFC_AD[25:27]/ IFC_A[25:27]} 01 {GPIO2[25:271]} 10 {IFC_WP_B[1:3]}
414-415	Reserved		
SoC-Specific Configuration (Bits 416-447)			
416	1588	This field configures the functionality of the IEEE 1588 pins {TSEC_1588_CLK_IN, TSEC_1588_TRIG_IN1, TSEC_1588_TRIG_IN2, TSEC_1588_ALARM_OUT1, TSEC_1588_ALARM_OUT2, TSEC_1588_CLK_OUT, TSEC_1588_PULSE_OUT1, TSEC_1588_PULSE_OUT2}. The corresponding GPIOs for these pins are GPIO3[0:7].	Options: 0 GPIO3[0:7] 1 IEEE 1588
417-418	EC1	This field configures the functionality assigned to the EC1 parallel mode pins.	Options: 00 Frame Manager MAC3 RGMII 01 Reserved 10 GPIO3[11:23] 11 Reserved
419-420	EC2	This field configures the functionality assigned to the EC2 parallel mode pins.	Options: 00 Frame Manager MAC4 RGMII 01 Frame Manager MAC10 RGMII 10 GPIO3[24:31], GPIO4[27:31] 11 Reserved

Table continues on the next page...

Table 4-39. RCW Field Descriptions (continued)

Bit(s) (of 0-511)	Field Name	Description	Notes/comments
421-422	I2C3	This field configures the functionality of the I2C3 pins.	Options: 00 {IIC3_SCL, IIC3_SDA} 01 GPIO4[0:1] 10 Reserved 11 Reserved
423-424	I2C4	This field configures the functionality of the I2C4 pins.	Options: 00 {IIC4_SCL, IIC4_SDA} 01 GPIO4[2:3] 10 EVT_B[5:6] 11 Reserved **NOTE:** EVT_B[5:6] functionality, if selected by this field, defaults to input functionality until software reconfigures to output functionality.
425-443	Reserved		
444	DMA1	This field configures the functionality of the DMA1 pins.	Options: 0 {DMA1_DREQ0_B, DMA1_DACK0_B, DMA1_DDONE0_B} 1 {GPIO4[4:6]}
445	SDHC	This field configures the functionality of the SDHC pins.	Options: 0 {SDHC_CD_B, SDHC_WP} 1 {GPIO4[24:25]}
446-447	DMA2	This field configures the functionality of the DMA2 pins.	Options: 00 {DMA2_DREQ0_B, DMA2_DACK0_B, DMA2_DDONE0_B} 01 {GPIO4[7:9]} 10 {GPIO4[7], EVT_B[7:8]} **NOTE:** EVT_B[7:8] functionality, if selected by this field, defaults to input functionality until reconfigured by software to output functionality.
PLL and Clocking Configuration Expansion (Bits 448-511)			
448-508	Reserved		
509-511	HWA_CGA_M2_CLK_SEL	Hardware accelerator block, cluster group A, mux 2 clock select.	000 Reserved 001 Asynchronous mode - Cluster group A PLL 2 / 1 010 Asynchronous mode - Cluster group A PLL 2 / 2 011 Asynchronous mode - Cluster group A PLL 2 / 3

Table 4-39. RCW Field Descriptions

Bit(s) (of 0-511)	Field Name	Description	Notes/comments
			100 Asynchronous mode - Cluster group A PLL 2 / 4 101 Cluster clock 110 Asynchronous mode - Cluster group A PLL 1 / 2 111 Asynchronous mode - Cluster group A PLL 1 / 3

4.6.5 Clocking

The following sections describe the clocking within the chip.

4.6.5.1 IP Logic Clock Distribution and Configuration

The chip takes primary clocking input from the external SYSCLK signal. As shown in the figures below, the SYSCLK input (frequency) is multiplied using multiple phase locked loops (PLL) to create a variety of frequencies which can then be passed to a variety of internal logic, including cores and peripheral IP modules.

The DDR PLL is used to provide clocking to the DDR memory controller complex. The DDR PLL uses the DDRCLK input clock as a reference to create a unique DDR memory controller complex clock. The DDR complex operates asynchronously with respect to the platform clock.

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 figure describe 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 SerDes modules are clocked by dedicated SerDes reference clock inputs.

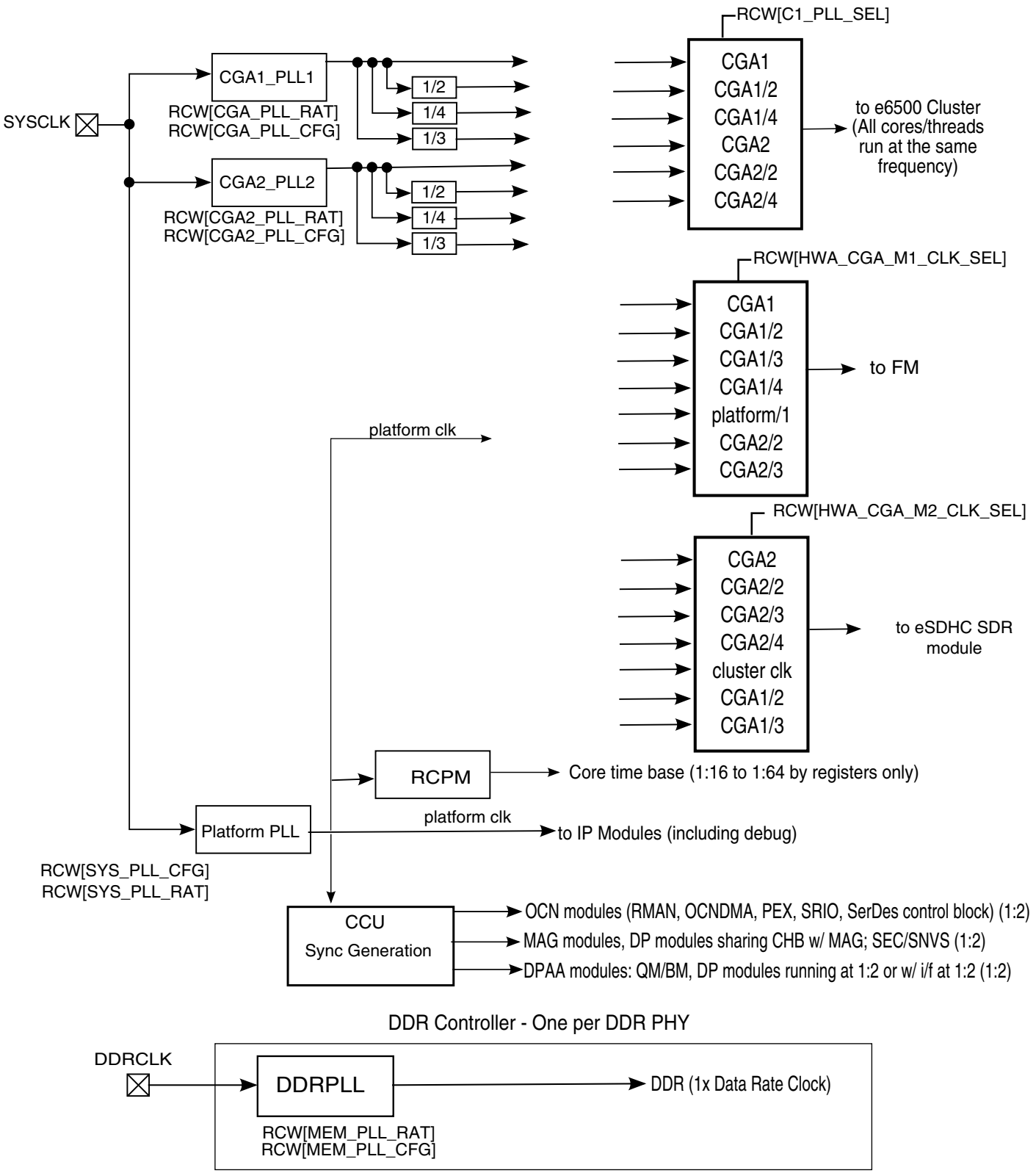


Figure 4-23. T2080 Logic Clock Subsystem Block Diagram

4.6.5.2 Dynamic frequency switching (DFS) control

Dynamic frequency switching (DFS) is supported for each core/hardware-accelerator target through software control. Each target can select from the clock frequencies generated by the PLL cluster which it is connected to. The targets can select from clocks generated by the local PLLs within the CGU or in some cases select from sources external to the CGU. Switching from one frequency to another must take place without introduction of any glitches in the clock path to each core. In order to accomplish this, a state machine is used to properly sequence the switching of clocks.

Each target's clock source is determined at POR by the RCW settings. Once the system is up and running, the CLKCnCSR[CLKSEL] register fields can be written to change each core's operating frequency on-the-fly. When the CLKSEL value for any core changes, the DFS logic will detect the change and sequence the clock switch such that no glitches are introduced. The active clock is always disabled first, then the new target clock source is enabled.

Note that the user MUST wait between writes to the CLKSEL field to allow enough time for the DFS sequence to complete. Back-to-back writes to the CLKSEL field that occur too quickly may result in unpredictable behavior.

4.6.6 Voltage ID

The overall performance boost provided by improved device integration, complexity, and a smaller transistor process technology comes with a penalty of higher leakage current. Leakage currents are present in circuitry independent of the specific speed. To guarantee maximum performance with minimum leakage current, a specific method of selecting the optimum voltage level is required to be implemented.

During manufacturing, Freescale determines the voltage required to achieve the target frequency bin and programs this information into on-chip Voltage ID (VID) efuses. The VID values are accessed through the fuse status register in the device configuration module (DCFG_FUSESR). Using the VID, initialization software can program the system's voltage regulator to the appropriate value. Abstracting the on-chip voltage ID data from the interface and programming requirements of the regulator allows flexibility in selecting a solution. The power supply design section in the hardware specifications provides examples of system design options.

Systems must use the VID to change the voltage regulators in the system using a reliable and safe methodology. At initial power on, the system regulator must come up at the default voltage. Then, very early in the boot process, initialization software reads the

value in DCFG_FUSESR and translates that value into a command sequence that will program the new voltage value for the regulator. Once the regulator has been sent the new values, allow the regulator time to change voltage and stabilize before enabling non-essential boot features and higher clock rates.

In some cases, changing voltages too fast can cause a fault or shutdown. On many recent model regulators, the voltage changes gradually and the time delay is programmable. However, some older regulators may need to be stepped by software to avoid over-current protection events.

Chapter 5
Pre-Boot Loader (PBL)

5.1 PBL Overview

The pre-boot loader (PBL) performs configuration register reads and writes to initialize the I^2C, IFC FCM (NAND Flash), eSDHC, or SPI interface, loads the RCW and pre-boot initialization commands from external memory device through I^2C, IFC FCM, IFC GPCM, eSDHC, or SPI and writes data to configuration registers or memory before the local cores are permitted to boot.

Figure 5-1 shows a block diagram of PBL, showing the functional organization of the module, which includes interface command modules (ICMs) for eSDHC, I2C1, eSPI, IFC, and a common control module.

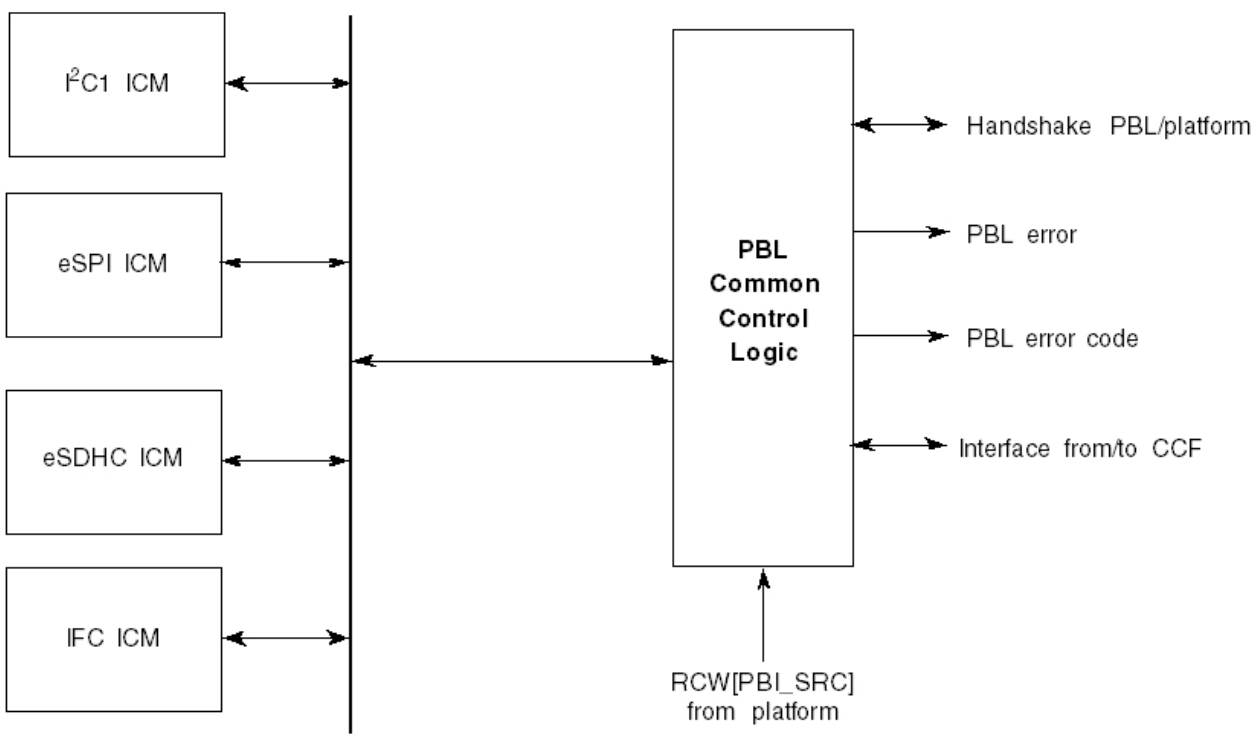


Figure 5-1. Pre-Boot Loader (PBL) Block Diagram

5.2 PBL Features Summary

- Initializes and loads RCW/pre-boot initialization commands from eSDHC SD/MMC interface
  - Supports multi-media card (MMC)
    - 1-bit data transfer on SD interface
    - Compliant with version 4.2 of the specification
    - Standard capacity (< 2 Gbytes)
    - High-capacity (> 2 Gbytes)
    - 1.8V, or 3.3 V (with external voltage translator)
  - Supports Secure Digital (SD) memory card
    - 1-bit data transfer on MMC interface
    - Compliant with version 2.00 of the specification
    - Standard capacity (< 2 Gbytes)
    - High-capacity (> 2 Gbytes)
    - 3.3 V (with external voltage translator)
- Initializes and load RCW/pre-boot initialization commands from I2C1 interface
- Initializes and load RCW/pre-boot initialization commands from IFC FCM/GPCM interface
- Initializes and load RCW/pre-boot initialization commands from eSPI 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-byte 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

5.3 PBL Modes of Operation

- Load both RCW and pre-boot initialization commands from individual interface.
- Load RCW only.
- Load pre-boot initialization commands only if default RCW is to be used.

5.4 PBL Functional Description

This section contains the following subsections:

5.4.1 Configuration of Device via 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 RCWSR for more information. The RCWSRs have specific 64-byte aligned address offsets that the PBL should ensure is written. Any address offset that is not that specific 64-byte aligned address offset is reported by the PBL as an error to the platform, which asserts the RESET_REQ output pin upon receiving the error from the PBL.

5.4.2 Device Initialization by PBL

The cfg_rcw_src configuration input determines which 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 I2C1, eSPI, eSDHC, or IFC FCM (NAND flash) interface before accessing the RCW or PBI commands. Note that IFC GPCM (NOR flash) does not require initialization by the PBL even though it may be selected as the source for 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 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.

5.4.2.1 CCSR Registers Blocked from PBL During Secure Boot

Table 5-1 lists the CCSR address ranges that are blocked from PBL during secure boot.

Table 5-1. CCSR Registers Blocked from PBL During Secure Boot

Address Range	Size	Description
0x00_0000	4 bytes	CSSBARH
0x00_0004	4 bytes	CSSBARL
0x00_0008	4 bytes	CCSRAR
0x02_0000-0x02_FFFF	64 Kbytes	PAMU partitions 1-16
0x0E_8000-0x0E_8FFF	4 Kbytes	Security fuse processor (SFP)
0x30_0000-0x30_FFFF	64 Kbytes	Security Engine (SEC)
0x31_4000- 0x31_4FFF	4 Kbytes	Security monitor

5.4.3 Required Format of Data Structure Consumed 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 5-2. Required Format of Data Structure Consumed 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							
	BYTE2							
	.....							
	BYTE63							
First Pre-Boot Initialization Command (optional)	ACS	BYTE_CNT						CONT=1
	SYS_ADDR[23-16]							
	SYS_ADDR[15-8]							

Table continues on the next page...

Table 5-2. Required Format of Data Structure Consumed by PBL (continued)

	0	1	2	3	4	5	6	7
	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 = 00100 (4 bytes)						CONT=0
	SYS_ADDR[23:16] = 0x13[2]							
	SYS_ADDR[15:8] = 0x80[2]							
	SYS_ADDR[7:0] = 0x40[2]							
	CRC0							
	CRC1							
	CRC2							
	CRC3							

1. SYS_ADDR for the RCW should point to the RCW status register (RCWSR[1-16]) in the device configuration and pin control block.
2. SYS_ADDR = 0x13_8040 is a PBI CRC check command (PBL block base address 0x13_8000 with offset 0x040). See Pre-Boot Initialization Command Loading by PBL for more information.

Table 5-3 provides field descriptions for the PBL data structure.

Table 5-3. 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 datum.
BYTE_CNT	Byte Count. Represents the number of bytes associated with this addr/data pair. Note that an encoding of all zeros represents 64 bytes. Only power of 2 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 CCSRBAR or Alt Config BAR (depending on value of ACS) and then concatentated with SYS_ADDR to form the full address associated with the command.
BYTEn	Byte number *n* where *n* may range from 0 up to 63. Byte 0 corresponds to addr 0 relative to the SYS_ADDR (starting address), byte 1 for addr 1..., and byte 63 for addr 63. Note the first command must be the RCW addr/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 $1 + x^1 + x^2 + x^4 + x^5 + x^7 + x^8 + x^{10} + x^{11} + x^{12} + x^{16} + x^{22} + x^{23} + x^{26} + x^{32}$

5.4.4 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 (RCWSR*n*) in the device configuration and pin control block.

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.

5.4.5 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 space, DDR, or 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 5-3 for the supported encodings. Note that if byte count is 1 or 2, 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.

5.4.6 Reserved Address Space Used as Internal PBL Commands

When a PBL command accesses the 4-Kbyte CCSR address space that is assigned to the PBL, the PBL treats this command entry as a PBL internal command. 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.

Table 5-4 defines each PBL command.

Table 5-4. Description of PBL Commands

Command Name	Offset from CCSRBAR	Parameter(s)	Command Description
Flush	0x13_8000	N/A	Chases the previous write with a read from the same address. The purpose of this command is to ensure previous write has taken effect. **NOTE:** Use of the FLUSH command is restricted to CCSR space. Software should use the WAIT command after

Table continues on the next page...

Table 5-4. Description of PBL Commands (continued)

Command Name	Offset from CCSRBAR	Parameter(s)	Command Description
			commands to non-CCSR space to allow them time to complete before issuing subsequent commands to non-CCSR space.
CRC check	0x13_8040	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	0x13_8080	The 4-bytes of data indicate the target address for the jump destination. For I^2C/eSPI/eSDHC/NOR flash, this address is 4-byte address relative to current address. The address after the jump is a 4-byte aligned address. For 4K large page (128 pages per block) NAND flash, the higher 15 bits of address is non-bad block index relative to current block, the lower 17 bits are ignored. The address after jump is the start of target non-bad block. For 4K large page (64 pages per block) NAND flash, the higher 16 bits of address is non-bad block index relative to current block, the lower 16 bits are ignored. The address after jump is the start of target non-bad block. For 2K large page NAND flash, the higher 17 bits are used as a non-bad block index relative to current block, the lower 15 bits are ignored. The address after the jump is the start of target non-bad block. For small page NAND flash, the higher 20 bits are used as a non-bad block index relative to the current block, The lower 12 bits are ignored. The address after the jump is the start of target non-bad block.	Upon receiving this command, the PBL reads next data from address starting from jump pointer indicated by the command parameter.
Wait	0x13_80C0	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

5.4.7 PBL 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 RSTRQPBLSR[ERR_CODE] in the device configuration and pin control block. See Reset Request Preboot Loader Status Register (DCFG_CCSR_RSTRQPBLSR) for more information.

The following table shows the encoding for each pre-boot error condition.

Table 5-5. Pre-Boot Error Encoding

Error Code[0:6]	Error Condition
0x00	Reserved
0x01	I^2C timeout while waiting for I2CSR[MIF] to assert
0x02	I^2C lost arbitration
0x03	I^2C did not receive ACK during address transmit
0x04	I^2C SCL signal was stuck low at POR
0x05-0x0F	Reserved
0x10	eSPI timeout while waiting for SPIE[DON] to assert
0x11-0x22	Reserved
0x23	IFC detects error in NAND_EVTER_STAT register, NAND_EVTER_STAT can be scanned out upon receiving this error
0x24	IFC timeout error
0x25-0x3F	Reserved
0x40	eSDHC: Err_Reset_1 Indicates that eSDHC has not completed its soft-reset sequence. SYSCTL:RSTA (offset: 0x02C, mask: 0x0100_0000) did not clear within 1 second. It should clear when eSDHC completes its 'reset' sequence.
0x40	eSDHC: Err_Reset_2 Indicates that eSDHC has not completed its soft-reset sequence. SYSCTL:RSTA (offset: 0x02C, mask: 0x0100_0000) did not clear within 1 second. It should clear when eSDHC completes its 'reset' sequence.
0x41	eSDHC: Err_Card_Ins Indicates that eSDHC has not detected an inserted card. PRSSTAT:CINS (offset: 0x024, mask: 0x0001_0000) did not set within 1 second. It should set when eSDHC detects that a card has been inserted.
0x42	eSDHC: Err_Cmd_Data_Rdy Indicates that eSDHC has not detected an idle command and data interface. PRSSTAT:CDIHB (offset: 0x024, mask: 0x0000_0002) and PRSSTAT:CIHB (offset: 0x024, mask: 0x0000_0001) did not clear within 1 second. They should clear when eSDHC detects an idle command and data interface.
0x42	eSDHC: Err_CMD0_Cmd_Data_Rdy Indicates that eSDHC has not detected an idle command and data interface before attempting to send CMD0. PRSSTAT:CDIHB (offset: 0x024, mask: 0x0000_0002) and PRSSTAT:CIHB (offset: 0x024, mask: 0x0000_0001) did not clear within 1 second. They should clear when eSDHC detects an idle command and data interface.
0x42	eSDHC: Err_SD_CMD8_Cmd_Data_Rdy

Table continues on the next page...

Table 5-5. Pre-Boot Error Encoding (continued)

Error Code[0:6]	Error Condition
	Indicates that eSDHC has not detected an idle command and data interface before attempting to send SD CMD8. PRSSTAT:CDIHB (offset: 0x024, mask: 0x0000_0002) and PRSSTAT:CIHB (offset: 0x024, mask: 0x0000_0001) did not clear within 1 second. They should clear when eSDHC detects an idle command and data interface.
0x42	eSDHC: Err_CMD55_Cmd_Data_Rdy Indicates that eSDHC has not detected an idle command and data interface before attempting to send CMD55. PRSSTAT:CDIHB (offset: 0x024, mask: 0x0000_0002) and PRSSTAT:CIHB (offset: 0x024, mask: 0x0000_0001) did not clear within 1 second. They should clear when eSDHC detects an idle command and data interface.
0x42	eSDHC: Err_SD_ACMD41_Cmd_Data_Rdy Indicates that eSDHC has not detected an idle command and data interface before attempting to send SD ACMD41. PRSSTAT:CDIHB (offset: 0x024, mask: 0x0000_0002) and PRSSTAT:CIHB (offset: 0x024, mask: 0x0000_0001) did not clear within 1 second. They should clear when eSDHC detects an idle command and data interface.
0x42	eSDHC: Err_MMC_CMD1_Cmd_Data_Rdy Indicates that eSDHC has not detected an idle command and data interface before attempting to send MMC CMD1. PRSSTAT:CDIHB (offset: 0x024, mask: 0x0000_0002) and PRSSTAT:CIHB (offset: 0x024, mask: 0x0000_0001) did not clear within 1 second. They should clear when eSDHC detects an idle command and data interface.
0x42	eSDHC: Err_CMD2_Cmd_Data_Rdy Indicates that eSDHC has not detected an idle command and data interface before attempting to send CMD2. PRSSTAT:CDIHB (offset: 0x024, mask: 0x0000_0002) and PRSSTAT:CIHB (offset: 0x024, mask: 0x0000_0001) did not clear within 1 second. They should clear when eSDHC detects an idle command and data interface.
0x42	eSDHC: Err_SD_CMD3_Cmd_Data_Rdy Indicates that eSDHC has not detected an idle command and data interface before attempting to send SD CMD3. PRSSTAT:CDIHB (offset: 0x024, mask: 0x0000_0002) and PRSSTAT:CIHB (offset: 0x024, mask: 0x0000_0001) did not clear within 1 second. They should clear when eSDHC detects an idle command and data interface.
0x42	eSDHC: Err_MMC_CMD3_Cmd_Data_Rdy Indicates that eSDHC has not detected an idle command and data interface before attempting to send MMC CMD3. PRSSTAT:CDIHB (offset: 0x024, mask: 0x0000_0002) and PRSSTAT:CIHB (offset: 0x024, mask: 0x0000_0001) did not clear within 1 second. They should clear when eSDHC detects an idle command and data interface.
0x42	eSDHC: Err_CMD9_Cmd_Data_Rdy Indicates that eSDHC has not detected an idle command and data interface before attempting to send CMD9. PRSSTAT:CDIHB (offset: 0x024, mask: 0x0000_0002) and PRSSTAT:CIHB (offset: 0x024, mask: 0x0000_0001) did not clear within 1 second. They should clear when eSDHC detects an idle command and data interface.
0x42	eSDHC: Err_CMD7_Cmd_Data_Rdy Indicates that eSDHC has not detected an idle command and data interface before attempting to send CMD7. PRSSTAT:CDIHB (offset: 0x024, mask: 0x0000_0002) and PRSSTAT:CIHB (offset: 0x024, mask: 0x0000_0001) did not clear within 1 second. They should clear when eSDHC detects an idle command and data interface.
0x42	eSDHC: Err_CMD18_Cmd_Data_Rdy

Table continues on the next page...

Table 5-5. Pre-Boot Error Encoding (continued)

Error Code[0:6]	Error Condition
	Indicates that eSDHC has not detected an idle command and data interface before attempting to send CMD18. PRSSTAT:CDIHB (offset: 0x024, mask: 0x0000_0002) and PRSSTAT:CIHB (offset: 0x024, mask: 0x0000_0001) did not clear within 1 second. They should clear when eSDHC detects an idle command and data interface.
0x42	eSDHC: Err_CMD12_Cmd_Data_Rdy Indicates that eSDHC has not detected an idle command and data interface before attempting to send CMD12. PRSSTAT:CDIHB (offset: 0x024, mask: 0x0000_0002) and PRSSTAT:CIHB (offset: 0x024, mask: 0x0000_0001) did not clear within 1 second. They should clear when eSDHC detects an idle command and data interface.
0x43	eSDHC: Err_Init_Clks Indicates that eSDHC has not completed its initialization clock sequence to the card. SYSCTL:INITA (offset: 0x02C, mask: 0x0800_0000) did not clear within 1 second. It should clear when eSDHC completes transmission of initialization clocks to the card.
0x44	eSDHC: Err_CMD0_Cmd_Stat Indicates that eSDHC has not detected an error or command complete after sending CMD0. IRQSTAT:CCE (offset: 0x030, mask: 0x0002_0000), IRQSTAT:CTOE (offset: 0x030, mask: 0x0001_0000), IRQSTAT:CC (offset: 0x030, mask: 0x0000_0001) did not set within 1 second. They should set when eSDHC detects an error or command complete.
0x44	eSDHC: Err_SD_CMD8_Cmd_Stat Indicates that eSDHC has not detected an error or command complete after sending SD CMD8. IRQSTAT:CIE (offset: 0x030, mask: 0x0008_0000), IRQSTAT:CEBE (offset: 0x030, mask: 0x0004_0000), IRQSTAT:CCE (offset: 0x030, mask: 0x0002_0000), IRQSTAT:CTOE (offset: 0x030, mask: 0x0001_0000), IRQSTAT:CC (offset: 0x030, mask: 0x0000_0001) did not set within 1 second. They should set when eSDHC detects an error or command complete.
0x44	eSDHC: Err_CMD55_Cmd_Stat Indicates that eSDHC has not detected an error or command complete after sending CMD55. IRQSTAT:CIE (offset: 0x030, mask: 0x0008_0000), IRQSTAT:CEBE (offset: 0x030, mask: 0x0004_0000), IRQSTAT:CCE (offset: 0x030, mask: 0x0002_0000), IRQSTAT:CTOE (offset: 0x030, mask: 0x0001_0000), IRQSTAT:CC (offset: 0x030, mask: 0x0000_0001) did not set within 1 second. They should set when eSDHC detects an error or command complete.
0x44	eSDHC: Err_SD_CMD41_Cmd_Stat Indicates that eSDHC has not detected an error or command complete after sending SD ACMD41. IRQSTAT:CEBE (offset: 0x030, mask: 0x0004_0000), IRQSTAT:CCE (offset: 0x030, mask: 0x0002_0000), IRQSTAT:CTOE (offset: 0x030, mask: 0x0001_0000), IRQSTAT:CC (offset: 0x030, mask: 0x0000_0001) did not set within 1 second. They should set when eSDHC detects an error or command complete.
0x44	eSDHC: Err_MMC_CMD1_Cmd_Stat Indicates that eSDHC has not detected an error or command complete after sending MMC CMD1. IRQSTAT:CEBE (offset: 0x030, mask: 0x0004_0000), IRQSTAT:CCE (offset: 0x030, mask: 0x0002_0000), IRQSTAT:CTOE (offset: 0x030, mask: 0x0001_0000), IRQSTAT:CC (offset: 0x030, mask: 0x0000_0001) did not set within 1 second. They should set when eSDHC detects an error or command complete.
0x44	eSDHC: Err_CMD2_Cmd_Stat Indicates that eSDHC has not detected an error or command complete after sending CMD2. IRQSTAT:CEBE (offset: 0x030, mask: 0x0004_0000), IRQSTAT:CCE (offset: 0x030, mask: 0x0002_0000), IRQSTAT:CTOE (offset: 0x030, mask: 0x0001_0000), IRQSTAT:CC (offset: 0x030, mask: 0x0000_0001) did not set within 1 second. They should set when eSDHC detects an error or command complete.

Table continues on the next page...

Table 5-5. Pre-Boot Error Encoding (continued)

Error Code[0:6]	Error Condition
0x44	eSDHC: Err_SD_CMD3_Cmd_Stat Indicates that eSDHC has not detected an error or command complete after sending SD CMD3. IRQSTAT:CIE (offset: 0x030, mask: 0x0008_0000), IRQSTAT:CEBE (offset: 0x030, mask: 0x0004_0000), IRQSTAT:CCE (offset: 0x030, mask: 0x0002_0000), IRQSTAT:CTOE (offset: 0x030, mask: 0x0001_0000), IRQSTAT:CC (offset: 0x030, mask: 0x0000_0001) did not set within 1 second. They should set when eSDHC detects an error or command complete.
0x44	eSDHC: Err_MMC_CMD3_Cmd_Stat Indicates that eSDHC has not detected an error or command complete after sending MMC CMD3. IRQSTAT:CIE (offset: 0x030, mask: 0x0008_0000), IRQSTAT:CEBE (offset: 0x030, mask: 0x0004_0000), IRQSTAT:CCE (offset: 0x030, mask: 0x0002_0000), IRQSTAT:CTOE (offset: 0x030, mask: 0x0001_0000), IRQSTAT:CC (offset: 0x030, mask: 0x0000_0001) did not set within 1 second. They should set when eSDHC detects an error or command complete.
0x44	eSDHC: Err_CMD9_Cmd_Stat Indicates that eSDHC has not detected an error or command complete after sending CMD9. IRQSTAT:CEBE (offset: 0x030, mask: 0x0004_0000), IRQSTAT:CCE (offset: 0x030, mask: 0x0002_0000), IRQSTAT:CTOE (offset: 0x030, mask: 0x0001_0000), IRQSTAT:CC (offset: 0x030, mask: 0x0000_0001) did not set within 1 second. They should set when eSDHC detects an error or command complete.
0x44	eSDHC: Err_CMD7_Cmd_Stat Indicates that eSDHC has not detected an error or command complete after sending CMD7. IRQSTAT:CIE (offset: 0x030, mask: 0x0008_0000), IRQSTAT:CEBE (offset: 0x030, mask: 0x0004_0000), IRQSTAT:CCE (offset: 0x030, mask: 0x0002_0000), IRQSTAT:CTOE (offset: 0x030, mask: 0x0001_0000), IRQSTAT:CC (offset: 0x030, mask: 0x0000_0001) did not set within 1 second. They should set when eSDHC detects an error or command complete.
0x44	eSDHC: Err_CMD18_Cmd_Stat Indicates that eSDHC has not detected an error or command complete after sending CMD18. IRQSTAT:CIE (offset: 0x030, mask: 0x0008_0000), IRQSTAT:CEBE (offset: 0x030, mask: 0x0004_0000), IRQSTAT:CCE (offset: 0x030, mask: 0x0002_0000), IRQSTAT:CTOE (offset: 0x030, mask: 0x0001_0000), IRQSTAT:CC (offset: 0x030, mask: 0x0000_0001) did not set within 1 second. They should set when eSDHC detects an error or command complete.
0x44	eSDHC: Err_CMD12_Cmd_Stat Indicates that eSDHC has not detected an error or command complete after sending CMD12. IRQSTAT:CIE (offset: 0x030, mask: 0x0008_0000), IRQSTAT:CEBE (offset: 0x030, mask: 0x0004_0000), IRQSTAT:CCE (offset: 0x030, mask: 0x0002_0000), IRQSTAT:CTOE (offset: 0x030, mask: 0x0001_0000), IRQSTAT:CC (offset: 0x030, mask: 0x0000_0001) did not set within 1 second. They should set when eSDHC detects an error or command complete.
0x45	eSDHC: Err_CMD0_Cmd_Line_Conflict Indicates that eSDHC has detected a command line conflict after sending CMD0. IRQSTAT:CCE (offset: 0x030, mask: 0x0002_0000) and IRQSTAT:CTOE (offset: 0x030, mask: 0x0001_0000) set. They should set when eSDHC detects a command line conflict. All will set for one of the corresponding reasons. - Command Line Conflict Error (CCE and CTOE)
0x45	eSDHC: Err_CMD8_Cmd_Line_Conflict Indicates that eSDHC has detected a command line conflict after sending SD CMD8. IRQSTAT:CCE (offset: 0x030, mask: 0x0002_0000) and IRQSTAT:CTOE (offset: 0x030, mask: 0x0001_0000) set. They should set when eSDHC detects a command line conflict. All will set for one of the corresponding reasons. - Command Line Conflict Error (CCE and CTOE)
0x45	eSDHC: Err_CMD55_Cmd_Line_Conflict

Table continues on the next page...

Table 5-5. Pre-Boot Error Encoding (continued)

Error Code[0:6]	Error Condition
	Indicates that eSDHC has detected a command line conflict after sending CMD55. IRQSTAT:CCE (offset: 0x030, mask: 0x0002_0000) and IRQSTAT:CTOE (offset: 0x030, mask: 0x0001_0000) set. They should set when eSDHC detects a command line conflict. All will set for one of the corresponding reasons. - Command Line Conflict Error (CCE and CTOE)
0x45	eSDHC: Err_SD_ACMD41_Cmd_Line_Conflict Indicates that eSDHC has detected a command line conflict after sending SD ACMD41. IRQSTAT:CCE (offset: 0x030, mask: 0x0002_0000) and IRQSTAT:CTOE (offset: 0x030, mask: 0x0001_0000) set. They should set when eSDHC detects a command line conflict. All will set for one of the corresponding reasons. - Command Line Conflict Error (CCE and CTOE)
0x45	eSDHC: Err_MMC_CMD1_Cmd_Line_Conflict Indicates that eSDHC has detected a command line conflict after sending MMC CMD1. IRQSTAT:CCE (offset: 0x030, mask: 0x0002_0000) and IRQSTAT:CTOE (offset: 0x030, mask: 0x0001_0000) set. They should set when eSDHC detects a command line conflict. All will set for one of the corresponding reasons. - Command Line Conflict Error (CCE and CTOE)
0x45	eSDHC: Err_CMD2_Cmd_Line_Conflict Indicates that eSDHC has detected a command line conflict after sending CMD2. IRQSTAT:CCE (offset: 0x030, mask: 0x0002_0000) and IRQSTAT:CTOE (offset: 0x030, mask: 0x0001_0000) set. They should set when eSDHC detects a command line conflict. All will set for one of the corresponding reasons. - Command Line Conflict Error (CCE and CTOE)
0x45	eSDHC: Err_SD_CMD3_Cmd_Line_Conflict Indicates that eSDHC has detected a command line conflict after sending SD CMD3. IRQSTAT:CCE (offset: 0x030, mask: 0x0002_0000) and IRQSTAT:CTOE (offset: 0x030, mask: 0x0001_0000) set. They should set when eSDHC detects a command line conflict. All will set for one of the corresponding reasons. - Command Line Conflict Error (CCE and CTOE)
0x45	eSDHC: Err_MMC_CMD3_Cmd_Line_Conflict Indicates that eSDHC has detected a command line conflict after sending MMC CMD3. IRQSTAT:CCE (offset: 0x030, mask: 0x0002_0000) and IRQSTAT:CTOE (offset: 0x030, mask: 0x0001_0000) set. They should set when eSDHC detects a command line conflict. All will set for one of the corresponding reasons. - Command Line Conflict Error (CCE and CTOE)
0x45	eSDHC: Err_CMD9_Cmd_Line_Conflict Indicates that eSDHC has detected a command line conflict after sending CMD9. IRQSTAT:CCE (offset: 0x030, mask: 0x0002_0000) and IRQSTAT:CTOE (offset: 0x030, mask: 0x0001_0000) set. They should set when eSDHC detects a command line conflict. All will set for one of the corresponding reasons. - Command Line Conflict Error (CCE and CTOE)
0x45	eSDHC: Err_CMD7_Cmd_Line_Conflict Indicates that eSDHC has detected a command line conflict after sending CMD7. IRQSTAT:CCE (offset: 0x030, mask: 0x0002_0000) and IRQSTAT:CTOE (offset: 0x030, mask: 0x0001_0000) set. They should set when eSDHC detects a command line conflict. All will set for one of the corresponding reasons. - Command Line Conflict Error (CCE and CTOE)
0x45	eSDHC: Err_CMD18_Cmd_Line_Conflict Indicates that eSDHC has detected a command line conflict after sending CMD18. IRQSTAT:CCE (offset: 0x030, mask: 0x0002_0000) and IRQSTAT:CTOE (offset: 0x030, mask: 0x0001_0000) set. They should set when eSDHC detects a command line conflict. All will set for one of the corresponding reasons. - Command Line Conflict Error (CCE and CTOE)

Table continues on the next page...

Table 5-5. Pre-Boot Error Encoding (continued)

Error Code[0:6]	Error Condition
0x45	eSDHC: Err_CMD12_Cmd_Line_Conflict Indicates that eSDHC has detected a command line conflict after sending CMD12. IRQSTAT:CCE (offset: 0x030, mask: 0x0002_0000) and IRQSTAT:CTOE (offset: 0x030, mask: 0x0001_0000) set. They should set when eSDHC detects a command line conflict. All will set for one of the corresponding reasons. - Command Line Conflict Error (CCE and CTOE)
0x46	eSDHC: Err_SD_CMD8_Cmd_Rsp Indicates that eSDHC has detected an error with the command response for SD CMD8. IRQSTAT:CIE (offset: 0x030, mask: 0x0008_0000), IRQSTAT:CEBE (offset: 0x030, mask: 0x0004_0000), or IRQSTAT:CCE (offset: 0x030, mask: 0x0002_0000) set. Each will set for one of the corresponding reasons. - Command Index Error (CIE) - Command End Bit Error (CEBE) - Command CRC Error (CCE)
0x46	eSDHC: Err_CMD55_Cmd_Rsp Indicates that eSDHC has detected an error with the command response for CMD55. IRQSTAT:CIE (offset: 0x030, mask: 0x0008_0000), IRQSTAT:CEBE (offset: 0x030, mask: 0x0004_0000), or IRQSTAT:CCE (offset: 0x030, mask: 0x0002_0000) set. Each will set for one of the corresponding reasons. - Command Index Error (CIE) - Command End Bit Error (CEBE) - Command CRC Error (CCE)
0x46	eSDHC: Err_SD_ACMD41_Cmd_Rsp Indicates that eSDHC has detected an error with the command response for SD ACMD41. IRQSTAT:CEBE (offset: 0x030, mask: 0x0004_0000) set. It will set for one of the corresponding reasons. - Command End Bit Error (CEBE)
0x46	eSDHC: Err_MMC_CMD1_Cmd_Rsp Indicates that eSDHC has detected an error with the command response for MMC CMD1. IRQSTAT:CEBE (offset: 0x030, mask: 0x0004_0000) set. It will set for one of the corresponding reasons. - Command End Bit Error (CEBE)
0x46	eSDHC: Err_CMD2_Cmd_Rsp Indicates that eSDHC has detected an error with the command response for CMD2. IRQSTAT:CEBE (offset: 0x030, mask: 0x0004_0000) or IRQSTAT:CCE (offset: 0x030, mask: 0x0002_0000) set. Each will set for one of the corresponding reasons. - Command End Bit Error (CEBE) - Command CRC Error (CCE)
0x46	eSDHC: Err_SD_CMD3_Cmd_Rsp Indicates that eSDHC has detected an error with the command response for SD CMD3. IRQSTAT:CIE (offset: 0x030, mask: 0x0008_0000), IRQSTAT:CEBE (offset: 0x030, mask: 0x0004_0000), or IRQSTAT:CCE (offset: 0x030, mask: 0x0002_0000) set. Each will set for one of the corresponding reasons. - Command Index Error (CIE) - Command End Bit Error (CEBE)

Table continues on the next page...

Table 5-5. Pre-Boot Error Encoding (continued)

Error Code[0:6]	Error Condition
	- Command CRC Error (CCE)
0x46	eSDHC: Err_MMC_CMD3_Cmd_Rsp Indicates that eSDHC has detected an error with the command response for MMC CMD3. IRQSTAT:CIE (offset: 0x030, mask: 0x0008_0000), IRQSTAT:CEBE (offset: 0x030, mask: 0x0004_0000), or IRQSTAT:CCE (offset: 0x030, mask: 0x0002_0000) set. Each will set for one of the corresponding reasons. - Command Index Error (CIE) - Command End Bit Error (CEBE) - Command CRC Error (CCE)
0x46	eSDHC: Err_CMD9_Cmd_Rsp Indicates that eSDHC has detected an error with the command response for CMD9. IRQSTAT:CEBE (offset: 0x030, mask: 0x0004_0000) or IRQSTAT:CCE (offset: 0x030, mask: 0x0002_0000) set. Each will set for one of the corresponding reasons. - Command End Bit Error (CEBE) - Command CRC Error (CCE)
0x46	eSDHC: Err_CMD7_Cmd_Rsp Indicates that eSDHC has detected an error with the command response for CMD7. IRQSTAT:CIE (offset: 0x030, mask: 0x0008_0000), IRQSTAT:CEBE (offset: 0x030, mask: 0x0004_0000), or IRQSTAT:CCE (offset: 0x030, mask: 0x0002_0000) set. Each will set for one of the corresponding reasons. - Command Index Error (CIE) - Command End Bit Error (CEBE) - Command CRC Error (CCE)
0x46	eSDHC: Err_CMD18_Cmd_Rsp Indicates that eSDHC has detected an error with the command response for CMD18. IRQSTAT:CIE (offset: 0x030, mask: 0x0008_0000), IRQSTAT:CEBE (offset: 0x030, mask: 0x0004_0000), or IRQSTAT:CCE (offset: 0x030, mask: 0x0002_0000) set. Each will set for one of the corresponding reasons. - Command Index Error (CIE) - Command End Bit Error (CEBE) - Command CRC Error (CCE)
0x46	eSDHC: Err_CMD12_Cmd_Rsp Indicates that eSDHC has detected an error with the command response for CMD12. IRQSTAT:CIE (offset: 0x030, mask: 0x0008_0000), IRQSTAT:CEBE (offset: 0x030, mask: 0x0004_0000), or IRQSTAT:CCE (offset: 0x030, mask: 0x0002_0000) set. Each will set for one of the corresponding reasons. - Command Index Error (CIE) - Command End Bit Error (CEBE) - Command CRC Error (CCE)
0x47	eSDHC: Err_SD_CMD8_Rsp_Data_Match

Table continues on the next page...

Table 5-5. Pre-Boot Error Encoding (continued)

Error Code[0:6]	Error Condition
	Indicates that eSDHC cannot properly communicate with the card due to a mismatch of the expected response of SD CMD8. CMDRSP0:VHS (offset: 0x010, mask: 0x0000_0F00) was not equal to the configured SD 'coarse' voltage or CMDRSP0:CHKPTRN (offset: 0x010, mask: 0x0000_00FF) was not equal to the configured SD check pattern. Both fields have to match in order to guarantee proper communication with the card.
0x47	eSDHC: Err_SD_ACMD41_Rsp_Data_Match Indicates that eSDHC cannot properly communicate with the card due to a mismatch of the expected response of SD ACMD41. CMDRSP0:VDDVW (offset: 0x010, mask: 0x00FF_FFFF) was not covered (matches at least one bit) by the configured SD 'fine' voltage. At least one bit must match in order to guarantee proper communication with the card.
0x47	eSDHC: Err_MMC_CMD1_Rsp_Data_Match Indicates that eSDHC cannot properly communicate with the card due to a mismatch of the expected response of MMC CMD1. CMDRSP0:VDDVW (offset: 0x010, mask: 0x00FF_FFFF) was not covered (matches at least one bit) by the configured MMC voltage. At least one bit must match in order to guarantee proper communication with the card.
0x48	eSDHC: Err_SD_ACMD41_Cmd_Rsp_Timeout Indicates that eSDHC has detected a timeout with the command response for SD ACMD41. IRQSTAT:CTOE (offset: 0x030, mask: 0x0001_0000) set. It will set for one of the corresponding reasons. - Command Timeout Error (CTOE)
0x48	eSDHC: Err_MMC_CMD1_Cmd_Rsp_Timeout Indicates that eSDHC has detected a timeout with the command response for MMC CMD1. IRQSTAT:CTOE (offset: 0x030, mask: 0x0001_0000) set. It will set for one of the corresponding reasons. - Command Timeout Error (CTOE)
0x48	eSDHC: Err_CMD2_Cmd_Rsp_Timeout Indicates that eSDHC has detected a timeout with the command response for CMD2. IRQSTAT:CTOE (offset: 0x030, mask: 0x0001_0000) set. It will set for one of the corresponding reasons. - Command Timeout Error (CTOE)
0x48	eSDHC: Err_SD_CMD3_Cmd_Rsp_Timeout Indicates that eSDHC has detected a timeout with the command response for SD CMD3. IRQSTAT:CTOE (offset: 0x030, mask: 0x0001_0000) set. It will set for one of the corresponding reasons. - Command Timeout Error (CTOE)
0x48	eSDHC: Err_MMC_CMD3_Cmd_Rsp_Timeout Indicates that eSDHC has detected a timeout with the command response for MMC CMD3. IRQSTAT:CTOE (offset: 0x030, mask: 0x0001_0000) set. It will set for one of the corresponding reasons. - Command Timeout Error (CTOE)
0x48	eSDHC: Err_CMD9_Cmd_Rsp_Timeout Indicates that eSDHC has detected a timeout with the command response for CMD9. IRQSTAT:CTOE (offset: 0x030, mask: 0x0001_0000) set. It will set for one of the corresponding reasons. - Command Timeout Error (CTOE)
0x48	eSDHC: Err_CMD7_Cmd_Rsp_Timeout Indicates that eSDHC has detected a timeout with the command response for CMD7. IRQSTAT:CTOE (offset: 0x030, mask: 0x0001_0000) set. It will set for one of the corresponding reasons. - Command Timeout Error (CTOE)

Table continues on the next page...

Table 5-5. Pre-Boot Error Encoding (continued)

Error Code[0:6]	Error Condition
0x48	eSDHC: Err_CMD18_Cmd_Rsp_Timeout Indicates that eSDHC has detected a timeout with the command response for CMD18. IRQSTAT:CTOE (offset: 0x030, mask: 0x0001_0000) set. It will set for one of the corresponding reasons. - Command Timeout Error (CTOE)
0x48	eSDHC: Err_CMD12_Cmd_Rsp_Timeout Indicates that eSDHC has detected a timeout with the command response for CMD12. IRQSTAT:CTOE (offset: 0x030, mask: 0x0001_0000) set. It will set for one of the corresponding reasons. - Command Timeout Error (CTOE)
0x49	eSDHC: Err_SD_ACMD41_Rdy_Timeout Indicates that the card always indicates busy by examining the response of SD ACMD41. CMDRSP0:RDY (offset: 0x010, mask: 0x8000_0000) did not set within 1 second. It should set when the card completes its initialization sequence.
0x49	eSDHC: Err_MMC_CMD1_Rdy_Timeout Indicates that the card always indicates busy by examining the response of MMC CMD1. CMDRSP0:RDY (offset: 0x010, mask: 0x8000_0000) did not set within 1 second. It should set when the card completes its initialization sequence.
0x4A	eSDHC: Err_Rd_Buf_Rdy Indicates that eSDHC has not received the watermark's amount (configured to 4 bytes) of data in its read buffer from CMD18. PRSSTAT:BREN (offset: 0x024, mask: 0x0000_0800) did not set within 1 second. It should set when eSDHC receives the watermark's amount of data in its read buffer.
0x4B	eSDHC: Err_Data Indicates that eSDHC has detected an error with the read data from CMD18. IRQSTAT:DEBE (offset: 0x030, mask: 0x0040_0000) or IRQSTAT:DCE (offset: 0x030, mask: 0x0020_0000) set. Each will set for one of the corresponding reasons. - Data End Bit Error (DEBE) - Data CRC Error (DCE)
0x4C	eSDHC: Err_Data_Timeout Indicates that eSDHC has detected a timeout with the read data from CMD18. IRQSTAT:DTOE (offset: 0x030, mask: 0x0010_0000) set. It will set for one of the corresponding reasons. - Data Timeout Error (DTOE)
0x4D	eSDHC: Err_Data_Reset Indicates that eSDHC has not completed its soft-reset sequence. SYSCTL:RSTD (offset: 0x02C, mask: 0x0400_0000) did not clear within 1 second. It should clear when eSDHC completes its data 'reset' sequence.
0x4E	eSDHC: Err_PBI_Timeout Indicates that pre-boot initialization (PBI) has not started within 1 second. An indication should be received when PBI starts.
0x4F	eSDHC: Reserved
0x50-0x6F	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

Table continues on the next page...

Table 5-5. Pre-Boot Error Encoding (continued)

Error Code[0:6]	Error Condition
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 word source encoding
0x7B-0x7F	Reserved

5.5 Registers Written by PBL During RCW and PBI Phases

This section lists registers written by PBL during RCW and PBI phases.

5.5.1 I^2C Registers

If any of the following registers are written, they are reset back to their POR value when the PBL has completed:

- I2CFDR
- I2CCR
- I2CSR
- I2CDR
- I2CDFSRR

5.5.2 eSPI Registers

If any of the following registers are written, they are reset back to their POR value when the PBL has completed:

- SPMODE
- SPIE
- SPCOM
- SPITF
- SPIRF (read only)
- SPMODE0

5.5.3 IFC Registers

If any of the following registers are written, they are reset back to their POR value when the PBL has completed:

- NAND_MDR
- NANDSEQ_STRT
- NAND_EVTER_STAT
- NAND_ERATTR0
- NCFGR
- CSPR0
- CSOR0
- NAND_FIR0
- NAND_FCR0
- ROW0
- COL0
- NAND_BC

CSOR0 register holds the pages per block information from configuration word. The register field PB of CSOR0 is read by PBL to determine the pages per block.

5.5.4 eSDHC Registers

If any of the following registers are written, they are reset back to their POR value when the PBL has completed. The following eSDHC registers are reset by performing a software reset, asserting SYSCTL:RSTA:

- BLKATTR (Offset: 0x004)
- CMDARG (Offset: 0x008)
- XFERTYP (Offset: 0x00C)
- CMDRSP0 (Offset: 0x010)
- CMDRSP2 (Offset: 0x018)
- DATPORT (Offset: 0x020)
- PRSSTAT (Offset: 0x024)
- PROCTL (Offset: 0x028)
- SYSCTL (Offset: 0x02C)
- IRQSTAT (Offset: 0x030)
- IRQSTATEN (Offset: 0x034)
- WML (Offset: 0x044)

5.6 Addressing Multiple I²C EEPROMs

When standard I²C EEPROMs are used, it is assumed that each EEPROM holds 256 bytes of data. After 256 bytes have been received, the PBL will increment the slave address and access the next EEPROM. A maximum of eight standard I²C EEPROMs may be used with the PBL.

Extended I²C EEPROMs have variable sizes. However, the I²C controller increments the slave address after 64 Kbytes have been received. In some cases, the least significant bit of the slave address refers to a different page address on an extended EEPROM (for the larger density EEPROMs). However, the PBL should still be able to access the page successfully. Alternatively, the least significant bit may reference a second extended EEPROM on the I²C bus. Multiple extended EEPROMs of sizes less than 64 Kbytes are not supported (unless a jump command is used to force the PBL to the next EEPROM device).

5.7 eSPI Mode Assumptions

eSPI mode assumptions are listed as follows:

- Chip select-when using eSPI as the source of PBL, the first chip select (CS0) is used. PBL uses the SPMODE0 register in the eSPI block to use CS0 during initialization.
- Supported SPI mode-when using eSPI as the source for PBL, only SPI Mode 0 is supported (referring to the SPI clock phase and polarity). The SPI slave device must also support SPI Mode 0.
- SPI Read command-when using eSPI as the source for PBL, a value of 0x03 is used for the read command. The SPI slave device must decode 0x03 as a read command.

5.8 PBL 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.

If booting from NAND, the boot image must be placed on the next 4-Kbyte boundary. The PBL searches for the next good block should that 4-Kbyte boundary fall on a block boundary. The PBL then loads 4 Kbytes into the buffer.

This PBL block search routine continues to search as necessary until reaching the end of the memory device.

5.8.1 Starting Addresses

Table 5-6 lists the starting addresses of data fetched from each interface. Note that in case of IFC NAND flash, the address starts from the first good block.

Table 5-6. Starting Addresses

Interface	Starting Address
IFC	0x00000000
I^2C[1]	0x00000000
eSPI	0x00000000
eSDHC	0x00001000

1. The 7-bit calling address for the I^2C slave is 0x50.

5.8.2 Software Restrictions

The following are restrictions for programming the PBI commands:

- CCSRBAR cannot be updated through PBI commands. Software must wait for the cores to boot to change CCSRBAR.
- Software must issue a Flush command after updating the alternate configuration space registers (ALTCBARH, ALTCBARL, and ALTCAR) 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 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.

5.8.3 Software Recommendations

The following are recommended 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 6
Secure Boot and Trust Architecture 2.0

6.1 Trust Architecture overview

The Trust Architecture 2.0 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 Freescale and deployed on other Freescale platforms. These capabilities have been modified and extended 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 as a whole, followed by detailed information on the security fuse processor (SFP) and Security Monitor. 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.

Customers already familiar with Trust Architecture 1.x should note that the primary differences between 1.x and 2.0 are:

- Reduced secure boot time by offloading cryptography to the SEC
- Support for a primary and alternate (secondary) signed image
- Support for revocation of super root keys
- Monotonic counter
- Battery-backed, general purpose storage registers

6.1.1 Objectives of trust architecture 2.0

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 device's implementation of the Trust Architecture, if developers properly leverage the hardware hooks in the device, they can '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 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 heavy 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 which are intended to persist between resets of the system. Trust architecture 2.0 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 non-volatile memory.
- Unauthorized exposure of system ephemeral secrets. These are secrets which 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 which are encrypted with a JDKEK (also known as, "Black Keys").
- Accidental or deliberate use of the private resources of one software partition by any other software partition

While some developers consider security policy enforcement to be a critical feature of the device, 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 Freescale to provision devices or sign code. Freescale is not part of the system development or manufacturing chain of trust. Developer provisioning of devices is designed to be simple, with minimal impacts on manufacturing cycle times.

6.1.2 Trust Architecture as implemented on the chip

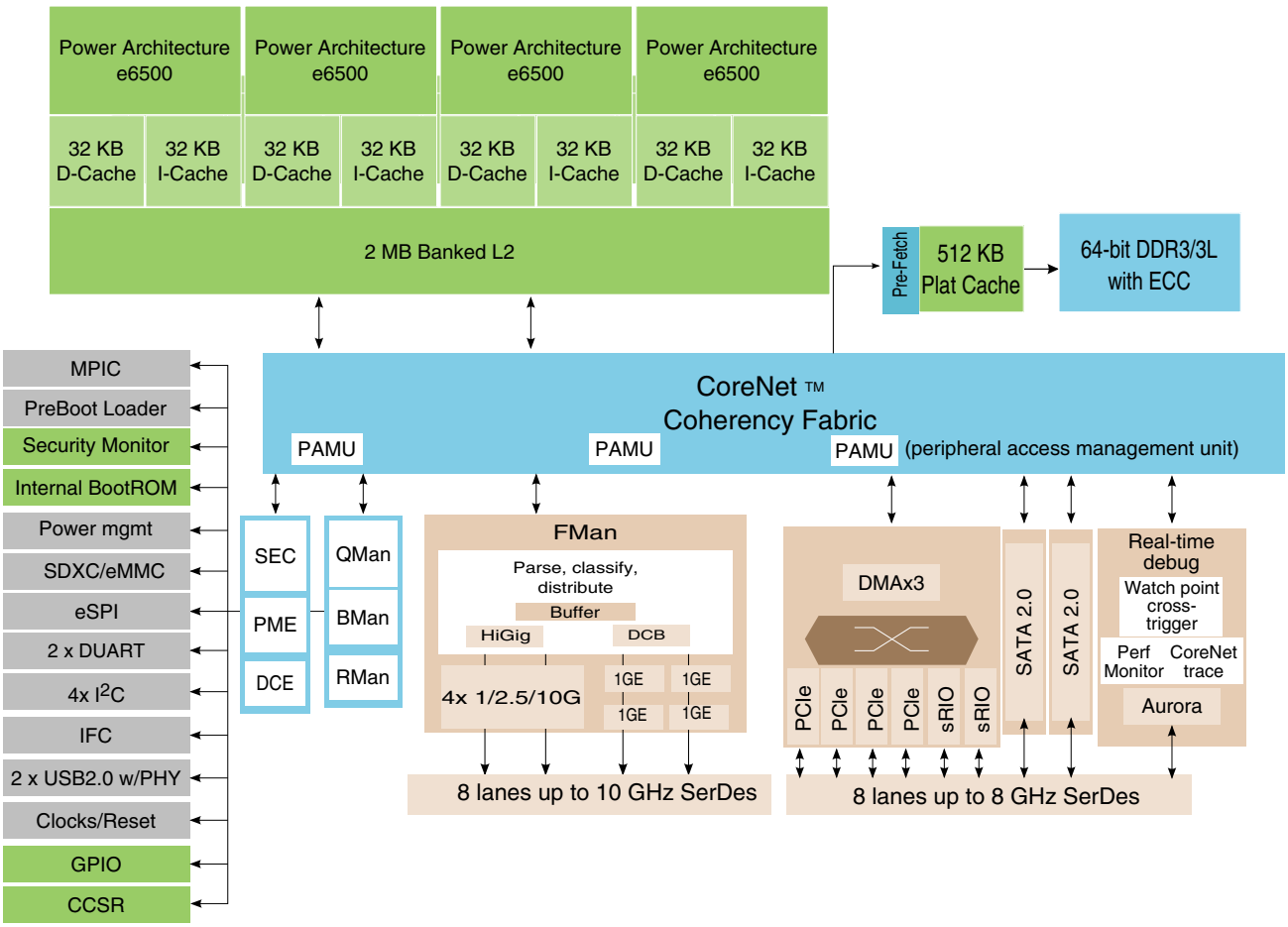


Figure 6-1. T2080 with QorIQ platform's Trust Architecture

6.1.2.1 Power Architecture core

The Power Architecture core has an important role to play in both secure boot and secure runtime operations.

If the device is configured to perform secure boot, at power on reset the booting core is released to begin execution from the internal boot ROM (IBR).

The instructions executed from the IBR allows the booting core to determine if code outside the IBR is safe to execute. This process will be discussed further in later sections.

6.1.2.2 No execute bit (UX and SX bits)

The Power Architecture Book-III E translation look-aside buffer (TLB) includes control bits that CPU use to determine read, write, execute, and caching rules for the memory pages.

The UX and SX bits in the TLB control whether a page's contents can be executed as instructions by user (UX) and supervisor (SX) programs. The ability to define pages as non-executable provides a significant barrier against attacks that overflow data buffers into code memory space.

6.1.2.3 Hypervisor

The core's embedded hypervisor architecture introduces a third privilege level called "guest state" (GS bit in the core's machine state register).

This allows Freescale or 3rd party hypervisor software to run at the most privileged level, with operating systems at a less privileged level, and applications at the least privileged level. This allows for a hypervisor software layer to virtualize the core (vcpu) and block any guest OS attempts to modify critical security configurations, such as modifications of page table entries. By virtualizing all CPUs in the chip, as well as the platform as a whole, systems can run with a mixture of trusted and untrusted partitions.

Details of the core can be found in Introduction .

6.1.2.4 PAMUs

The core's MMU settings determine which memory ranges are accessible by each partition, and the hypervisor prevents these settings from being altered by operating system or application software.

In order to prevent system masters other than the cores from reading or writing sensitive memory regions, the chip implements a number of I/O MMUs, known as peripheral access management units or PAMUs.

The PAMUs prevent internal and external DMAs (non-CPU masters) from accessing memory for which they have not been granted explicit access permission.

The hypervisor assigns each non-CPU master in the chip one or more logical I/O device numbers (LIODNs). A non-CPU master asserts an LIODN (based on the ID of the partition which requested the non-CPU master to perform an operation) on each bus access. The PAMU grants or denies access to particular memory ranges based upon the

LIODN value. Non-CPU masters cannot alter the LIODN values they use for their transactions, so this value serves to identify and authenticate the bus transaction as having originated at one of a set of masters that share that particular LIODN value.

When secure boot is enabled, PAMUs block all external masters by default. Authenticated software can change the PAMU access permissions to authorize external masters later. This means that the chip cannot be booted as an agent when secure boot is enabled.

Details of the PAMUs can be found in Peripheral access management unit (PAMU).

6.1.2.5 Secure debug controller

The secure debug controller (SDC) supports four levels of access.

- Open-External debug agents connected to the Aurora or 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 Aurora or 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) and 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.

6.1.2.6 SEC 5.2

The primary function of the SEC 5.2 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, one-time programmable master key, battery backed ZMK, and randomly initialized JDKEKs. 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.

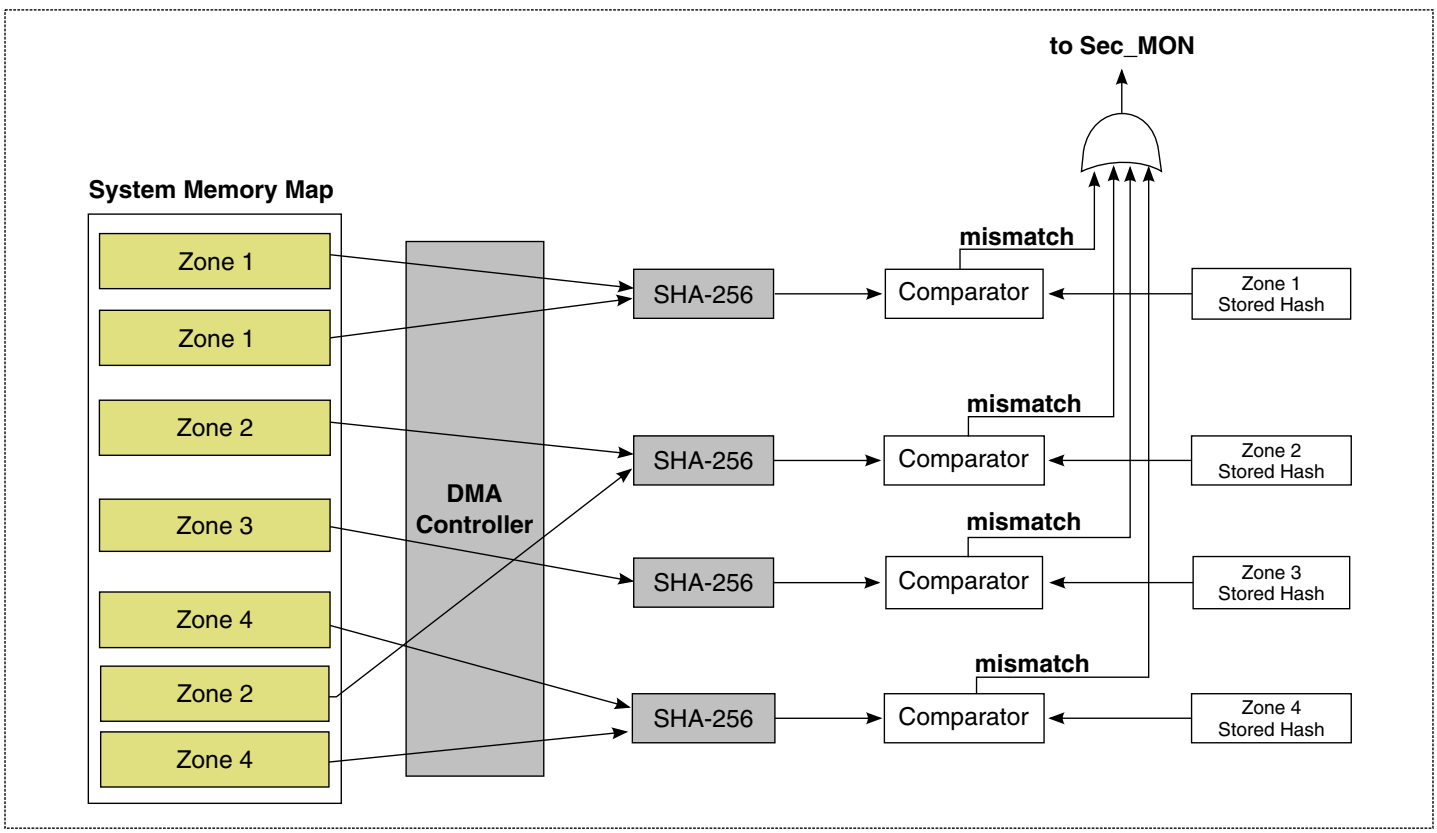


Figure 6-2. SEC 5.2 Performing Memory Integrity Checking

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.

6.1.2.7 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 by an attacker. 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 2.0 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 2.0 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 Freescale Code Signing Tool supports both 1.x and 2.0 devices, however the user must specify the Trust Arch generation when creating the signed image. The ISBC in Trust 2.0 devices will not validate images signed as 1.x devices.

6.1.2.8 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 a tamper detection input signal (TMP_DETECT).

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 input at 1.0 V, with the

security monitor reacting to a voltage drop in this signal. Detection of an external tamper event is reported in the SM_HP Security Violation Status Register (SECMON_HPSVSR) and SM_LP Status Register (SECMON_LPSR) .

In addition to the high-power (HP) tamper detect, the chip implements a low-power tamper input for recieving tamper signals while the chip is powered off. LP tamp-detect is relevant when using the battery backed zeriozable secret key.

6.1.2.9 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 PBL writes 512 bits of RCW, beginning with the first RCWSR register in the device 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 device (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 which 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.

6.1.2.10 Security fuse processor

The security fuse processor (SFP) has the following roles:

- To physically burn fuses during device 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 Freescale 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 device.

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).

6.1.2.11 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 SEC 5.2 to use persistent and ephemeral secrets. Details of the security monitor can be found in Security monitor.

6.1.3 Code signing

As described in Objectives of trust architecture 2.0, the device 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 device 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 no-execute bits (UX and SX).

The starting point for a trusted platform is the creation (by the developer) of a bugfree 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 device's SDK along with the Code Signing Tool.

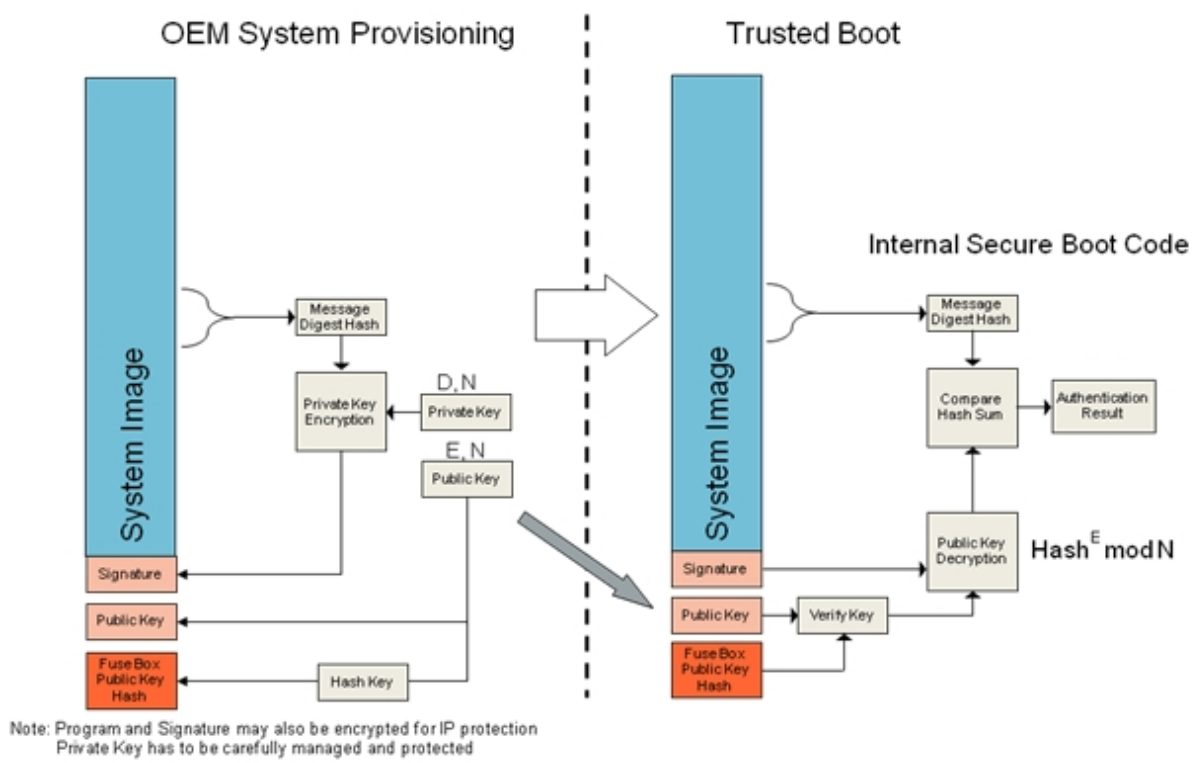


Figure 6-3. Code signing and signature validation

As shown in the figure above, the developer calculates a hash (the device 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.

6.1.4 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.

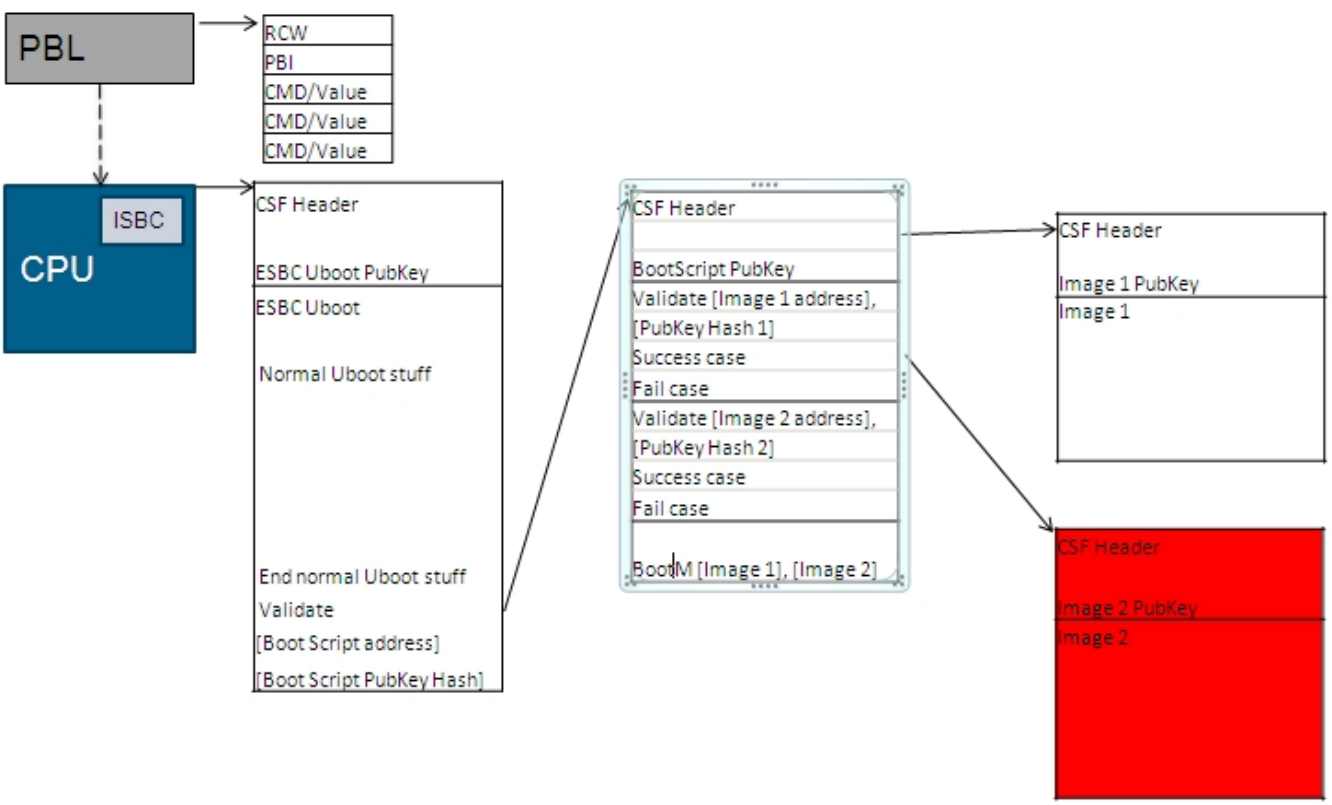


Figure 6-4. Secure boot chain of trust

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.

6.1.4.1 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 device, essentially locking down interfaces, memory permissions, and MMU configurations until trusted software is executing.

6.1.4.2 ISBC phase

1. Platform self tests; ensuring that the device 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 SCRATCHRW1 register), validating the public key to be used in authentication, image authentication, and first instruction validation. Note that Freescale generically refers to the external image to be validated as the External Secure Boot Code (ESBC).
4. Error Reporting; if the ESBC cannot be successfully validated, the ISBC will write an error code to the device's SCRATCHRW2 register.
5. If no errors, the ISBC writes the Sec_Mon command register, transitioning the device to "Trusted," and the master key (OTPMK or ZMK) is made available to the SEC.

6.1.4.3 ESBC phase

Unlike the ISBC, which is in an internal ROM and therefore unchangeable, the ESBC is Freescale-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 Freescale's reference software for secure boot.

6.1.5 Key revocation

Trust Architecture 2.0 introduces support for revoking 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 Freescale Code Signing Tool (CST), the OEM defines whether the device 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.

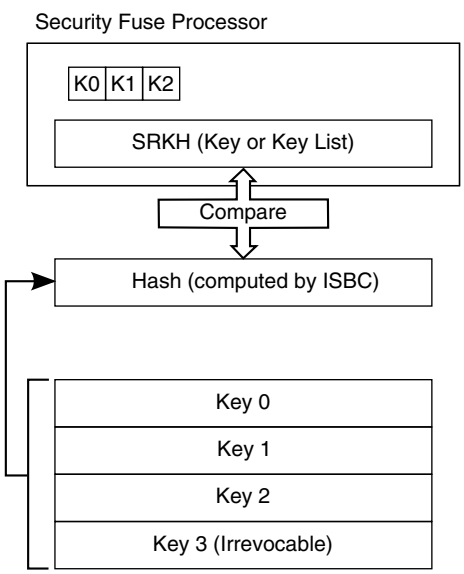


Figure 6-5. ESBC validation with a key list

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 enforce using the keys in a specific order.

The key revocation fuses in the SFP_OSPR are not protected by the OEM section's Write Protect. They are protected by a SFP "Write Disable", which is set by the ISBC when the "Write Protect" flag is set in the CSF header. In regular operation, the OEM should use the Freescale Code Signing Tool to sign an image with CSF header flag set to "Write Protect". When circumstances call for revoking a key, the OEM signs an image with the CSF header "Write Protect" flag not set. Following signature validation, the SFP will be in a state in which key revocation fuses can be set. Logically, the OEM would then load a new signed image, with new CSF header indicating which of the remaining non-revoked key to use and the "Write Protect" bit set. Only the possessor of a legitimate RSA private key can enable key revocation.

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.

6.2 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.

6.2.1 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 device, or via an external interface, such as the JTAG. See , and Freescale Section Write Protect Register (SFP_FSWPR), for more information on write protection.

The device is enabled for fuse writing by connecting a signal (POVDD) to a 1.8 V source. During reset, POVDD should be tied to GND. After the fuses are written, POVDD must be returned to GND. Once the desired values are written to the mirror registers, the SFP is instructed (via a write to the instruction register) to begin blowing fuses in the array.

The fuse programming sequence and example values are included in the Freescale whitepaper, *Manufacturing for Trust*.

6.2.2 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 SM_HP Security Violation Status Register (SECMON_HPSVSR). The SFP's ability to detect specific bad fuse values is described in , and the security monitor's ability to detect a bad OTPMK is described in SM_HP Status Register (SECMON_HPSR). If fuse errors are suspected in other 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.

6.2.3 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 Freescale section are typical, the exact values will depend on the values programmed into the fuse array prior to the device leaving Freescale'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 about to be written to) the fuse array. Mirror registers are further defined as correlating to Freescale sections of the fuse array or OEM sections of the fuse array.

All register are 32-bit, 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 registers offsets.

SFP memory map

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E_8020	Instruction Register (SFP_INGR)	32	R/W	0000_0000h	6.2.3.1/235
E_8024	Secret Value Hamming Error Status Register (SFP_SVHESR)	32	R	0000_0000h	6.2.3.2/236
E_8028	SFP Configuration Register (SFP_SFPCR)	32	R/W	See section	6.2.3.3/237
E_8034	SFP Version Register (SFP_VERSION)	32	R	See section	6.2.3.4/238
E_8200	OEM Security Policy Register (SFP_OSPR)	32	R/W	0000_0000h	6.2.3.5/239
E_8204	Freescale Section Write Protect Register (SFP_FSWPR)	32	R/W	0000_0001h	6.2.3.6/241
E_8208	Debug Permissions Register (SFP_DPR)	32	R/W	0000_0000h	6.2.3.7/242
E_820C	Debug Challenge Value Register n (SFP_DCVR0)	32	R/W	0000_0000h	6.2.3.8/243
E_8210	Debug Challenge Value Register n (SFP_DCVR1)	32	R/W	0000_0000h	6.2.3.8/243
E_8214	Debug Response Value Register n (SFP_DRVR0)	32	W	0000_0000h	6.2.3.9/244
E_8218	Debug Response Value Register n (SFP_DRVR1)	32	W	0000_0000h	6.2.3.9/244
E_821C	One Time Programmable Master Key n (SFP_OTPMKR0)	32	W	0000_0000h	6.2.3.10/ 246
E_8220	One Time Programmable Master Key n (SFP_OTPMKR1)	32	W	0000_0000h	6.2.3.10/ 246
E_8224	One Time Programmable Master Key n (SFP_OTPMKR2)	32	W	0000_0000h	6.2.3.10/ 246
E_8228	One Time Programmable Master Key n (SFP_OTPMKR3)	32	W	0000_0000h	6.2.3.10/ 246
E_822C	One Time Programmable Master Key n (SFP_OTPMKR4)	32	W	0000_0000h	6.2.3.10/ 246
E_8230	One Time Programmable Master Key n (SFP_OTPMKR5)	32	W	0000_0000h	6.2.3.10/ 246
E_8234	One Time Programmable Master Key n (SFP_OTPMKR6)	32	W	0000_0000h	6.2.3.10/ 246
E_8238	One Time Programmable Master Key n (SFP_OTPMKR7)	32	W	0000_0000h	6.2.3.10/ 246
E_823C	Super Root Key Hash n (SFP_SRKHR0)	32	R/W	0000_0000h	6.2.3.11/ 249
E_8240	Super Root Key Hash n (SFP_SRKHR1)	32	R/W	0000_0000h	6.2.3.11/ 249
E_8244	Super Root Key Hash n (SFP_SRKHR2)	32	R/W	0000_0000h	6.2.3.11/ 249
E_8248	Super Root Key Hash n (SFP_SRKHR3)	32	R/W	0000_0000h	6.2.3.11/ 249
E_824C	Super Root Key Hash n (SFP_SRKHR4)	32	R/W	0000_0000h	6.2.3.11/ 249
E_8250	Super Root Key Hash n (SFP_SRKHR5)	32	R/W	0000_0000h	6.2.3.11/ 249
E_8254	Super Root Key Hash n (SFP_SRKHR6)	32	R/W	0000_0000h	6.2.3.11/ 249

Table continues on the next page...

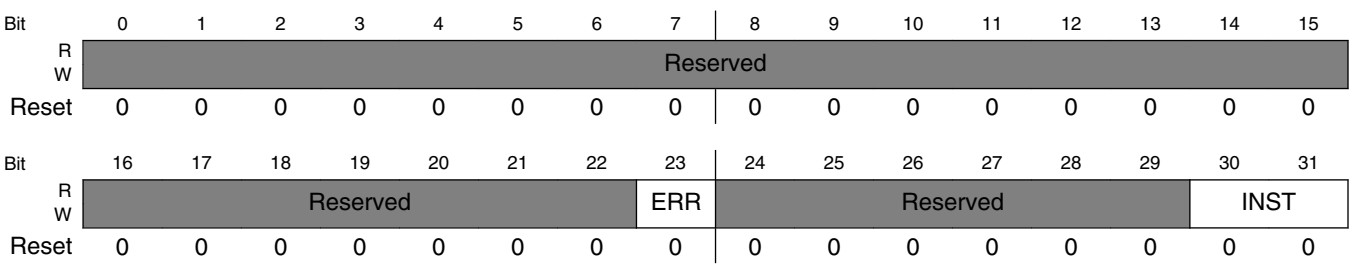
SFP memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E_8258	Super Root Key Hash n (SFP_SRKHR7)	32	R/W	0000_0000h	6.2.3.11/ 249
E_825C	OEM Unique ID Register (SFP_OUIDR)	32	R/W	0000_0000h	6.2.3.12/ 250
E_8268	OEM Scratch Pad Fuse Register n (SFP_OSPFR0)	32	R/W	0000_0000h	6.2.3.13/ 250
E_826C	OEM Scratch Pad Fuse Register n (SFP_OSPFR1)	32	R/W	0000_0000h	6.2.3.13/ 250
E_8270	Freescale Unique ID Register (SFP_FUIDR)	32	R/W	See section	6.2.3.14/ 251
E_8278	Freescale Scratch Pad Fuse Register n (SFP_FSPFR0)	32	R/W	0000_0000h	6.2.3.15/ 251
E_827C	Freescale Scratch Pad Fuse Register n (SFP_FSPFR1)	32	R/W	0000_0000h	6.2.3.15/ 251

6.2.3.1 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.

Address: E_8000h base + 20h offset = E_8020h



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.

Table continues on the next page...

SFP_INGR field descriptions (continued)

Field	Description
24–29 -	This field is reserved. Reserved
30–31 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.

6.2.3.2 Secret Value Hamming Error Status Register (SFP_SVHESR)

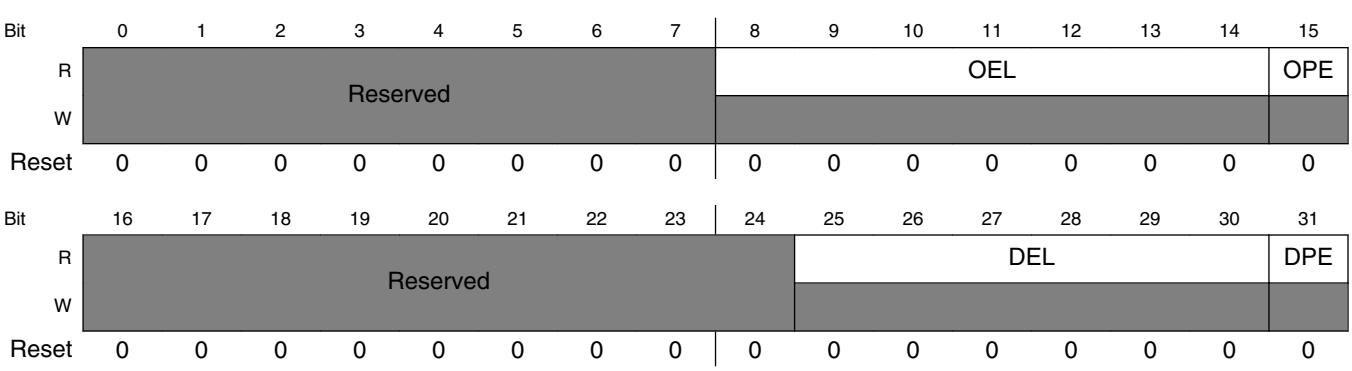
NOTE

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: E_8000h base + 24h offset = E_8024h



SFP_SVHESR field descriptions

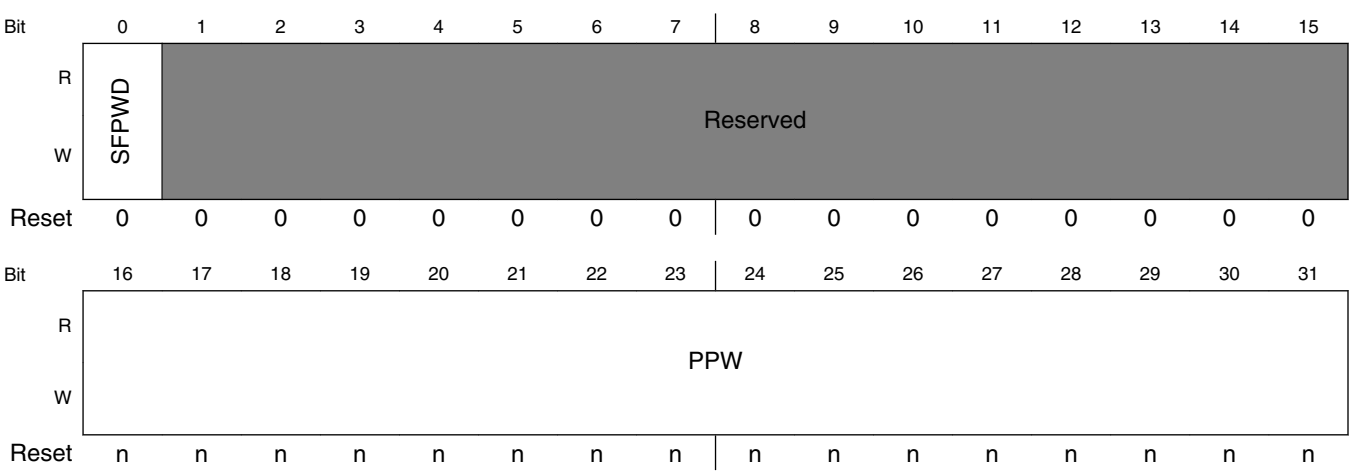
Field	Description
0–7 -	This field is reserved. Reserved
8–14 OEL	OTPMKR error location. Hamming code syndrome bits for OTPMKR. 000000 OTPMKR is error free (OPE == 0) or OTPMKR contains more than one error (OPE == 1). Non-zero OTPMKR contains an error at location encoded in OEL.
15 OPE	OTPMKR parity error. 0 OTPMKR is error free (OEL == 0) or OTPMKR contains more than one error (OEL != 0). 1 OTPMKR contains at least one error.
16–24 -	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.

6.2.3.3 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.

Address: E_8000h base + 28h offset = E_8028h



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.
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.

6.2.3.4 SFP Version Register (SFP_VERSION)

VERSION contains the current version of the secure fuse processor.

Address: E_8000h base + 34h offset = E_8034h

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

6.2.3.5 OEM Security Policy Register (SFP_OSPR)

NOTE

This register is protected by the OEM write protect bits. See OEM Security Policy Register (SFP_OSPR) and Freescale Section Write Protect Register (SFP_FSWPR) for more information on write protection.

This register configures critical security policy decisions which 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 device 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.

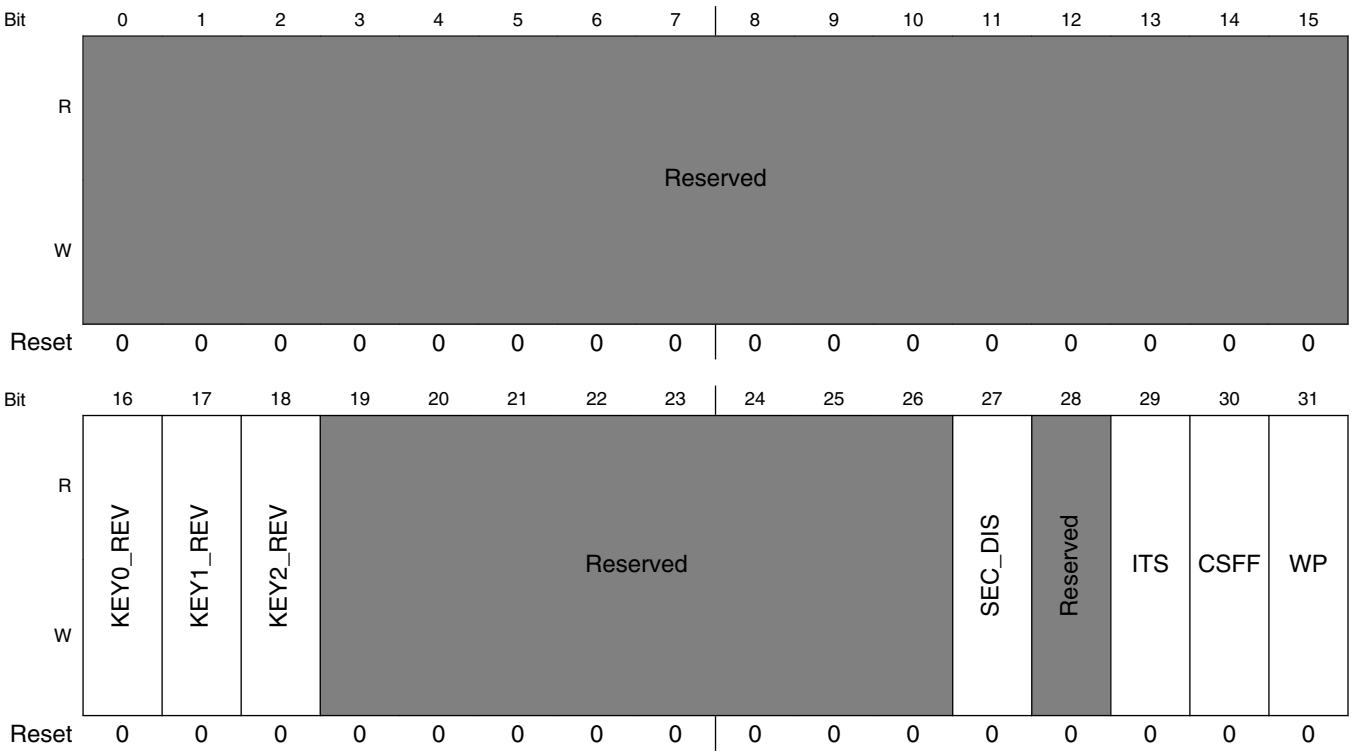
Sensitive flip flops are any flops holding secret keys, or intermediate plaintext data that is not eligible for leaving the security monitor, security engine (SEC), SFP, or SDC. Clear Sensitive Flip Flops (CSFF) is a security policy decision related to blocking expert level debug modes collectively known 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 Freescale 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 device reset. Once the WP fuse is programmed, writes to mirror registers and further programming of the fuse block is permanently disabled.

The device is available from Freescale in pin compatible export-controlled (SEC enabled) and non-export-controlled (SEC disabled) versions. Freescale 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 device (SEC enabled), and permanently convert it to a SEC disabled device when the presence of an export controlled processor

in the system complicates system level export control. Note that the device's part markings will indicate an export controlled device, 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. In Trust 2.0 devices, setting SEC_DIS disables export controlled confidentiality algorithms. Non-export controlled functionality (hashing, including HMAC, random number generation, and CRCs) are not disabled. Users should only set (SEC_DIS) if they intend to permanently disable the confidentiality algorithms. Note that SEC disabled devices cannot perform secure boot.

Address: E_8000h base + 200h offset = E_8200h



SFP_OSPR field descriptions

Field	Description
0–15 -	This field is reserved. Reserved
16 KEY0_REV	Key 0 Revoke. 0 Super root key 0 is not revoked. 1 Super root key 0 is revoked.
17 KEY1_REV	Key 1 Revoke. 0 Super root key 1 is not revoked. 1 Super root key 1 is revoked.
18 KEY2_REV	Key 2 Revoke.

Table continues on the next page...

SFP_OSPR field descriptions (continued)

Field	Description
	0 Super root key 2 is not revoked. 1 Super root key 2 is revoked.
19–26 -	This field is reserved. Reserved
27 SEC_DIS	Setting this bit will permanently disable the confidentiality algorithms of the SEC
28 -	This field is reserved. Reserved
29 ITS	Intent to Secure.
30 CSFF	Clear Sensitive Flip Flops.
31 WP	Write Protect OEM Section of SFP.

6.2.3.6 Freescale Section Write Protect Register (SFP_FSWPR)

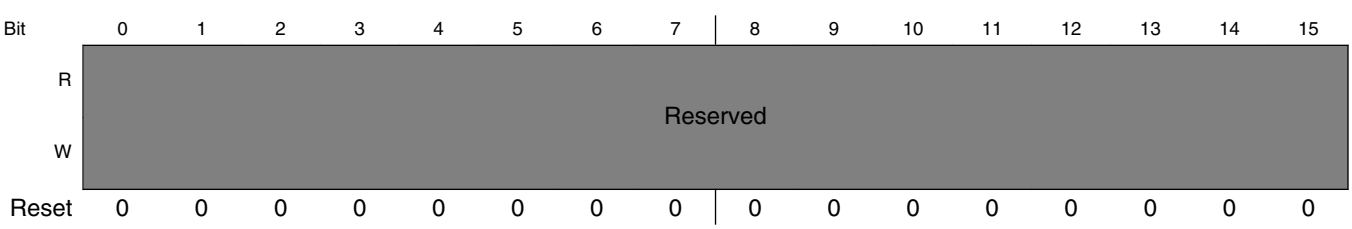
Freescale programs the Write Protect (WP) fuse bit on all devices sold. This prevents further writes to the FSL section of the SFP, including mirror registers and associated fuses.

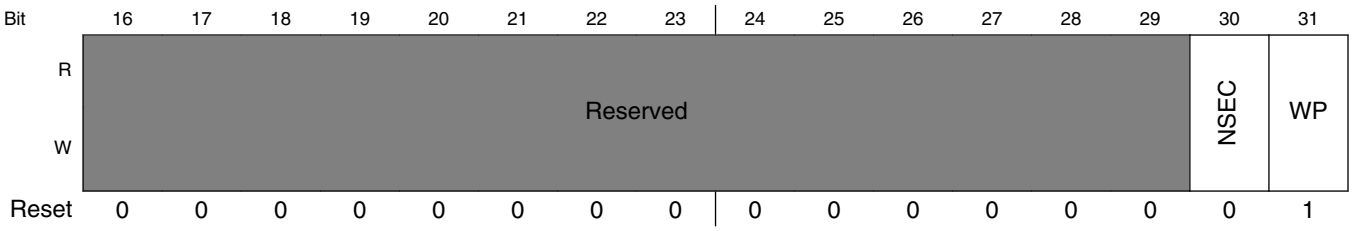
NOTE

This register is protected by the Freescale write protect bits. See OEM Security Policy Register (SFP_OSPR) and Freescale Section Write Protect Register (SFP_FSWPR) for more information on write protection.

The device 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. Note that secure boot can still be used in devices without the SEC, however without the ability to use the OTPMK or ZMK.

Address: E_8000h base + 204h offset = E_8204h





SFP_FSWPR field descriptions

Field	Description
0–29 -	This field is reserved. Reserved
30 NSEC	When NSEC is cleared, it indicates that this device has active SEC. When NSEC is set, it indicates that this device does not have an active SEC. **NOTE:** Writable until programmed into fuse array.
31 WP	Set to 1 by Freescale to enforce Write Protection of Freescale section of SFP.

6.2.3.7 Debug Permissions Register (SFP_DPR)

The DPR is set by OEMs to configure the debug access permissions for the secure debug controller (SDC).

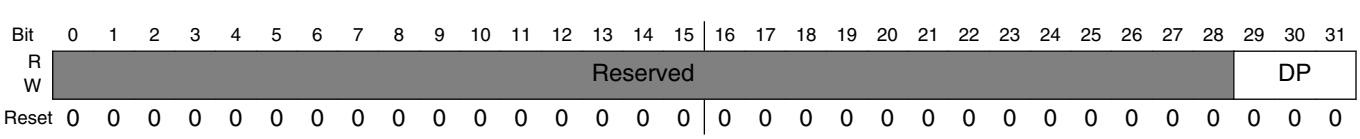
NOTE

This register is protected by the OEM write protect bits. See OEM Security Policy Register (SFP_OSPR) and Freescale Section Write Protect Register (SFP_FSWPR) for more information on write protection.

NOTE

Debug permissions can still be set in SEC-disabled devices. A SEC-disabled device 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: E_8000h base + 208h offset = E_8208h



SFP_DPR field descriptions

Field	Description
0–28 -	This field is reserved. Reserved
29–31 DP	Debug Permissions A detailed description of these permission levels can be found in Secure debug controller . 000 Open 001 Conditionally closed, without notification. 01x Conditionally closed, with notification. 1xx Closed.

6.2.3.8 Debug Challenge Value Register n (SFP_DCVR*n*)

NOTE

This register is protected by the OEM write protect bits. See OEM Security Policy Register (SFP_OSPR) and Freescale Section Write Protect Register (SFP_FSWPR) for more information on write protection.

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 device 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: c0c0c0c0_d0d0d0d0

Write to DCVR 1: c0c0c0c0

Write to DCVR 0: d0d0d0d0

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: E_8000h base + 20Ch 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	DCVR1 is the upper part (bits 0-31) and DCVR0 is the lower part (bits 32-63) of the 64-bit debug challenge value. Both of these registers reset to 0x0000_0000 unless programmed into the fuse array.

6.2.3.9 Debug Response Value Register n (SFP_DRVR*n*)

NOTE

This register is protected by the OEM write protect bits. See OEM Security Policy Register (SFP_OSPR) and Freescale Section Write Protect Register (SFP_FSWPR) for more information on write protection.

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 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).

DRVR0: key bits 32-63

DRVR1: key bits 0-31

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 (31,47,55,59,61,62,63) with the error detection code, as shown in the following reference code:

```
#include stdio.h
```

```
#include stdlib.h

#define bit(value,N) ((value >> N) 0x1)
#define uint64 unsigned long long

extern unsigned debug;

// Generate the Hamming code bits for the 64 bits stored in Number
// according to sfp's ECC algorithm.
int Generate_code_bits(uint64 and Number)
{
// response must be masked out the hamming coding bits first
uint64 mask = 0;
for(int i = 0; i < 64; i = (2*(i+1) - 1))
{
mask |= ((uint64) 1 << i);
}
Number = Number  ~mask;

uint64 mask = ((uint64) 1 << i);
unsigned parity = bit(Number,i);
if (debug > 1)
printf ("hamming bits %d, %d--\n\t", i, parity);
for(int j = i+1; j < 64; j++)
{
if (debug > 1)
printf (" %d/%d", j, bit(Number,j));
parity ^= bit(Number,j);
}
}
Number = (Number  ~mask) | ((uint64) parity << i);
if (debug > 1)
printf ("\nparity = %d, mask = %llx, new number = %llx\n\n", parity, mask, Number);
}

mask = ((uint64) 1 << 63);

// Calculate the overall parity
unsigned parity = 0;
for(int j = 0; j < 64; j++)
{
parity ^= bit(Number,j);
}

Number = (Number  ~mask) | ((uint64) parity << 63);

if (debug > 1)
printf ("parity = %d, final number = %llx\n\n", parity, Number);

return 0;
}
```

Example:

DRV with Hamming code applied and written to the DRVRs:

Original 64b value: 01aa00bb_cafecafe

After applying hamming code: 01aa00bb_cafeca77

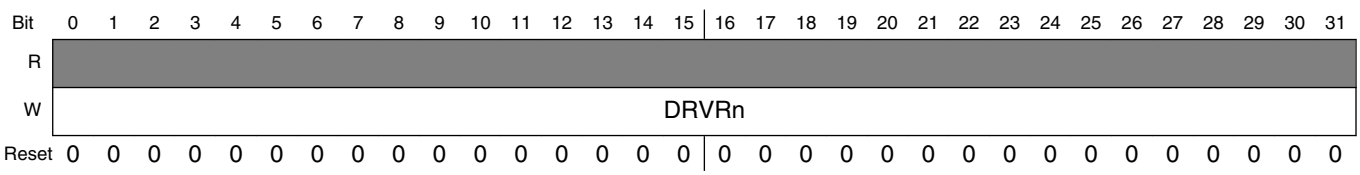
Write to DRVR0: cafeca77

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. Three failed challenge/response operations trigger a security violation, and lock the SDC against further challenge/response cycles until the device is reset.

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 device can permanently prevent the OEM from using device's debug resources. Freescale does not have a super user debug response value.

Address: E_8000h base + 214h offset + (4d × i), where i=0d to 1d



SFP_DRVR*n* field descriptions

Field	Description
0–31 DRVRn	DRVR0 is the upper part and DRVR1 is the lower part of the 64-bit debug response value according to this formula for each bit field range [(n × 32):((n × 32) + 31)]. Reset to 0x0000_0000 unless programmed into fuse array. DRVR0: key bits 32-63 DRVR1: key bits 0-31

6.2.3.10 One Time Programmable Master Key n (SFP_OTPMKR*n*)

NOTE

This register is protected by the OEM write protect bits. See OEM Security Policy Register (SFP_OSPR) and Freescale Section Write Protect Register (SFP_FSWPR) for more information on write protection.

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, as shown in the following reference code:

```
// Generate the Hamming code bits for the 256 bits
// stored in Number. The values at the locations of the code bits
// are ignored and is overwritten with the generated values.
Generate_code_bits(bool Number[256])
{
int i, j;

// Calculate each code bit in turn
for(i = 1; i <= 128; i = (i << 1)) {
// Examine each data bit
// Only bits greater than i need to be checked as no
// bit less than i will ever be XOR'ed into i
// J starts at i so that Number[i] is initialized to 0
for(j = i; j <= 255; j = j + 1) {
if ( (i & j) != 0) {
Number[i] = Number[i] ^ Number[j];
}
}
}
// Calculate the overall parity
// J starts at 0 so that Number[0] is initialized to 0
// Number[0] contains the even parity of all of the bits
for(j = 0; j <= 255; j = j + 1) {
Number[0] = Number[0] ^ Number[j];
}
```

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

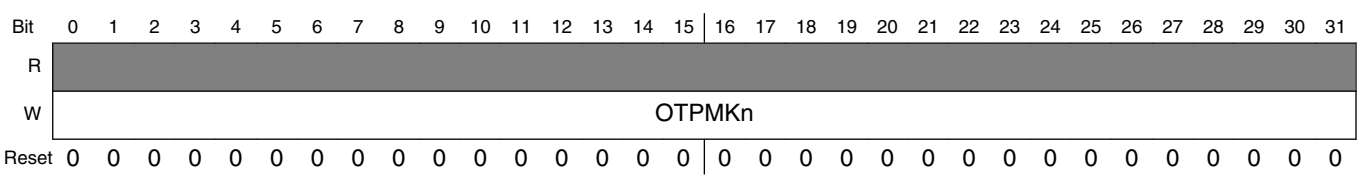
OTPMKR4: key bits 127-96

OTPMKR5: key bits 95-64

OTPMKR6: key bits 63-32

OTPMKR7: key bits 31-0

Address: E_8000h base + 21Ch offset + (4d × i), where i=0d to 7d



SFP_OTPMKR*n* field descriptions

Field	Description
0–31 OTPMKn	• OTPMKR0: key bits 255-224

SFP_OTPMKR*n* field descriptions (continued)

Field	Description
	• 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.

6.2.3.11 Super Root Key Hash n (SFP_SRKHR*n*)

NOTE

This register is protected by the OEM write protect bits. See OEM Security Policy Register (SFP_OSPR) and Freescale Section Write Protect Register (SFP_FSWPR) for more information on write protection.

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: E_8000h base + 23Ch 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 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.

6.2.3.12 OEM Unique ID Register (SFP_OUIDR)

NOTE

This register is protected by the OEM write protect bits. See OEM Security Policy Register (SFP_OSPR) and Freescale Section Write Protect Register (SFP_FSWPR) for more information on write protection.

The OUIDR is set by OEMs to configure an unalterable 32-bit software readable value which can be (optionally) included as part of the ESBC for the purposes of digital signature validation. It is up to the OEM to determine whether each device is provisioned with a truly unique ID.

Note that to bind an image to a specific device, the FUIDR must also be included in the ESBC for digital signature validation. Otherwise it would be possible for an attacker with access to a new chip to program the OUIDR to match the value found in a fielded chip.

Address: E_8000h base + 25Ch offset = E_825Ch

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	OEM_Unique_ID																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SFP_OUIDR field descriptions

Field	Description
0–31 OEM_Unique_ID	Reset to all zeros unless programmed into fuse array.

6.2.3.13 OEM Scratch Pad Fuse Register n (SFP_OSPFR*n*)

The OSPFRs 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_OSFPR0 is protected by the OEM write protect bits. See OEM Security Policy Register (SFP_OSPR) and Freescale Section Write Protect Register (SFP_FSWPR) for more information on write protection.

Address: E_8000h base + 268h 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	OEM_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_OSPFR*n* field descriptions

Field	Description
0–31 OEM_Scratch_ PAD_FUSES	OEM Scratch Pad Fuses

6.2.3.14 Freescale Unique ID Register (SFP_FUIDR)

NOTE

This register is protected by the Freescale write protect bits. See OEM Security Policy Register (SFP_OSPR) and Freescale Section Write Protect Register (SFP_FSWPR) for more information on write protection.

The FUIDR is set (and write protected) by Freescale prior to device shipment to provision a globally unique 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 device. OEM can consider this to be an unalterable globally unique value for the purposes of device identification.

Address: E_8000h base + 270h offset = E_8270h

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_Unique_ID																															
Reset	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n

SFP_FUIDR field descriptions

Field	Description
0–31 FS_Unique_ID	Reset to globally unique value.

6.2.3.15 Freescale 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. Freescale may use this register for non-tamperable storage of configuration data. OEMs may observe non-zero values in these registers.

NOTE

This register is protected by the Freescale write protect bits. See OEM Security Policy Register (SFP_OSPR) and Freescale Section Write Protect Register (SFP_FSWPR) for more information on write protection.

Address: E_8000h base + 278h 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	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	Freescale Scratch Pad Fuses

6.3 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.

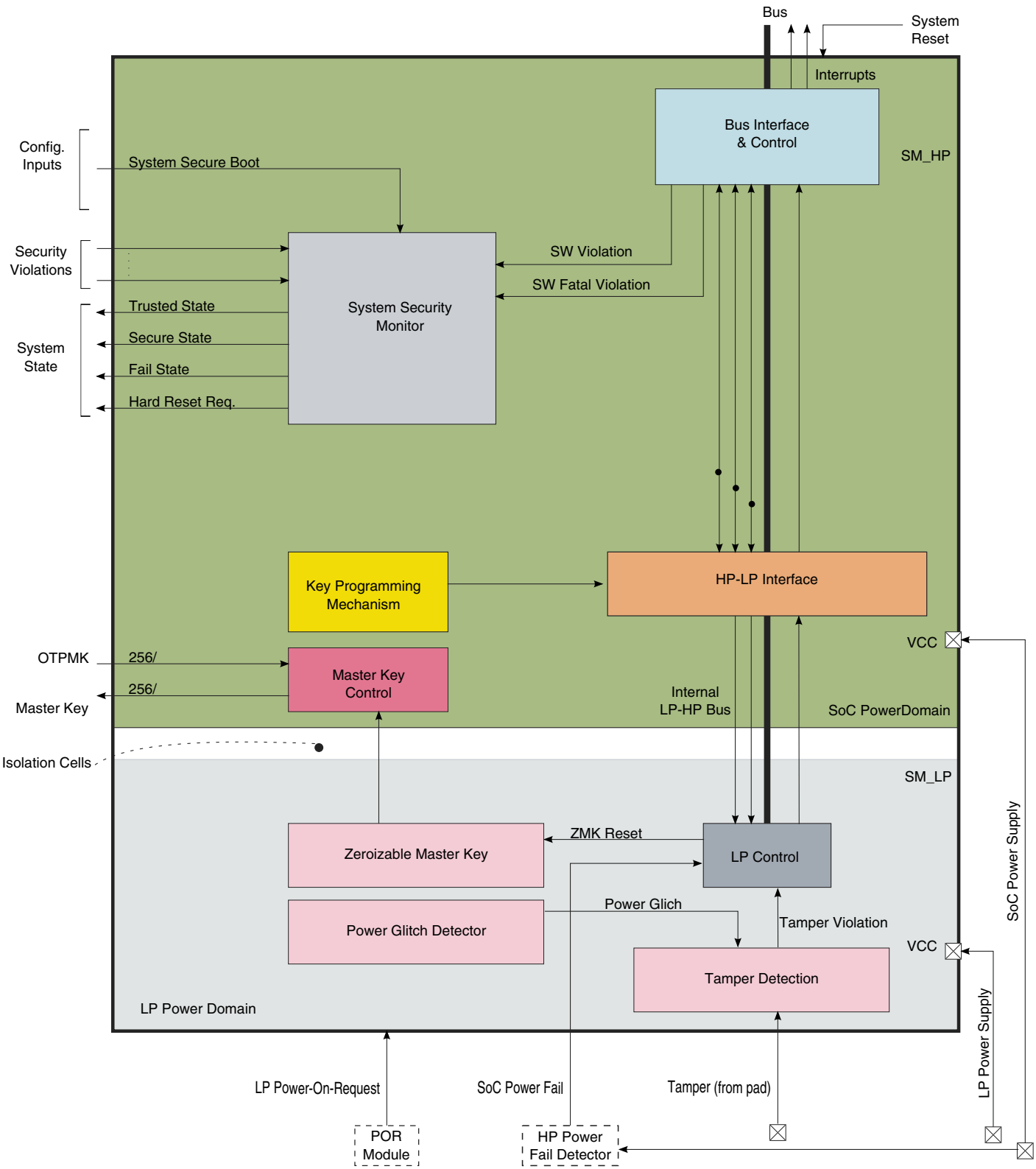


Figure 6-45. Security Monitor Block Diagram

6.3.1 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 state is a battery-backed persistent secret value
  - 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 by HW cryptographic accelerators
  - The key value will be zeroized in case of a critical security violation
- Secured Monotonic Counter
  - The Monotonic Counter state is a battery-backed persistent secret value
  - The counter can only increment
  - The counter value is invalidated in case of a critical security violation
- General Purpose Register
  - The General Purpose Register state is nonvolatile
- 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
    - Invalid OTPMK (ECC check failure)
    - ZMK ECC check failure
    - Monotonic counter roll-over
  - 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, force zeroization of sensitive information,
- Reports to software (interrupts) security violations

6.3.2 SM Modes of Operation

Security Monitor operates in either the system power-down or system power-up mode of operation. During system power-down, SM_HP is powered-down. SM_LP is powered from the backup power supply and is electrically isolated from the rest of the chip. In this mode, SM_LP keeps its registers' values and monitors the SM_LP tamper detection inputs.During system power-up, SM_HP and SM_LP are both powered-up and all Security Monitor functions are operational.

6.3.3 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.

6.3.4 Signals

This section provides information on the external signals.

Table 6-40. External Signals

Signal name	I/O	Description	
TMP_DETECT_B	I	The TMP_DETECT pin indicates to the high-power section of the security monitor that an external security event has occurred. Asserting this signal when SOC power or the system clock is off will have no effect. Asserting this signal when SOC power is on and the system clock is on may cause an interrupt to the core. Asserting this signal when the security monitor's security state machine is in trusted or secure state will cause an interrupt, but when the security state machine is in non-secure mode as interrupt will occur only if enabled via the HPSICR[SVI_EN3] setting.	
		State Meaning	This signal is active-high, so an external security event is signaled by asserting the pin high.
		Timing	Assertion/Negation - The signal may be asserted at any time asynchronous to any clock, but it must be asserted for a period equivalent to at least 3 system clocks.
LP_TMP_DETECT_B	I	The LP_TMP_DETECT_B pin indicates to the low-power (always-on) section of the security monitor that an external security event (as defined by the OEM) has occurred. The setting of the LPTDCR[ET1_EN] bit determines whether asserting LP_TMP_DETECT_B is regarded as a tamper event by the low-power section of the security monitor. If SOC power is on, the HPSVCR[LPSV_CFG] field determines whether a tamper event in the low-power section causes a security state machine transition in the high-power section of the security monitor.	
		State Meaning	This signal is active-low, so an external security event is signaled by asserting the pin low.
		Timing	Assertion/Negation -The signal may be asserted at any time asynchronous to any clock, but it must be asserted for a period equivalent to at least 3 system clocks.

6.3.5 Security monitor memory map/register definition

This section contains detailed register descriptions for the security monitor registers. Each description includes a standard register diagram with an associated figure number. Details of register bit and field function follow the register diagrams, in bit order.

When more than one software partition is operating on the CPU cores, it is recommended that only one partition have access to the security monitor. Access to the security monitor's register space can be controlled by the core's MMUs and the PAMUs as described in Trust Architecture as implemented on the chip.

Note that most security monitor registers are only writeable when the security monitor is in a functional state. Functional states include:

- Non-secure state
- Trusted state
- Secure state

For more information on security monitor states see Operational states

SECMON memory map

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
31_4000	SM_HP Lock Register (SECMON_HPLR)	32	R/W	0000_0000h	6.3.5.1/259
31_4004	SM_HP Command Register (SECMON_HPCOMR)	32	R/W	0000_0000h	6.3.5.2/261
31_400C	SM_HP Security Interrupt Control Register (SECMON_HPSICR)	32	R/W	0000_0000h	6.3.5.3/264
31_4010	SM_HP Security Violation Control Register (SECMON_HPSVCR)	32	R/W	0000_0000h	6.3.5.4/266
31_4014	SM_HP Status Register (SECMON_HPSR)	32	R	8000_0000h	6.3.5.5/268
31_4018	SM_HP Security Violation Status Register (SECMON_HPSVSR)	32	w1c	0000_0000h	6.3.5.6/270
31_401C	SM_HP High Assurance Counter Initial Value Register (SECMON_HPHACIVR)	32	R/W	0000_0000h	6.3.5.7/271
31_4020	SM_HP High Assurance Counter Register (SECMON_HPHACR)	32	R	0000_0000h	6.3.5.8/272
31_4034	SM_LP Lock Register (SECMON_LPLR)	32	R/W	0000_0000h	6.3.5.9/273
31_4038	SM_LP Control Register (SECMON_LPCR)	32	R/W	0000_0000h	6.3.5.10/ 275
31_403C	SM_LP Master Key Control Register (SECMON_LPMKCR)	32	R/W	0000_0000h	6.3.5.11/ 276
31_4040	SM_LP Security Violation Control Register (SECMON_LPSVCR)	32	R/W	0000_0000h	6.3.5.12/ 278
31_4048	SM_LP Tamper Detectors Configuration Register (SECMON_LPTDCR)	32	R/W	0000_0000h	6.3.5.13/ 279
31_404C	SM_LP Status Register (SECMON_LPSR)	32	w1c	0000_0000h	6.3.5.14/ 281
31_405C	SM_LP Secure Monotonic Counter MSB Register (SECMON_LPSMCMR)	32	R/W	0000_0000h	6.3.5.15/ 282
31_4060	SM_LP Secure Monotonic Counter LSB Register (SECMON_LPSMCLR)	32	R/W	0000_0000h	6.3.5.16/ 283

Table continues on the next page...

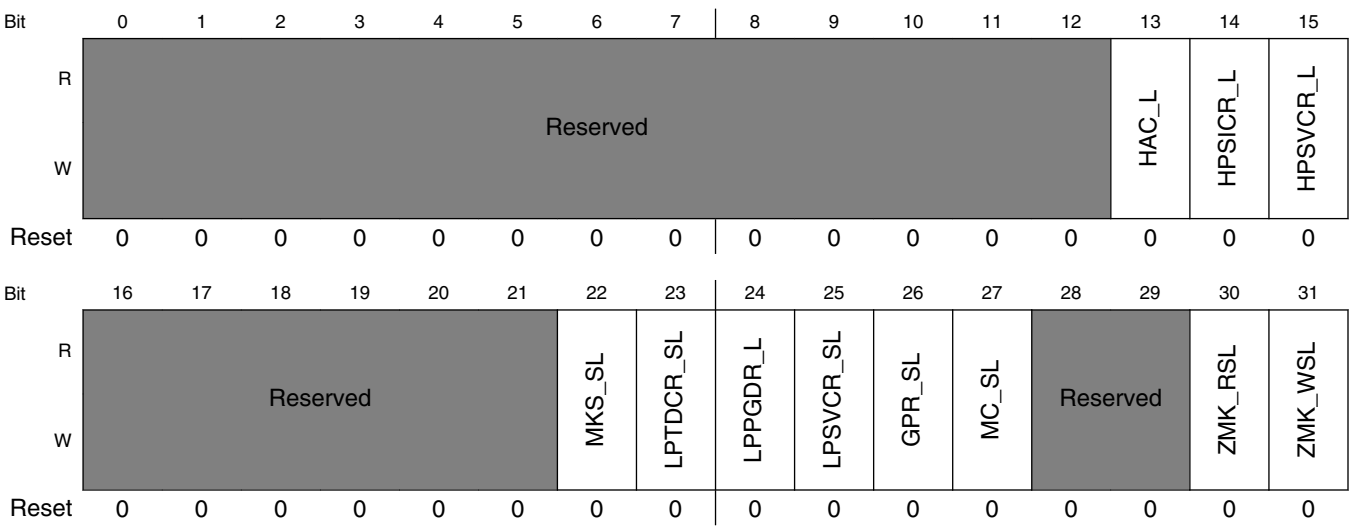
SECMON memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
31_4064	SM_LP Power Glitch Detector Register (SECMON_LPPGDR)	32	R/W	0000_0000h	6.3.5.17/ 283
31_4068	SM_LP General Purpose Register 0 Alias (SECMON_LPGPR0_ALIAS)	32	R/W	0000_0000h	6.3.5.18/ 284
31_406C	SM_LP Zeroizable Master Key Register (SECMON_LPZMKR0)	32	R/W	0000_0000h	6.3.5.19/ 284
31_4070	SM_LP Zeroizable Master Key Register (SECMON_LPZMKR1)	32	R/W	0000_0000h	6.3.5.19/ 284
31_4074	SM_LP Zeroizable Master Key Register (SECMON_LPZMKR2)	32	R/W	0000_0000h	6.3.5.19/ 284
31_4078	SM_LP Zeroizable Master Key Register (SECMON_LPZMKR3)	32	R/W	0000_0000h	6.3.5.19/ 284
31_407C	SM_LP Zeroizable Master Key Register (SECMON_LPZMKR4)	32	R/W	0000_0000h	6.3.5.19/ 284
31_4080	SM_LP Zeroizable Master Key Register (SECMON_LPZMKR5)	32	R/W	0000_0000h	6.3.5.19/ 284
31_4084	SM_LP Zeroizable Master Key Register (SECMON_LPZMKR6)	32	R/W	0000_0000h	6.3.5.19/ 284
31_4088	SM_LP Zeroizable Master Key Register (SECMON_LPZMKR7)	32	R/W	0000_0000h	6.3.5.19/ 284
31_4090	SM_LP General Purpose Register n (SECMON_LPGPR0)	32	R/W	0000_0000h	6.3.5.20/ 285
31_4094	SM_LP General Purpose Register n (SECMON_LPGPR1)	32	R/W	0000_0000h	6.3.5.20/ 285
31_4098	SM_LP General Purpose Register n (SECMON_LPGPR2)	32	R/W	0000_0000h	6.3.5.20/ 285
31_409C	SM_LP General Purpose Register n (SECMON_LPGPR3)	32	R/W	0000_0000h	6.3.5.20/ 285
31_4BF8	SM_HP Version ID Register 1 (SECMON_HPVIDR1)	32	R	See section	6.3.5.21/ 286
31_4BFC	SM_HP Version ID Register 2 (SECMON_HPVIDR2)	32	R	See section	6.3.5.22/ 286

6.3.5.1 SM_HP Lock Register (SECMON_HPLR)

The HPLR contains lock bits for the security monitor registers. See Initialization guidelines for information about the relationship between this register and other Security Monitor registers.

Address: 31_4000h base + 0h offset = 31_4000h



SECMON_HPLR field descriptions

Field	Description
0–12 -	This field is reserved. Reserved
13 HAC_L	High assurance configuration lock. When set, prevents any writes to HAC_EN bit of HPCOMR. Once set, this bit can only be reset by the system reset. 0 Write access is allowed. 1 Write access is not allowed.
14 HPSICR_L	HP security interrupt control register lock. When set, prevents any writes to HPSICR. Once set, this bit can only be reset by the system reset. 0 Write access is allowed. 1 Write access is not allowed.
15 HPSVCR_L	HP security violation control register lock. When set, prevents any writes to HPSVCR. Once set, this bit can only be reset by the system reset. 0 Write access is allowed. 1 Write access is not allowed.
16–21 -	This field is reserved. Reserved

Table continues on the next page...

SECMON_HPLR field descriptions (continued)

Field	Description
22 MKS_SL	Master Key Select Soft Lock. When set, prevents any writes to the MASTER_KEY_SEL field of the LPMKCR. Once set, this bit can only be reset by the system reset. 0 Write access is allowed. 1 Write access is not allowed.
23 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. 0 Write access is allowed. 1 Write access is not allowed.
24 LPPGDR_L	LP Power Glitch Detector Lock. When set, prevents any writes to the LPPGDR. Once set, this bit can only be reset by the system reset. 0 Write access is allowed. 1 Write access is not allowed.
25 LPSVCR_SL	LP Security Violation Control Register Soft Lock. When set, prevents any writes to the LPSVCR. Once set, this bit can only be reset by the system reset. 0 Write access is allowed. 1 Write access is not allowed.
26 GPR_SL	General Purpose Register Soft Lock. When set, prevents any writes to the GPR. Once set, this bit can only be reset by the system reset. 0 Write access is allowed. 1 Write access is not allowed.
27 MC_SL	Monotonic Counter Soft Lock. When set, prevents any writes (increments) to the MC Registers and MC_ENV bit. Once set, this bit can only be reset by the system reset. 0 Write access (increment) is allowed. 1 Write access (increment) is not allowed.
28–29 -	This field is reserved. Reserved
30 ZMK_RSL	Zeroizable Master Key Read Soft Lock. When set, prevents any SW reads to the ZMK Registers and ZMK_ECC_VALUE field of the LPMKCR. In the ZMK HW Programming mode (ZMK_HWP is set) the ZMK and ZMK_ECC_VALUE cannot be read by SW. Regardless of the setting of this bit the ZMK value can be used by HW when ZMK is selected. Once set, this bit can only be reset by the system reset. 0 Read access is allowed (only in SW Programming mode) 1 Read access is not allowed
31 ZMK_WSL	Zeroizable Master Key Write Soft Lock. When set, prevents any writes (SW and HW) to the ZMK Registers and MASTER_KEY_SEL, ZMK_HWP, ZMK_VAL, and ZMK_ECC_EN fields of the LPMKCR. Once set, this bit can only be reset by the system reset. 0 Write access is allowed. 1 Write access is not allowed.

6.3.5.2 SM_HP Command Register (SECMON_HPCOMR)

The HPCOMR contains command, configuration, and control bits of the security monitor. In addition to the write access in functional states some fields of this register can be also written in check and soft fail states.

Check:

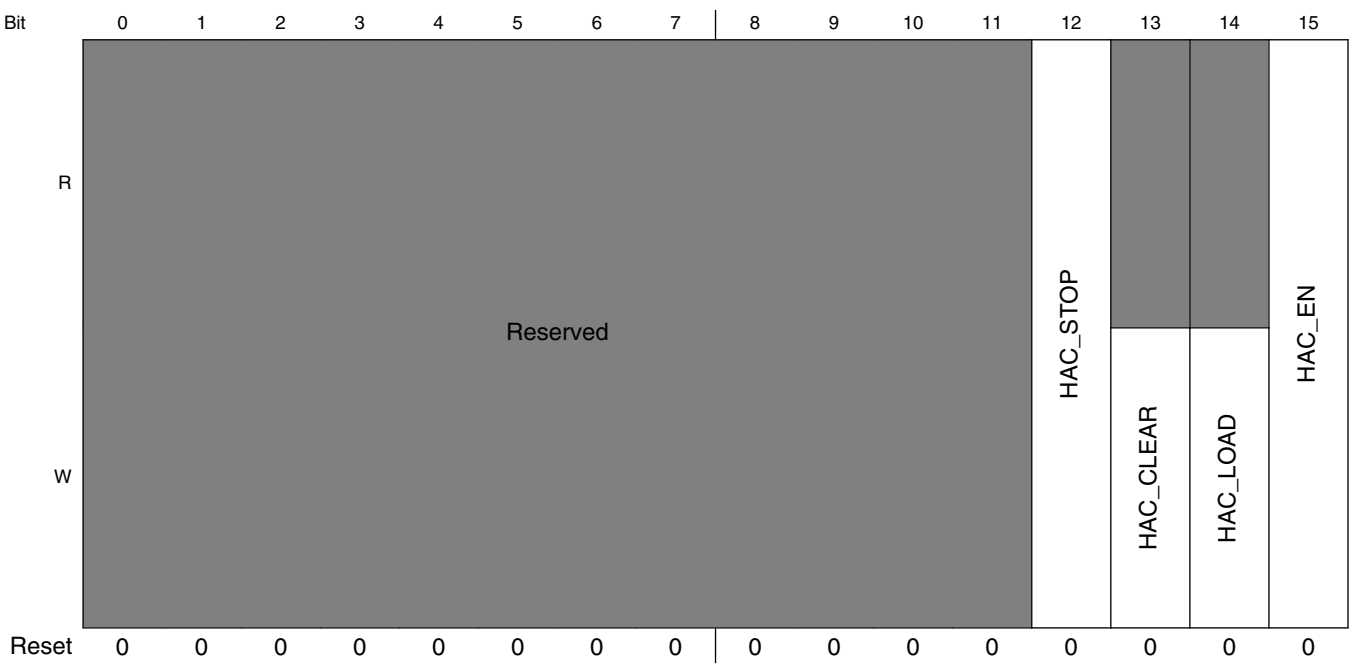
- SSM_ST, SW_SV, SW_FSV – these bits are writable by the ISBC, which is the only software running during Check state.

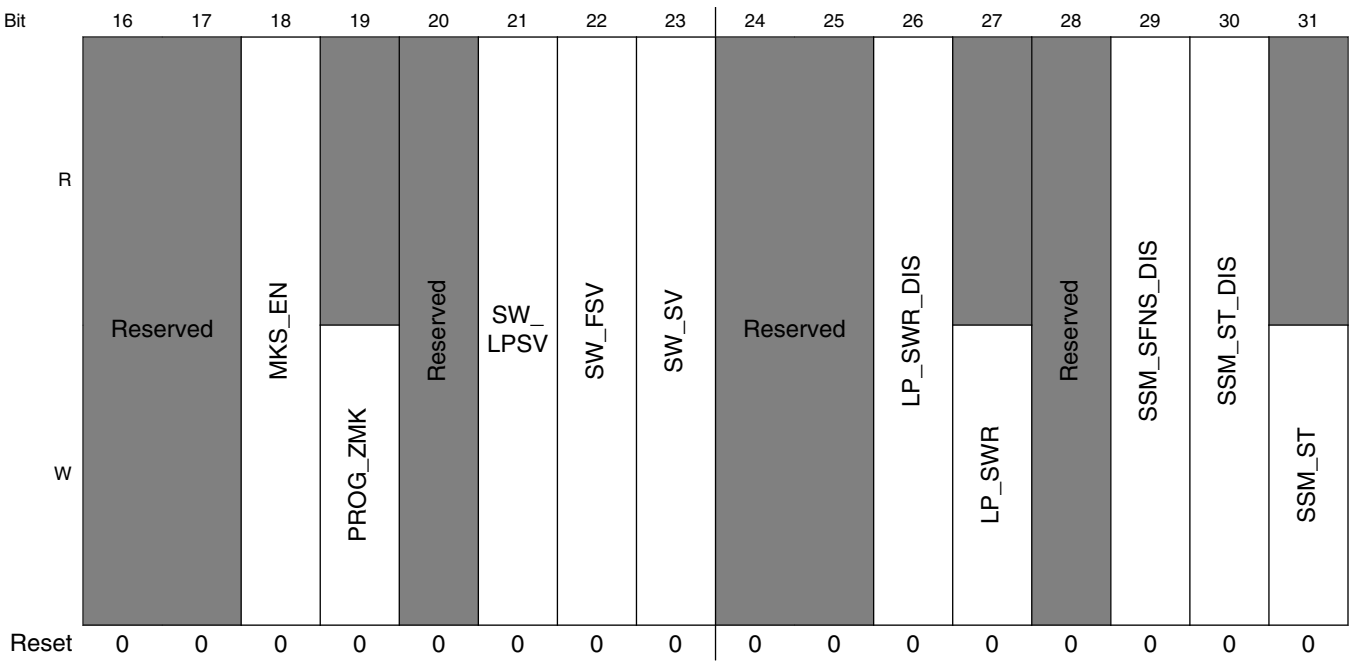
Soft Fail:

- SSM_ST, SW_SV, SW_FSV, HAC_STOP, HAC_LOAD, and HAC_CLEAR

See Initialization guidelines for information about the relationship between this register and other Security Monitor registers.

Address: 31_4000h base + 4h offset = 31_4004h





SECMON_HPCOMR field descriptions

Field	Description
0–11 -	This field is reserved. Reserved
12 HAC_STOP	High assurance counter (HAC) 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 one of the functional or soft fail states. If the bit is cleared in the soft fail state then high assurance counter will continue to count down from the place where it was stopped. 0 HAC can count down 1 HAC is stopped
13 HAC_CLEAR	High assurance counter clear. When set, this clears the high assurance counter register. This can be done in a functional state or in soft fail state. If the HAC is cleared in soft fail state the SSM will transition to hard fail state if high assurance configuration is enabled (HAC_EN is set). This self-clearing bit is always read as zero. 0 No action 1 Clear the high assurance counter
14 HAC_LOAD	High assurance counter load. When set, it loads the high assurance counter register with value of the high assurance counter load register. It can be done in functional state or in soft fail state. This self-clearing bit is always read as zero. 0 No action 1 Load the high assurance counter
15 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 high assurance counter before it expires, the SSM will transition to the "hard fail" state. This bit cannot be changed once HAC_L bit is set. 0 High assurance configuration is disabled 1 High assurance configuration is enabled

Table continues on the next page...

SECMON_HPCOMR field descriptions (continued)

Field	Description
16–17 -	This field is reserved. Reserved
18 MKS_EN	Master Key Select Enable. This bit controls the selection of the master key that is used to create cryptographic blobs. The security monitor can use the OTPMK or a key selected by LPMKCR[MASTER_KEY_SEL]. See SM_LP Master Key Control Register (SECMON_LPMKCR) for more information. Once set, this bit can only be reset by the system reset. 0 The security monitor uses the OTPMK as the master key 1 The security monitor master key is selected by the MASTER_KEY_SEL field of LPMKCR
19 PROG_ZMK	Program Zeroizable Master Key. This bit activates ZMK HW Programming Mechanism. This mechanism will be activated only if the ZMK is configured to the HW programming mode and ZMK in not locked for writes. This self-clearing bit is always read as zero. 0 No Action 1 Activate HW Key Programming Mechanism
20 -	This field is reserved. Reserved
21 SW_LPSV	LP SW Security Violation. When set, this bit is treated by SM_LP as a Security Violation. The LP Secure data will be zeroized or invalidated according to the configuration. This Security Violation may result in the System Security Monitor transition if the LP Security Violation is enabled in the SNVS_HP Security Violation Control Register.
22 SW_FSV	Software fatal security violation. When set this bit is treated by security monitor as a fatal security violation. This command results only in the following transitions of the SSM: • Check → soft fail • Non-secure → soft fail • Trusted → soft fail • Secure → soft fail
23 SW_SV	Software security violation. When set this bit is treated by security monitor as a non-fatal security violation. This command results only in the following transitions of the SSM: • Check → non-secure • Trusted → soft fail • Secure → soft fail
24–25 -	This field is reserved. Reserved
26 LP_SWR_DIS	LP SW Reset Disable. When set, disables the LP SW Reset. Once set, this bit can only be cleared by a chip power-on reset.. 0 LP SW Reset is enabled 1 LP SW Reset is disabled
27 LP_SWR	LP SW Reset. When set, it resets the SM_LP section, including zeroization of the ZMK, the GPRs, and the monotonic counter. This bit cannot be set when LP_SWR_DIS bit is set. This self-clearing bit is always read as zero. 0 No Action 1 Reset LP Section
28 -	This field is reserved. Reserved
29 SSM_SFNS_DIS	SSM soft fail to non-secure state transition disable. When set, disables the SSM to transition from soft fail to non-secure state. Once set after the reset this bit cannot be changed.

Table continues on the next page...

SECMON_HPCOMR field descriptions (continued)

Field	Description
	0 Soft fail to non-secure state transition is enabled 1 Soft fail to non-secure state transition is disabled
30 SSM_ST_DIS	SSM secure to trusted state transition disable. When set, disables the SSM to transition form secure to trusted state. Once set after the reset this bit can not be changed 0 Secure to trusted state transition is enabled 1 Secure to trusted state transition is disabled
31 SSM_ST	SSM state transition. Transition state of the security monitor. This self-clearing bit is always read as zero. This command results only in the following transitions of the SSM: • Check → non-secure (when non-secure boot) • Check → trusted (when secure boot) • Trusted → secure • Secure → trusted (if not disabled by SSM_ST_DIS bit) • Soft fail → non-secure (if not disabled by SSM_SFNS_DIS bit)

6.3.5.3 SM_HP Security Interrupt Control Register (SECMON_HPSICR)

The HPSICR defines security monitor security interrupt generation policy. See Initialization guidelines for information about the relationship between this register and other Security Monitor registers.

Address: 31_4000h base + Ch offset = 31_400Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	LPSVI_EN	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												SVI_EN3_TMP	SVI_EN2_SDC	SVI_EN1_SFP	SVI_EN0_RTIC
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SECMON_HPSICR field descriptions

Field	Description
0 LPSVI_EN	LP security violation interrupt enable. This bit enables generation of the security interrupt to the host processor upon security violation signal from the LP section.

Table continues on the next page...

SECMON_HPSICR field descriptions (continued)

Field	Description
	0 LP security violation interrupt is disabled 1 LP security violation interrupt is enabled
1–27 -	This field is reserved. Reserved
28 SVI_EN3_TMP	Security violation interrupt 3 enable. This bit enables generation of the security interrupt to the host processor upon security violation detection by the external tamper detect. 0 External tamper detect interrupt is Disabled 1 External tamper detect interrupt is Enabled
29 SVI_EN2_SDC	Security violation interrupt 2 enable. This bit enables generation of the security interrupt to the host processor upon security violation detection by the secure debug controller. 0 Secure Debug Controller interrupt is Disabled 1 Secure Debug Controller interrupt is Enabled
30 SVI_EN1_SFP	Security violation interrupt 1 enable. This bit enables generation of the security interrupt to the host processor upon security violation detection by the security fuse processor. 0 Security fuse processor interrupt is Disabled 1 Security fuse processor interrupt is Enabled
31 SVI_EN0_RTIC	Security violation interrupt 0 enable. This bit enables generation of the security interrupt to the host processor upon security violation detection by the run time integrity checker. 0 Run time integrity checker interrupt is Disabled 1 Run time integrity checker interrupt is Enabled

6.3.5.4 SM_HP Security Violation Control Register (SECMON_HPSVCR)

The HPSVCR defines type for each security violation input. See Initialization guidelines for information about the relationship between this register and other Security Monitor registers.

Address: 31_4000h base + 10h offset = 31_4010h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	LPSV_CFG		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												SV_CFG3_TMP	SV_CFG2_SDC	SV_CFG1_SFP	SV_CFG0_RTIC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SECMON_HPSVCR field descriptions

Field	Description
0–1 LPSV_CFG	LP security violation configuration. This field configures the LP security violation source. **NOTE:** Users who do not program a power glitch detection value in the LPPGDR should leave this field at its default value of 00 (LP security violation is disabled). 00 LP security violation is disabled 01 LP security violation is a non-fatal violation 1x LP security violation is a fatal violation
2–27 -	This field is reserved.
28 SV_CFG3_TMP	Security violation input 3 configuration. This field configures the security violation input on port 3. According to this setting the SSM knows how to respond when a Security Violation has been detected by the external tamper detect. 0 External tamper detect is a non-fatal violation 1 External tamper detect is a fatal violation
29 SV_CFG2_SDC	Security violation input 2 configuration. This field configures the security violation input on port 2 . According to this setting the SSM knows how to respond when a Security Violation has been detected by the secure debug controller.

Table continues on the next page...

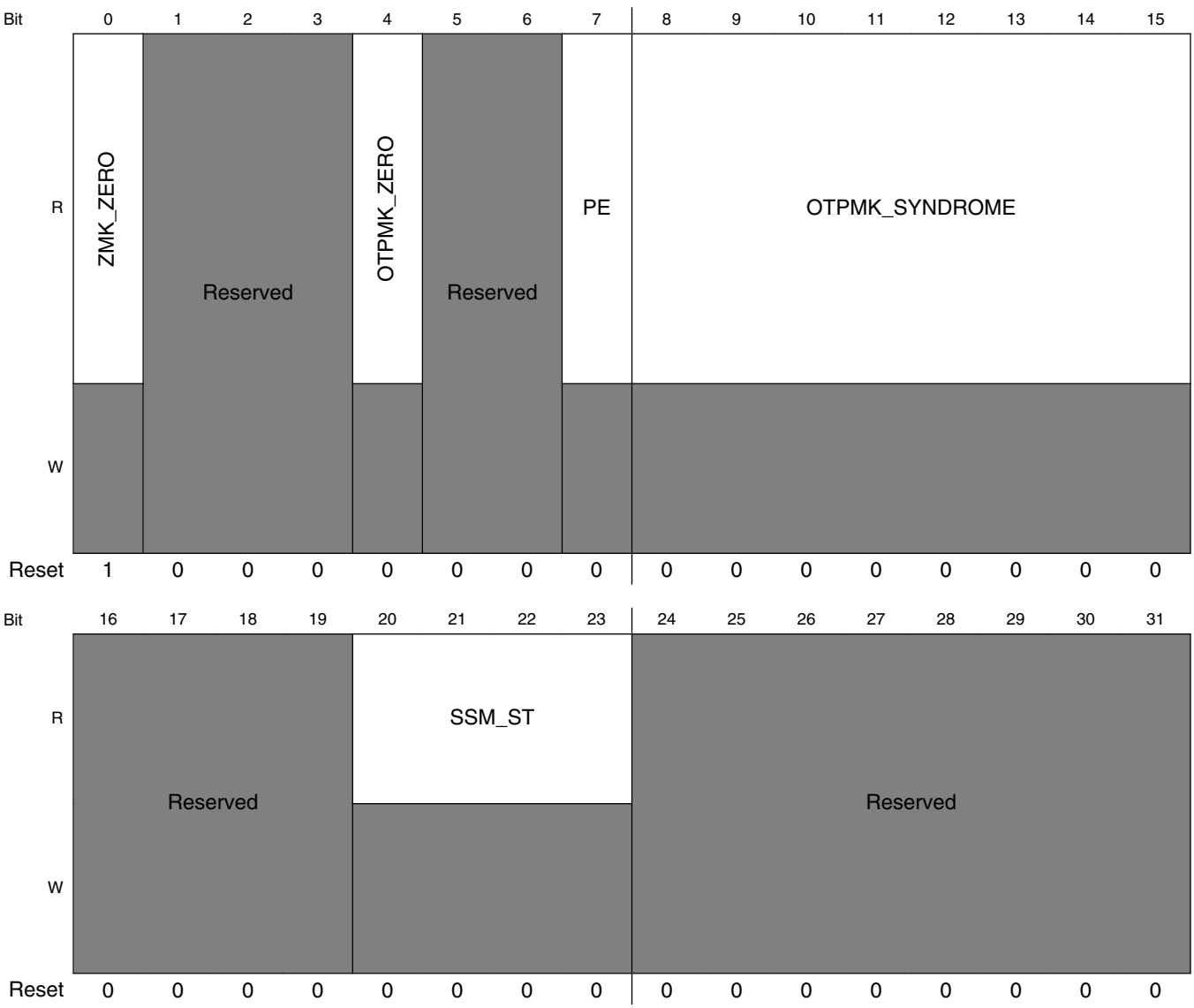
SECMON_HPSVCR field descriptions (continued)

Field	Description
	0 Secure debug controller is a non-fatal violation 1 Secure debug controller is a fatal violation
30 SV_CFG1_SFP	Security violation input 1 configuration. This field configures the security violation input on port 1 . AAccording to this setting the SSM knows how to respond when a Security Violation has been detected by the security fuse processor. 0 Security fuse processor is a non-fatal violation 1 Security fuse processor is a fatal violation
31 SV_CFG0_RTIC	Security violation input 0 configuration. This field configures the security violation input on port 0. According to this setting the SSM knows how to respond when a Security Violation has been detected by the run time integrity checker. 0 RTIC is a Non-Fatal Violation 1 RTIC is a Fatal Violation

6.3.5.5 SM_HP Status Register (SECMON_HPSR)

The HPSR reflects the internal state of the security monitor.

Address: 31_4000h base + 14h offset = 31_4014h



SECMON_HPSR field descriptions

Field	Description
0 ZMK_ZERO	Zeroizable Master Key is Equal to Zero. When set, this bit triggers “bad key” violation if theZMKis selected for use **NOTE:** The reset value of this bit depends on the value in the LPZMKR.

Table continues on the next page...

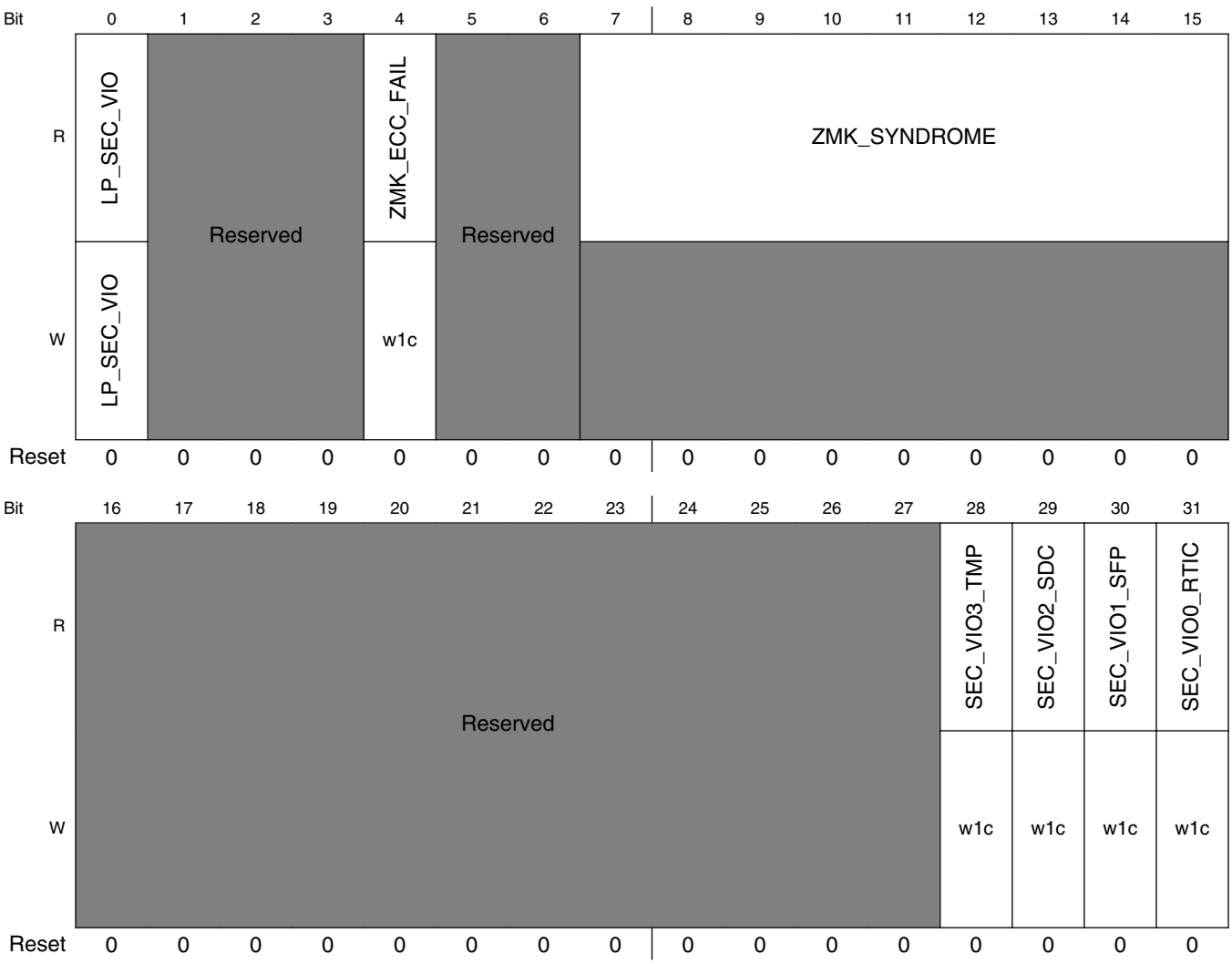
SECMON_HPSR field descriptions (continued)

Field	Description
	0 The ZMK is not zero 1 The ZMK is zero
1–3 -	This field is reserved. Reserved
4 OTPMK_ZERO	One Time Programmable Master Key = 0 Error 0 The OTPMK is not zero 1 The OTPMK is zero
5–6 -	This field is reserved. Reserved
7 PE	OTPMK Parity Error. This bit is set to '1' for any odd number of errors in the OTPMK, including errors in the error detection bits themselves. If any of the OTPMK_SYNDROME bits are set, and the OTRMK Parity Error = 0, then the OTPMK has 2 or more errors and the failing bit position cannot be determined.
8–15 OTPMK_ SYNDROME	This value indicates the error location in case of a single-bit error in the OTPMK. For example, syndrome word 10010110 indicates that key bit 150 has an error. See One Time Programmable Master Key n (SFP_OTPMKR*n*) for more information on OTPMK error detection.
16–19 -	This field is reserved. Reserved
20–23 SSM_ST	Security monitor state. This field contains the encoded state of the security monitor's internal state machine. The encoding of the possible states are: 0000 Init 1001 Check 1011 Non-Secure 1101 Trusted 1111 Secure 0011 Soft Fail 0001 Hard Fail
24–31 -	This field is reserved. Reserved

6.3.5.6 SM_HP Security Violation Status Register (SECMON_HPSVSR)

The HPSVSR reflects the HP domain security violation records. This register is writable in the Soft Fail state. See Initialization guidelines for information about the relationship between this register and other Security Monitor registers.

Address: 31_4000h base + 18h offset = 31_4018h



SECMON_HPSVSR field descriptions

Field	Description
0 LP_SEC_VIO	Security violation detected by the low power section.

Table continues on the next page...

SECMON_HPSVSR field descriptions (continued)

Field	Description
	0 No low power section security violation detected 1 Low power section security violation detected
1–3 -	Reserved This field is reserved.
4 ZMK_ECC_FAIL	Zeroizable Master Key Error Correcting Code Check Failure. When set, this bit triggers “bad key” violation to SSM and security violation to the SM_LP section, which clears Security Sensitive Data. Writing one to this bit clears record of this failure and also the ZMK_SYNDROME field of this register. 0 ZMK ECC Failure was not detected 1 ZMK ECC Failure was detected
5–6 -	This field is reserved. Reserved
7–15 ZMK_ SYNDROME	Zeroizable Master Key Syndrome Value. ZMK Syndrome indicates error location and parity similar to OTPMK Syndrome. This value is set and locked when ZMK ECC Failure is detected and cleared by writing one into ZMK_ECC_FAIL bit.
16–27 -	This field is reserved. Reserved
28 SEC_VIO3_TMP	Security violation detected by external tamper detect. See External tamper-detection for more information. 0 No external tamper detect security violation detected 1 External tamper detect security violation detected
29 SEC_VIO2_SDC	Security violation detected by secure debug controller. See Secure debug controller for more information. 0 No secure debug controller security violation detected 1 Secure debug controller security violation detected
30 SEC_VIO1_SFP	Security violation detected by security fuse processor. See Security fuse processor for more information. 0 No security fuse processor security violation detected 1 Security fuse processor security violation detected
31 SEC_VIO0_RTIC	Security violation detected by run-time integrity checker. See SEC 5.2 for more information. 0 No run-time integrity checker security violation detected 1 Run-time integrity checker security violation detected

6.3.5.7 SM_HP High Assurance Counter Initial Value Register (SECMON_HPHACIVR)

The HPHACIVR contains the initial value for the high assurance counter.

The high assurance counter decrements at the operating frequency of the Security Monitor, generally 400MHz (half of platform frequency). The maximum delay is ~10 seconds.

Address: 31_4000h base + 1Ch offset = 31_401Ch

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	HAC_COUNTER_IV																															
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

SECMON_HPHACIVR field descriptions

Field	Description
0–31 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.

6.3.5.8 SM_HP High Assurance Counter Register (SECMON_HPHACR)

The HPHACR 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.

Address: 31_4000h base + 20h offset = 31_4020h

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	HAC_COUNTER																															
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

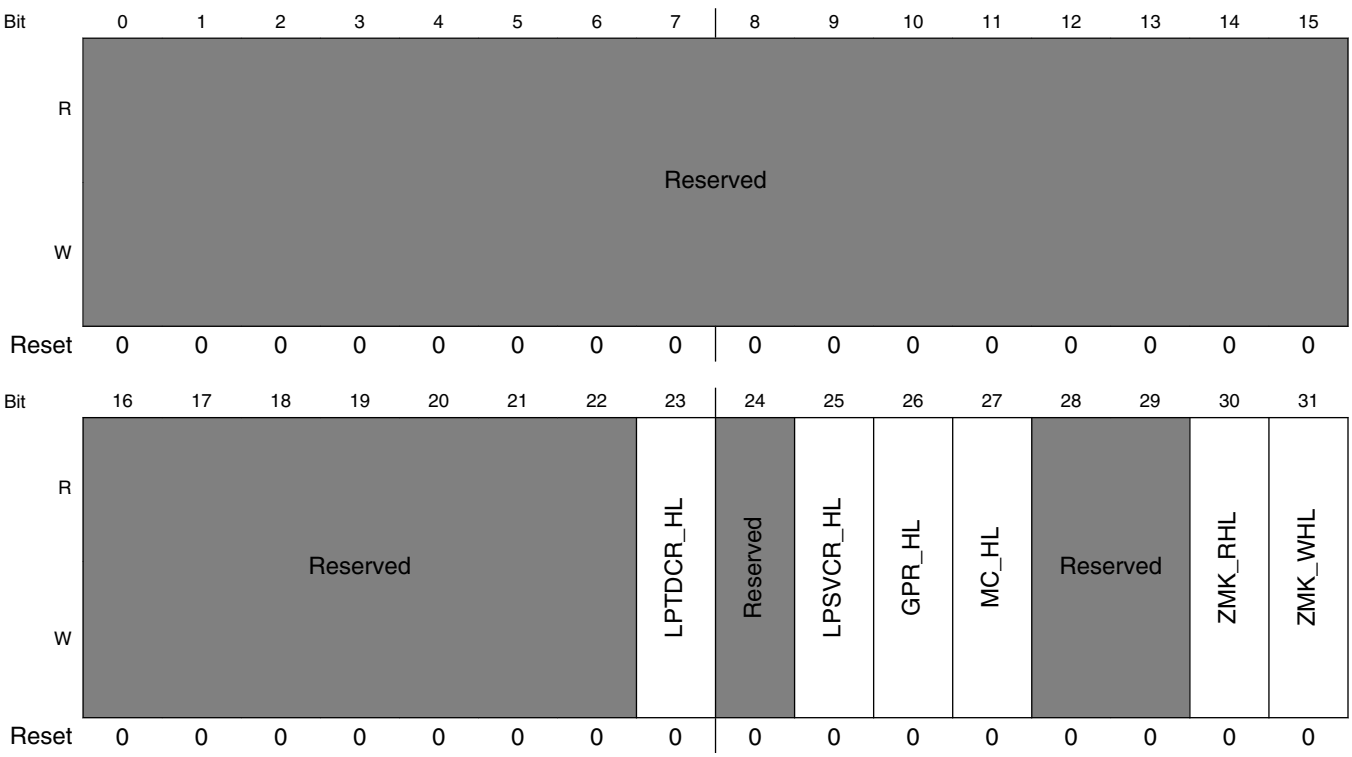
SECMON_HPHACR field descriptions

Field	Description
0–31 HAC_COUNTER	High assurance counter. When 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 counter reaches zero the SSM transition 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.

6.3.5.9 SM_LP Lock Register (SECMON_LPLR)

The SM_LP Lock Register contains lock bits for the SM_LP registers.

Address: 31_4000h base + 34h offset = 31_4034h



SECMON_LPLR field descriptions

Field	Description
0–22 —	This field is reserved. Reserved
23 LPTDCR_HL	LP Tamper Detectors Configuration Register Hard Lock. When set, prevents any writes to the LPTDCR. Once set, this bit can only be reset by the LP POR. 0 Write access is allowed 1 Write access is not allowed
24 -	This field is reserved. Reserved
25 LPSVCR_HL	LP Security Violation Control Register Hard Lock. When set, prevents any writes to the LPSVCR. Once set, this bit can only be reset by the LP POR. 0 Write access is allowed 1 Write access is not allowed
26 GPR_HL	General Purpose Register Hard Lock. When set, prevents any writes to the GPR. Once set, this bit can only be reset by the LP POR.

Table continues on the next page...

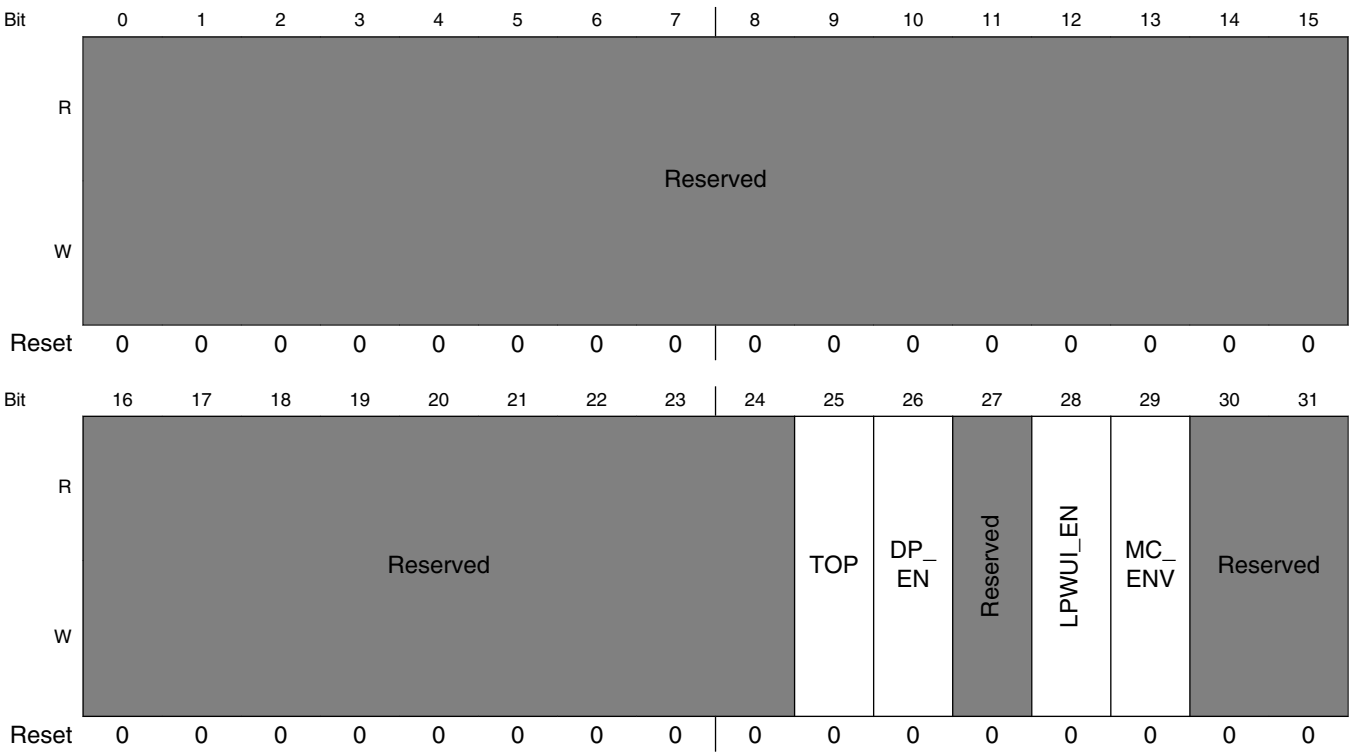
SECMON_LPLR field descriptions (continued)

Field	Description
	0 Write access is allowed 1 Write access is not allowed
27 MC_HL	Monotonic Counter Hard Lock. When set, prevents any writes (increments) to the MC Registers and MC_ENV bit. Once set, this bit can only be reset by the LP POR. 0 Write access (increment) is allowed 1 Write access (increment) is not allowed
28–29 -	This field is reserved. Reserved
30 ZMK_RHL	Zeroizable Master Key Read Hard Lock. When set, prevents any SW reads to the ZMK Registers and ZMK_ECC_VALUE field of the LPMKCR. In the ZMK HW Programming mode (ZMK_HWP is set) the ZMK and ZMK_ECC_VALUE cannot be read by SW. Regardless of the setting of this bit the ZMK value can be used by HW when ZMK is selected. Once set, this bit can only be reset by the LP POR. 0 Read access is allowed (only in SW Programming mode) 1 Read access is not allowed
31 ZMK_WHL	Zeroizable Master Key Write Hard Lock. When set, prevents any writes (SW and HW) to the ZMK Registers and ZMK_HWP, ZMK_VAL, and ZMK_ECC_EN fields of the LPMKCR. Once set, this bit can only be reset by the LP POR. 0 Write access is allowed 1 Write access is not allowed

6.3.5.10 SM_LP Control Register (SECMON_LPCR)

The SM_LP Control Register contains various control bits of the LP section of Security Monitor.

Address: 31_4000h base + 38h offset = 31_4038h



SECMON_LPCR field descriptions

Field	Description
0–24 -	This field is reserved. Reserved
25 TOP	Turn of System Power. Asserting this bit will cause a signal to be sent to the Power Management IC to turn off the system power. This bit will clear once power is off. This bit is only valid when the Dumb PMIC is enabled. 0 Leave system power on. 1 Turn system power off.
26 DP_EN	Dumb PMIC Enabled. When set the system power is controllable by software. When cleared the system requires a Smart PMIC to automatically turn power off. 0 Smart PMIC enabled. 1 Dumb PMIC enabled.
27 -	This field is reserved. Reserved

Table continues on the next page...

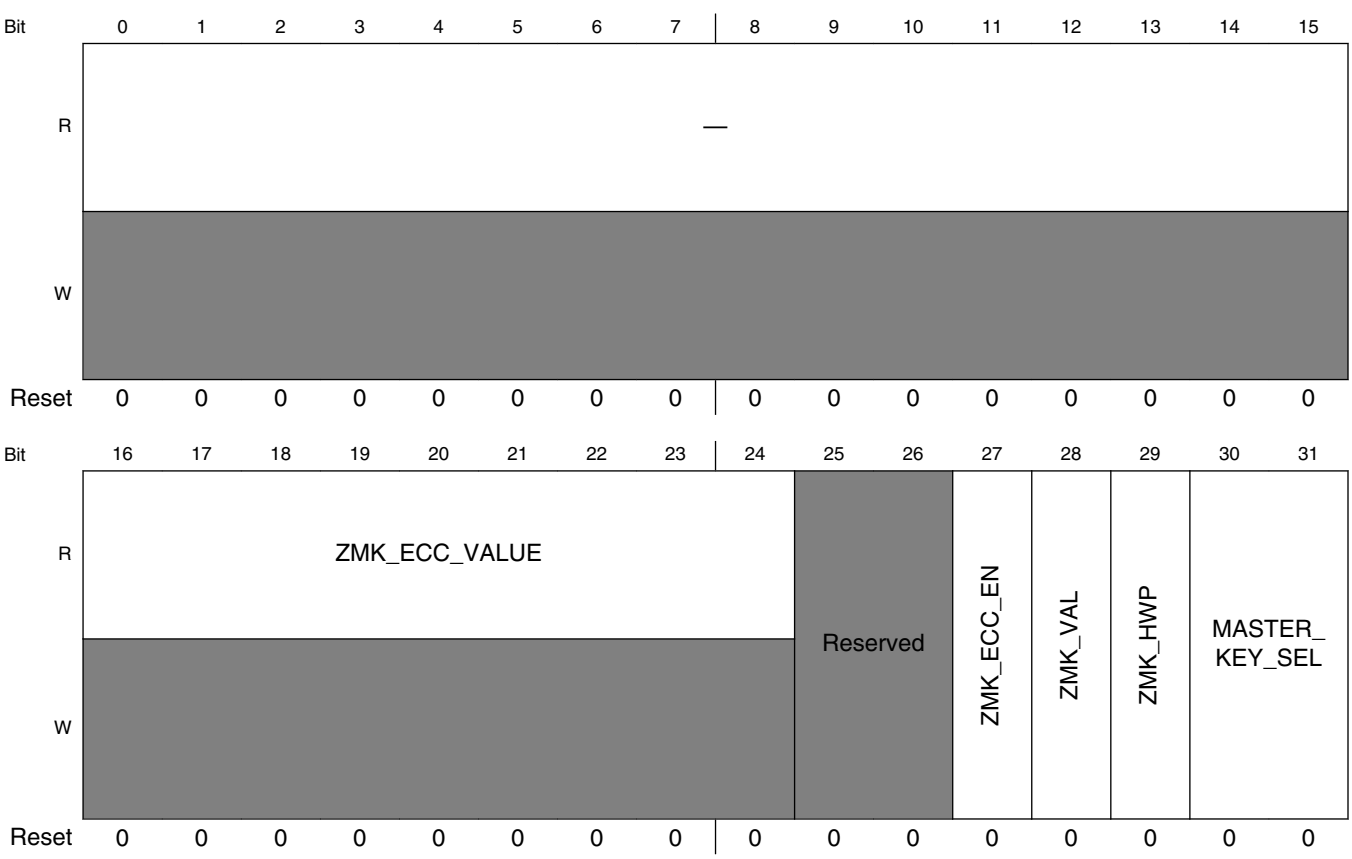
SECMON_LPCR field descriptions (continued)

Field	Description
28 LPWUI_EN	LP Wake-Up Interrupt Enable. This interrupt line should be connected to the external pin and is intended to inform the external chip about SM_LP event (tamper event, MC rollover,). This wake-up signal can be asserted only when the SoC (HP Section) is powered-down and the LP Section is isolated. 0 LP Wake-Up Interrupt is disabled. 1 LP Wake-Up Interrupt is enabled.
29 MC_ENV	Monotonic Counter Enable and Valid. When set the MC can be incremented (by write transaction to the LPSMCMR or LPSMCLR). This bit cannot be changed once MC_SL or MC_HL bit is set. 0 MC is disabled or invalid. 1 MC is enabled and valid.
30–31 -	This field is reserved. Reserved

6.3.5.11 SM_LP Master Key Control Register (SECMON_LPMKCR)

The SM_LP Lock Register contains contains Master Keys configuration.

Address: 31_4000h base + 3Ch offset = 31_403Ch



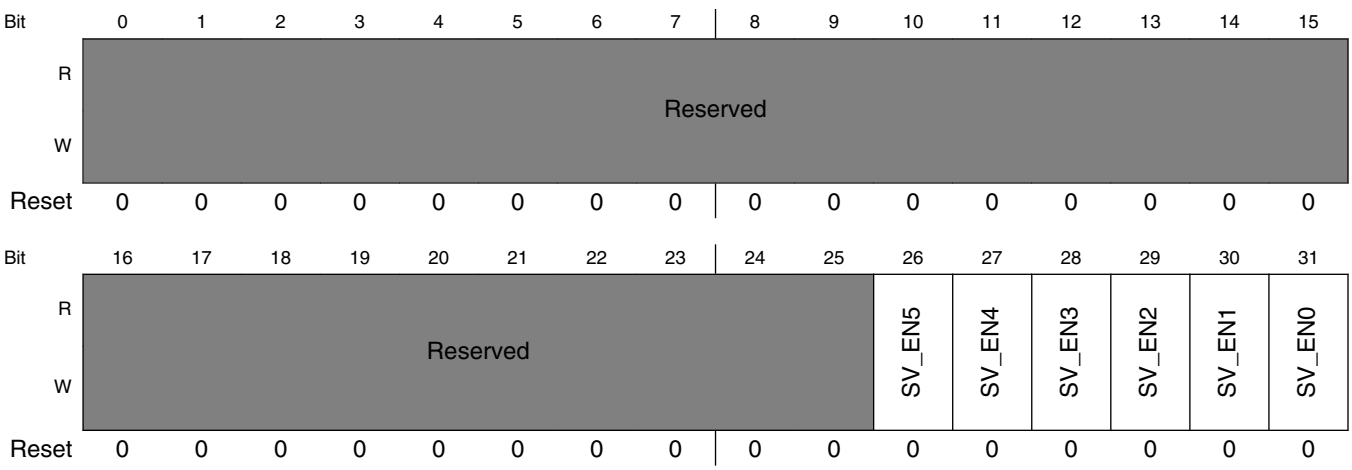
SECMON_LPMKCR field descriptions

Field	Description
0–15 —	Reserved
16–24 ZMK_ECC_VALUE	Zeroizable Master Key Error Correcting Code Value. This field is automatically calculated and set when one is written into ZMK_ECC_EN bit of this register. This field cannot be programmed by SW. It keeps ECC value of the Zeroizable Master Key, which allows checking that ZMK is not corrupted/altered with time. Note that this ZMK ECC code is equivalent to the ECC bits encoded into the OTPMK value but in case of ZMK this ECC value is kept separate from the ZMK value.
25–26 -	This field is reserved. Reserved
27 ZMK_ECC_EN	Zeroizable Master Key Error Correcting Code Check Enable. Writing one to this field automatically calculates and sets ZMK ECC Value in the ZMK_ECC_VALUE field of this register. When both, ZMK Value is valid (ZMK_VAL is set) and ZMK ECC Check is enabled (ZMK_ECC_EN is set) the ZMK Value is continuously checked to have valid ECC word. If ZMK ECC word calculated every clock cycle does not match one recorded in this register the ZMK ECC Check Fail Violation is generated. This bit cannot be programmed when ZMK_WSL bit is set. 0 ZMK ECC Check is Disabled 1 ZMK ECC Check is Enabled
28 ZMK_VAL	Zeroizable Master Key Valid. When set, the ZMK value can be selected by Master Key Control block for use by Cryptographic modules. In the HW Programming mode this bit is set by HW when the ZMK provisioning is complete. In the SW Programming mode this bit should be set by SW. This bit cannot be programmed when ZMK_WSL or ZMK_WHL bit is set. 0 ZMK is not Valid 1 ZMK is Valid
29 ZMK_HWP	Zeroizable Master Key HW Programming mode. When set, the ZMK can be programmed only by HW Key Programming Mechanism and it is not readable for SW. When not set, the ZMK can be programmed only by SW. This bit cannot be programmed when ZMK_WSL or ZMK_WHL bit is set. 0 ZMK is in the SW Programming Mode 1 ZMK is in the HW Programming Mode
30–31 MASTER_KEY_SEL	Master Key Select. These bits select the SNVS Master Key output when Master Key Select bits are enabled by MKS_EN bit in the HPCOMR. When MKS_EN bit is not set the One Time Programmable Master Key is selected by default. This field cannot be programmed when ZMK_WSL MKS_SL(or the hard lock) bit is set. 0x Select One Time Programmable Master Key 10 Select Zeroizable Master Key when MKS_EN bit is set 11 Select Combined Master Key when MKS_EN bit is set

6.3.5.12 SM_LP Security Violation Control Register (SECMON_LPSVCR)

The LP Security Violation Control Register configures Security Violation Inputs. This register cannot be programmed when LPSVCR Lock bit is set. Note that configurations of the security violation inputs in the HP Section (HPSVCR) and LP Section (LPSVCR Register) are independent and have different functionality.

Address: 31_4000h base + 40h offset = 31_4040h



SECMON_LPSVCR field descriptions

Field	Description
0–25 -	This field is reserved. Reserved
26 SV_EN5	Security Violation 5 Enable. This bit enables Security Violation Input 5. When this bit is set Security Violation 5 cause LP security violation, which clears LP sensitive data. 0 Security Violation 5 is Disabled in the LP Domain 1 Security Violation 5 is Enabled in the LP Domain
27 SV_EN4	Security Violation 4 Enable. This bit enables Security Violation Input 4. When this bit is set Security Violation 4 cause LP security violation, which clears LP sensitive data. 0 Security Violation 4 is Disabled in the LP Domain 1 Security Violation 4 is Enabled in the LP Domain
28 SV_EN3	Security Violation 3 Enable. This bit enables Security Violation Input 3. When this bit is set Security Violation 3 cause LP security violation, which clears LP sensitive data. 0 Security Violation 3 is Disabled in the LP Domain 1 Security Violation 3 is Enabled in the LP Domain
29 SV_EN2	Security Violation 2 Enable. This bit enables Security Violation Input 2. When this bit is set Security Violation 2 cause LP security violation, which clears LP sensitive data.

Table continues on the next page...

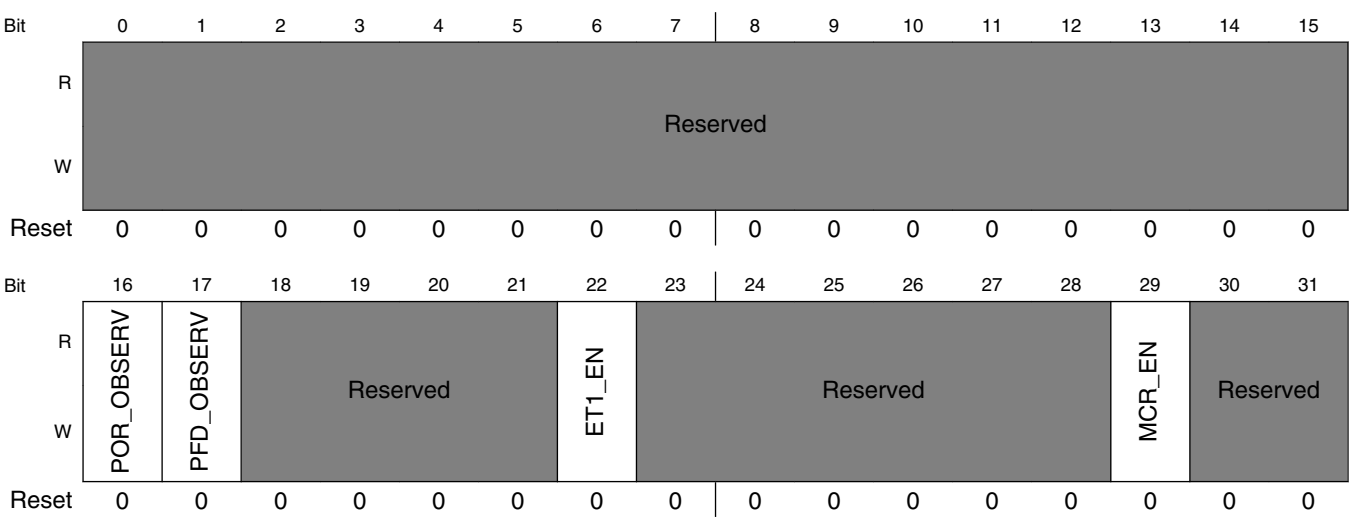
SECMON_LPSVCR field descriptions (continued)

Field	Description
	0 Security Violation 2 is Disabled in the LP Domain 1 Security Violation 2 is Enabled in the LP Domain
30 SV_EN1	Security Violation 1 Enable. This bit enables Security Violation Input 2. When this bit is set Security Violation 1 cause LP security violation, which clears LP sensitive data. 0 Security Violation 1 is Disabled in the LP Domain 1 Security Violation 1 is Enabled in the LP Domain
31 SV_EN0	Security Violation 0 Enable. This bit enables Security Violation Input 0. When this bit is set Security Violation 0 cause LP security violation, which clears LP sensitive data. 0 Security Violation 0 is Disabled in the LP Domain 1 Security Violation 0 is Enabled in the LP Domain

6.3.5.13 SM_LP Tamper Detectors Configuration Register (SECMON_LPTDCR)

The SM_LP Tamper Detectors Configuration Register is used to configure analog and digital tamper detector sources. This register cannot be programmed when LPTDCR is locked for write.

Address: 31_4000h base + 48h offset = 31_4048h



SECMON_LPTDCR field descriptions

Field	Description
0–15 -	This field is reserved. Reserved

Table continues on the next page...

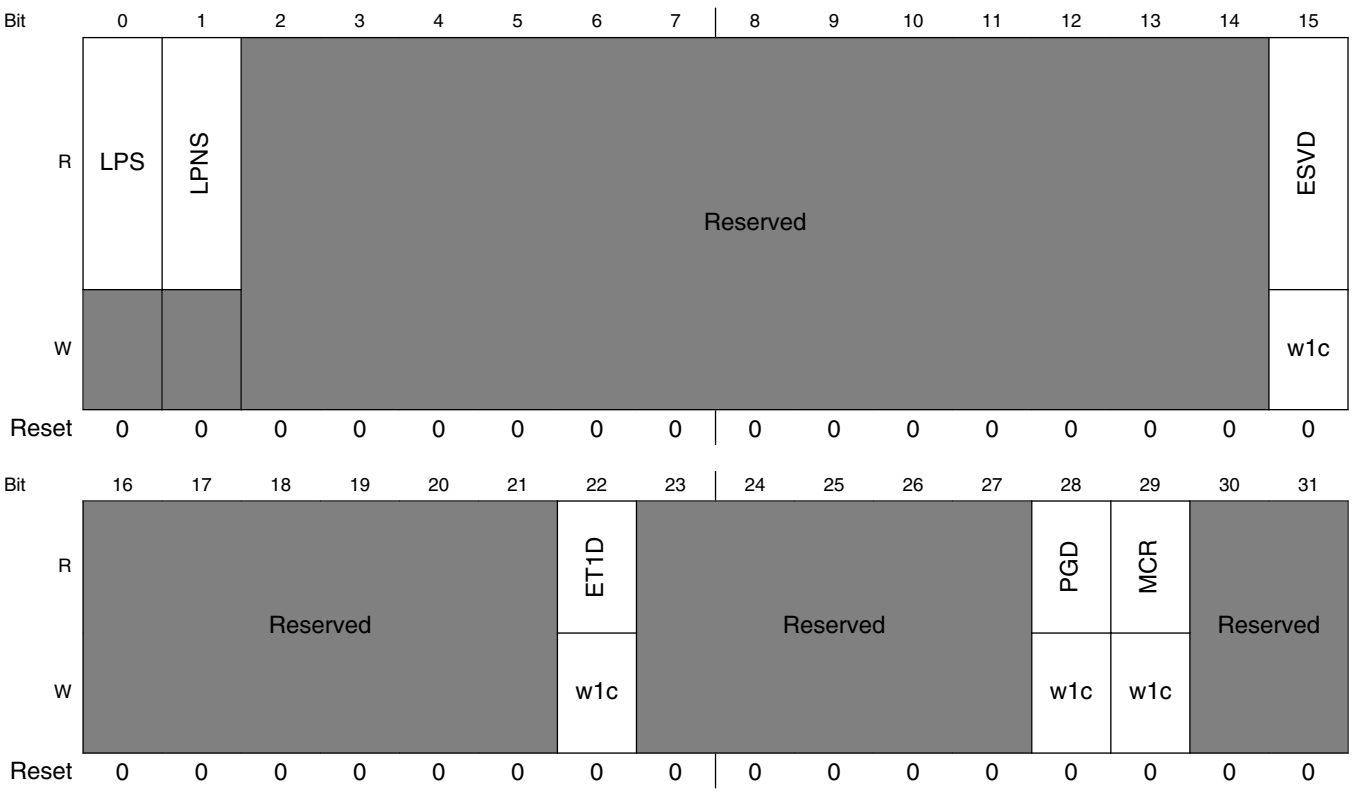
SECMON_LPTDCR field descriptions (continued)

Field	Description
16 POR_OBSERV	Power On Reset (POR) Observability Flop. The asynchronous reset input of this flop is connected directly to the output of the POR analog circuitry (external to the Security Monitor block). This flop can be used to detect brown-out voltage of the POR circuitry
17 PFD_OBSERV	System Power Fail Detector (PFD) Observability Flop. The asynchronous reset input of this flop is connected directly to the inverted output of the PFD analog circuitry (external to the Security Monitor block). This flop can be used to detect brown-out voltage of the PFD circuitry.
18–21 -	This field is reserved. Reserved
22 ET1_EN	External Tamper 1 (LP_TMP_DETECT_B) Enable. When set, External Tamper 1 (LP_TMP_DETECT_B) generates an LP Security Violation. 0 External Tamper 1 (LP_TMP_DETECT_B) is disabled 1 External Tamper 1 (LP_TMP_DETECT_B) is enabled
23–28 -	This field is reserved. Reserved
29 MCR_EN	MC Rollover Enable. When set, MC Rollover event generates LP Security Violation. 0 MC rollover is disabled 1 MC rollover is enabled
30–31 -	This field is reserved. Reserved

6.3.5.14 SM_LP Status Register (SECMON_LPSR)

The SM_LP status register reflects the internal state of the SM_LP, including LP security violations. If the LP sections violations are not disabled in the HPSVCR (SM_HP Security Violation Control Register (SECMON_HPSVCR)) scan and power glitch detection in the LP section will cause a SSM transition as defined in System security monitor . The LP security violations must be cleared before transitioning the SSM from Soft Fail → Non-Secure, provided that path is enabled.

Address: 31_4000h base + 4Ch offset = 31_404Ch



SECMON_LPSR field descriptions

Field	Description
0 LPS	LP section is secured. Indicates that LP section was provisioned/programmed in the secure or trusted state. When set the SM_LP section can not be programmed. 0 LP section was not programmed in secure or trusted state 1 LP section was programmed in secure or trusted state
1 LPNS	LP section is non-secured. Indicates that LP section was provisioned/programmed in the non-secure state. When set the SM_LP section will be cleared on the SSM transition from check to trusted state.

Table continues on the next page...

SECMON_LPSR field descriptions (continued)

Field	Description
	0 LP section was not programmed in the non-secure state 1 LP section was programmed in the non-secure state
2–14 -	This field is reserved. Reserved
15 ESVD	External Security Violation Detected. Indicates that Security Violation is detected on one of the HP Security Violation Ports. The record of the port on which the violation has occurred can be found in the HP Security Violation Status Register. 0 No External Security Violation 1 External Security Violation is detected
16–21 -	This field is reserved. Reserved
22 ET1D	External Tampering 1 (LP_TMP_DETECT_B) Detected. 0 External Tampering 1 (LP_TMP_DETECT_B) not detected 1 External Tampering 1 (LP_TMP_DETECT_B) detected
23–27 -	This field is reserved. Reserved
28 PGD	Power supply glitch detected. 0 No power supply glitch 1 Power supply glitch is detected
29 MCR	Monotonic counter rollover. 0 MC has not reached its maximum value. 1 MC has reached its maximum value.
30–31 -	This field is reserved. Reserved

6.3.5.15 SM_LP Secure Monotonic Counter MSB Register (SECMON_LPSMCMR)

The SM_LP Secure Monotonic Counter MSB Register contains the Monotonic Counter Era Bits and the most-significant 16 bits of the Monotonic Counter. The Monotonic Counter is incremented by one if there is a write command to the LPSMCMR or LPSMCLR Register.

Address: 31_4000h base + 5Ch offset = 31_405Ch

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	MC_ERA																MON_COUNTER															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SECMON_LPSMCMR field descriptions

Field	Description
0–15 MC_ERA	Monotonic Counter Era Bits. These bits are provided as inputs to the module and typically connected to the Non-Volatile Elements.
16–31 MON_COUNTER	Monotonic Counter most-significant 16 bits. The MC is incremented by one when a write transaction to the LPSMCMR or LPSMCLR Register is detected, the MC_ENV bit is set, and MC_SL and MC_HL bits are not set.

6.3.5.16 SM_LP Secure Monotonic Counter LSB Register (SECMON_LPSMCLR)

The SM_LP Secure Monotonic Counter LSB Register contains the least-significant 32 bits of the Monotonic Counter. The Monotonic Counter is incremented by one if there is a write command to the LPSMCMR or LPSMCLR Register.

Address: 31_4000h base + 60h offset = 31_4060h

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	MON_COUNTER																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SECMON_LPSMCLR field descriptions

Field	Description
0–31 MON_COUNTER	Monotonic Counter least-significant bits. The MC is incremented by one when a write transaction to the LPSMCMR or LPSMCLR Register is detected, the MC_ENV bit is set, and MC_SL and MC_HL bits are not set.

6.3.5.17 SM_LP Power Glitch Detector Register (SECMON_LPPGDR)

The SM_LP power glitch detector register is used for storing power glitch detector value as described in #d388e4a1310.

Address: 31_4000h base + 64h offset = 31_4064h

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	PGD																															
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

SECMON_LPPGDR field descriptions

Field	Description
0–31 PGD	Power glitch detector value.

6.3.5.18 SM_LP General Purpose Register 0 Alias (SECMON_LPGPR0_ALIAS)

The SM_LP General Purpose Register 0 alias provides a 32-bit read write register, that can be used by any application for retaining 32-bit data during a power-down mode.

NOTE

This register serves as an alias to LPGPR0.

Address: 31_4000h base + 68h offset = 31_4068h

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	GPR																															
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

SECMON_LPGPR0_ALIAS field descriptions

Field	Description
0–31 GPR	General Purpose Register. When GPR_SL or GPR_HL bit is set the register cannot be programmed.

6.3.5.19 SM_LP Zeroizable Master Key Register (SECMON_LPZMKR*n*)

The SM_LP Zeroizable Master Key Registers contain the 256-bit Zeroizable Master Key Value. These registers cannot be programmed when ZMK write lock bit is set. They can only be programmed by SW in the SW Programming mode (ZMK_HWP is not set) and by HW in the HW Programming mode (ZMK_HWP is set). These registers cannot be read by SW when ZMK_HWP or ZMK read lock bit is set.

Address: 31_4000h base + 6Ch 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	ZMK																															
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

SECMON_LPZMKR*n* field descriptions

Field	Description
0–31 ZMK	Zeroizable Master Key. Each of these registers contains part of the 256-bit ZMK Value: LPZMKR0 - ZMK[31:0] LPZMKR1 - ZMK[63:32] LPZMKR2 - ZMK[95:64] LPZMKR3 - ZMK[127:96] LPZMKR4 - ZMK[159:128] LPZMKR5 - ZMK[191:160] LPZMKR6 - ZMK[223:192] LPZMKR7 - ZMK[255:224]

6.3.5.20 SM_LP General Purpose Register n (SECMON_LPGPR*n*)

The SM_LP General Purpose Registers provide multiple 32-bit read/write registers, that can be used by any application for retaining data during a power-down mode.

Address: 31_4000h base + 90h offset + (4d × i), where i=0d to 3d

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	GPR																															
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

SECMON_LPGPR*n* field descriptions

Field	Description
0–31 GPR	General Purpose Register. When GPR_SL or GPR_HL bit is set the register cannot be programmed.

6.3.5.21 SM_HP Version ID Register 1 (SECMON_HPVIDR1)

The HPVIDR contains the current version of the security monitor. The version consists of a module ID, a major version number, and a minor version number.

Address: 31_4000h base + BF8h offset = 31_4BF8h

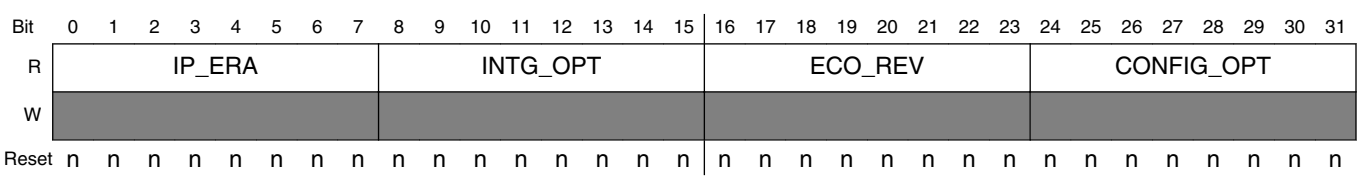
SECMON_HPVIDR1 field descriptions

Field	Description
0–15 IP_ID	Security monitor module ID = 0x003A
16–23 MAJOR_REV	Security monitor module major version number
24–31 MINOR_REV	Security monitor module minor version number

6.3.5.22 SM_HP Version ID Register 2 (SECMON_HPVIDR2)

The HPVIDR2 contains the current version of the security monitor. This version consists of ERA, integration, ECO, and configuration options of security monitor.

Address: 31_4000h base + BFCh offset = 31_4BFCh



SECMON_HPVIDR2 field descriptions

Field	Description
0–7 IP_ERA	IP ERA option
8–15 INTG_OPT	Security monitor integration option

Table continues on the next page...

SECMON_HPVIDR2 field descriptions (continued)

Field	Description
16–23 ECO_REV	Security monitor ECO revision
24–31 CONFIG_OPT	Security monitor configuration option

6.3.6 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.

In the Trust Architecture a portion of the Security Monitor can operate on battery power while the rest of the SoC is powered off. 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.

6.3.6.1 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.

6.3.6.1.1 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.

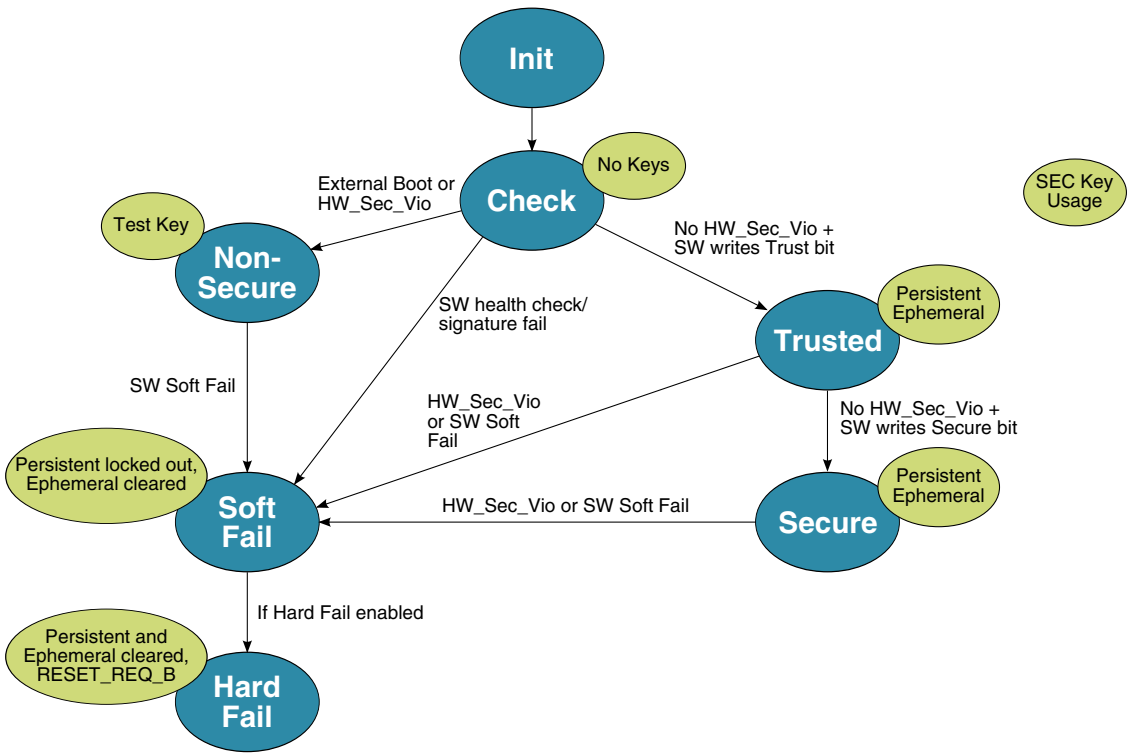


Figure 6-80. Security monitor states

6.3.6.1.2 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 Secure boot sequence.

  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 will transition the SSM to this state, and the non-secure system will be allowed to execute the suspect ESBC. Any requests by the ESBC for the SEC to use the OTPMK, ZMK, or KEK cause test keys to be used, rather than the real OTPMK , ZMK, or secret KEK. This state exists mainly to support debug.
- **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.

- **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 Freescale devices that also use the security monitor.

- **Soft Fail:** A security violation, reported by hardware or software, will cause a transition to Soft Fail state (except as previously described for the Check --> Non-Secure transition). Upon transitioning to Soft Fail, the Security Monitor signals the SEC to stop using the OTPMK, ZMK, and KEKs, and zeroize any ephemeral secrets within the SEC. The security monitor also generates an interrupt to inform software of the Soft Fail, and if configured to do so, starts the HAC_Counter count down toward Hard Fail. The device can operate with the SSM in the Soft Fail state

indefinitely; however, no operations involving the SEC can occur unless the SecMon is transitioned into the Non-Secure state, and the SEC has been reset and reinitialized. Even after reinitialization, the SEC is unable to perform operations involving the persistent secrets, or with the ephemeral secrets that existed prior to the entry into Soft Fail. The only transition back to Trusted/Secure state is through a HRESET or Power on Reset, which clears all internal states and returns the Security Monitor to Init.

Except for a few bits in the HP command register, the Security Monitor registers cannot be programmed in this state.

Transition from Soft Fail to the Non-Secure state can be triggered by software through a write to the HPCOMR SSM_ST bit, unless the HAC counter is actively counting or the transition to the Non-Secure state is disabled.

- **Hard Fail:** If the Security Monitor is configured for Hard Fail, upon expiration of the HAC_Counter, the Security Monitor will trigger a reset request to the platform, and initiate automatic zeroization of the platform caches and DDR. Security Monitor registers cannot be programmed in this state. The SSM remains in the Hard Fail state until the system is reset.

Table 6-75. System security state machine state transition table

Current state	Next state (transition to next state determined by conditions listed in the table cells)						
	Init	Check	Non-secure	Soft fail	Hard fail	Trusted	Secure
Init	All other conditions		-	-	-	-	-
Check	System reset	all other conditions	Non-fatal security violation[1] or write 1 to the SW_SV bit of HP Command Reg, or write 1 to the SSM_ST bit of the SM_HP Command Reg when non-secure boot	Fatal security violation[2] or write 1 to the SW_FSV bit of the SM_HP Command Reg	-	Write 1 to the SSM_ST bit of the SM_HP Command Reg when secure boot	-
Non-secure	System reset	-	all other conditions	Fatal security violation[2] or write 1 to the SW_FSV bit of the SM_HP Command Reg	-	-	-
Soft Fail	System reset	-	Write 1 to the SSM_ST bit of HP Command Reg (if not disabled by	all other conditions	HAC is Enabled[3] & HAC Counter[4] is Zero	-	-

Table continues on the next page...

Table 6-75. System security state machine state transition table (continued)

Current state	Next state (transition to next state determined by conditions listed in the table cells)						
	Init	Check	Non-secure	Soft fail	Hard fail	Trusted	Secure
			SSM_SFNS_DIS bit of the SM_HP Command Reg)				
Hard fail	System reset	-	-	-	all other conditions	-	-
Trusted	System reset	-	-	Non-fatal[1] or Fatal Security Violation[2] or write 1 to either the SW_FSV bit or the SW_SV bit of the SM_HP Command Reg	-	all other conditions	Write 1 to the SSM_ST bit of the SM_HP Command Reg
Secure	System reset	-	-	Non-fatal[1] or Fatal Security Violation[2] or write 1 to either the SW_FSV bit or the SW_SV bit of the SM_HP Command Reg	-	Write 1 to the SSM_ST bit of HP Command Reg (if not disabled by SSM_SFNS_DIS bit of the SM_HP Command Reg)	All other conditions

1. See HPSVCR for a list of non-fatal security violations.
2. See HPSVCR for a list of fatal security violations.
3. See the HAC_EN field of the SM_HP command register (HPCOMR).
4. See the HAC counter register (HPCOMR).

The SSM has different security violation sources. Security violation sources classified as fatal security violations always results in the SSM transition to the Soft Fail state. The non-fatal security violation results in transition from Check to Non-Secure or from Trusted/Secure to Soft Fail states. Refer to HP security violation policy for a detailed description of the Security Monitor security violation policy.

6.3.6.2 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 6-76. 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.

6.3.6.3 Master key checking and control

The Master Key Control block functionality depends on the particular security conditions and configuration. The Master Key Control block selects between the following Master Key sources:

- One Time Programmable Master Key (OTPMK)

  The OTPMK is a random value stored in the non-volatile memory (typically fuses) outside the SEC module.

- Zeroizable Master Key (ZMK)

  The ZMK is a random value stored in the ZMK Registers of the SM_LP section.

- Combined Master Key (CMK)

  This key is created by bitwise XOR operation of OTPMK and ZMK values.

The Master Key selection scheme is shown in Figure.

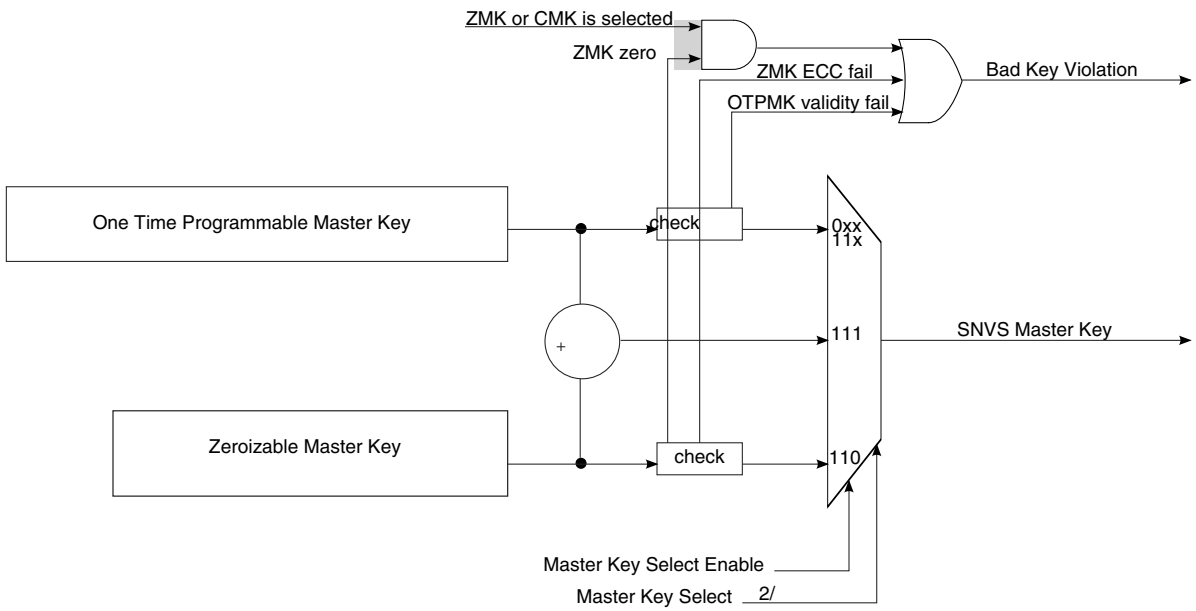


Figure 6-81. Master Key Control Scheme

The Master Key Control block incorporates hardware checking logic for the ZMK value to assure that the keys were actually programmed with the intended/valid value and have not been corrupted/degraded with time. If ZMK validity check fails, then this would indicate that the corresponding key value was either misprogrammed or was corrupted/degraded. The “bad key” violation is reported to the System Security Monitor and is treated by SSM as a non-fatal violation. The check results can always be found in the HP Status Registers.

The ZMK is checked to be different from the default all-zero value. If ZMK is found to contain the all-zero value and this key is selected for use and is marked as valid, the “bad key” violation is asserted. In addition, the ZMK can be optionally checked to generate a valid nine-bit Hamming codeword. When this feature is enabled and ZMK is found to generate invalid codeword, the “bad key” violation is asserted.

6.3.6.4 SM LP description

The SM_LP is a data storage subsystem with enhanced security capabilities.

The purpose of SM_LP is to store and protect system secure data, regardless of the main system power state. The SM_LP contains a Zeroizable Master Key and Power Glitch Detector. The SM_LP also incorporates security violations and tampers detection logic. The following sections describe in detail the SM_LP module functionality.

6.3.6.4.1 SM LP behavior during system power down

On system power down the SM_HP is powered-down and SM_LP is powered from the backup power supply.

It is isolated from the rest of the SoC by means of isolation cells. It retains the value of Zeroizable Master Key and all parameters stored in SM_LP registers. On system power down the SM_LP ignores all inputs from SM_HP. The Tamper input from the pin remains functional because it is in the same power domain and should stay stable during system power-down mode.

Input from Power Fail Detector deactivates SM_LP isolation once stable power is restored to the SoC.

6.3.6.4.2 Zeroizable master key

The SM_LP incorporates logic for storage of a 256-bit Zeroizable Master Key (ZMK) value.

The ZMK value or its modification can be provided as an output of the module for the use in SEC5.2. The ZMK value can be programmed either by SW or directly by HW without SW intervention. In the HW programming mode the ZMK value cannot be read by SW. In the SW programming mode the ZMK value can be read by SW before it is locked. The ZMK is asynchronously zeroized and invalidated in case of security violation.

6.3.7 Initialization guidelines

The following steps should be completed to properly initialize the Security Monitor module (following successful completion of secure boot):

1. Transition SSM from trusted state to secure state (if not already done by ESBC and if not using descriptor signing keys)
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: Set lock bits

5. Bit 31 (reserved) of the SM_HP lock register should be set before starting any functional 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.

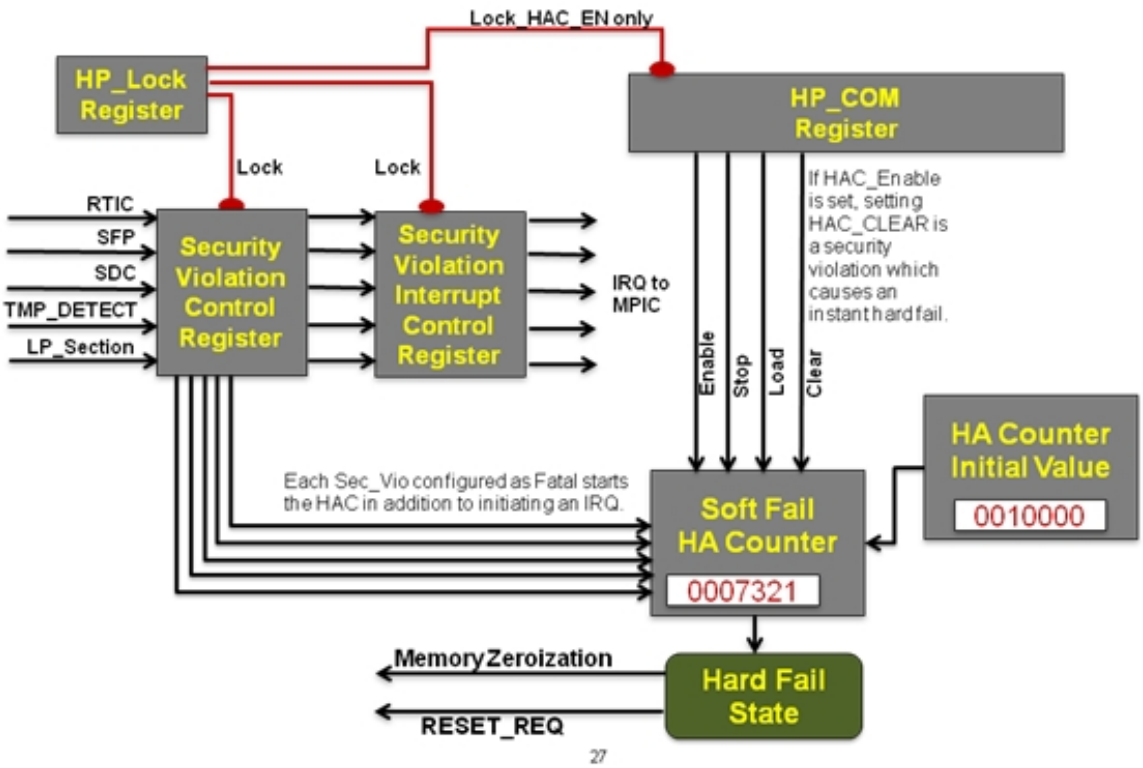


Figure 6-82. Relationship Between the Registers

6.3.8 Zeroizable master key programming guidelines

The Zeroizable Master Key (ZMK) can be programmed either by software or by hardware.

To program ZMK by software do the following:

- Verify that ZMK_HWP bit is not set
- Verify that ZMK is not locked for write
- Write key value to the ZMK registers
- Verify that the correct key value is written
- Set ZMK_VAL bit if ZMK will be read directly from hardware connection to the Cryptographic Module. There is no need to set this bit is ZMK register are only read by software

- (optional) Set ZMK_ECC_EN bit 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) Block software read accesses to the ZMK registers and ZMK_ECC_VALUE field by setting ZMK Read lock bit
- (optional) Block software write accesses to the ZMK registers by setting ZMK Write Lock bit
- Set MASTER_KEY_SEL and MKS_EN bits to select combination of OTPMK and ZMK to be provided to the HW Cryptographic Module
- (optional) Block software write accesses to the MASTER_KEY_SEL field by setting MKS lock bit

To program ZMK by hardware do the following:

- Set ZMK_HWP bit
- Verify that ZMK is not locked for write
- Write one to the PROG_ZMK bit
- Check when ZMK_VAL bit is set. This bit is set by hardware at the end of ZMK programming cycle
- (optional) Set ZMK_ECC_EN bit to enable ZMK error correction code verification
- (note) ZMK Registers and ZMK_ECC_VALUE field cannot be read by software in the hardware programming mode
- (optional) Block hardware programming option of the ZMK Registers by setting ZMK Write Lock bit
- Set MASTER_KEY_SEL and MKS_EN bits to select combination of OTPMK and ZMK to be provided to the HW Cryptographic Module
- (optional) Block software write accesses to the MASTER_KEY_SEL field by setting MKS lock bit

Chapter 7
e6500 Core Integration

7.1 Introduction

This chapter provides details about how the e6500 microprocessor cores are integrated into this chip. The core provides features that the integrated chip may not implement or may implement in a more specific way.

NOTE

The following documents are necessary for understanding how to program the core:

The *EREF: A Programmer's Reference Manual for Freescale Embedded Processors* describes the instruction, register, and interrupt models, as well as other functionality defined at the architecture level and implemented on e500 family processors.

The *e6500 Core Reference Manual* describes functionality that is specific to the e6500 and that is not defined by the architecture, such as core-specific instructions, registers, and register fields. It also provides e6500-specific details about how the e6500 implements functionality that is defined by the architecture.

The *AltiVec Technology Programming Interface Manual for Power ISA Processors* describes how programmers can access AltiVec functionality from programming languages such as C and C++. The AltiVec Power ISA PIM defines a programming model for use with the AltiVec instruction set.

The *AltiVec Technology Programming Environments Manual for Power ISA Processors* is a reference guide for assembler programmers. The AltiVec Power ISA PEM uses a standardized format to describe each instruction, showing

syntax, instruction format, register translation language (RTL) code that describes how the instruction works, and a listing of which, if any, registers are affected.

7.2 Overview

The e6500 core is a low-power implementation of the resources for embedded processors defined by the Power ISA. Each core supports the simultaneous execution of two threads. The core is a 64-bit implementation and implements two sets of 64-bit general-purpose registers; however it supports accesses to 40-bit physical addresses.

The e6500 is designed to be implemented in multicore integrated devices, and many of the features are defined to support multicore implementations, in particular to partition the cores in such a way that multiple operating systems can be run with the integrated device, as shown n the following figure.

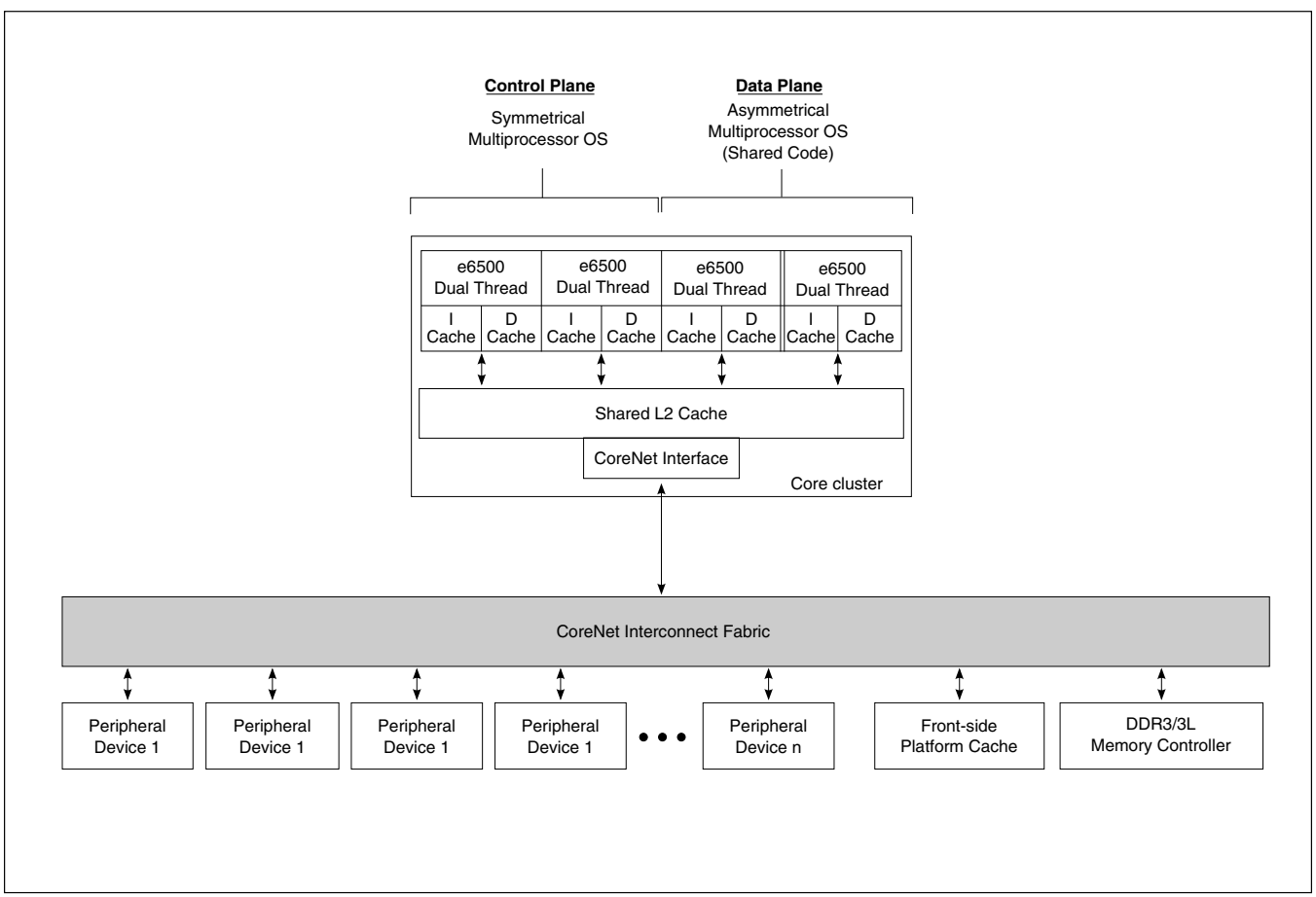


Figure 7-1. Example Partitioning Scenario of a Multicore Integrated Device

The CoreNet interface provides the primary on-chip interface between the core cluster and the rest of the SoC. CoreNet is a tag-based interface fabric that provides interconnections among the cores, peripheral devices, and system memory in a multicore implementation.

The architecture defines the resources required to allow orderly and secure interactions between thread procerssors, the cores, memory, peripheral devices, and virtual machines. These include a hypervisor and guest supervisor privilege levels, that determine whether certain activities, such as memory accesses and management, cache management, and interrupt handling, are to be carried on at a system-wide level (hypervisor level) or by the operating system within a partition (guest supervisor level).

The architecture defines the resources required to allow orderly and secure interactions between the cores, memory, peripheral devices, and virtual machines. These include a hypervisor and guest supervisor privilege levels, that determine whether certain activities, such as memory accesses and management, cache management, and interrupt handling, are to be carried on at a system-wide level (hypervisor level) or by the operating system within a partition (guest supervisor level).

Each core is a multi-threaded, superscalar processor that can decode two instructions and complete two instructions per thread per clock cycle. Instructions complete in order, but can execute out of order. Execution results are available to subsequent instructions in the same thread through the rename buffers, but those results are recorded into architected registers in program order, maintaining a precise exception model.

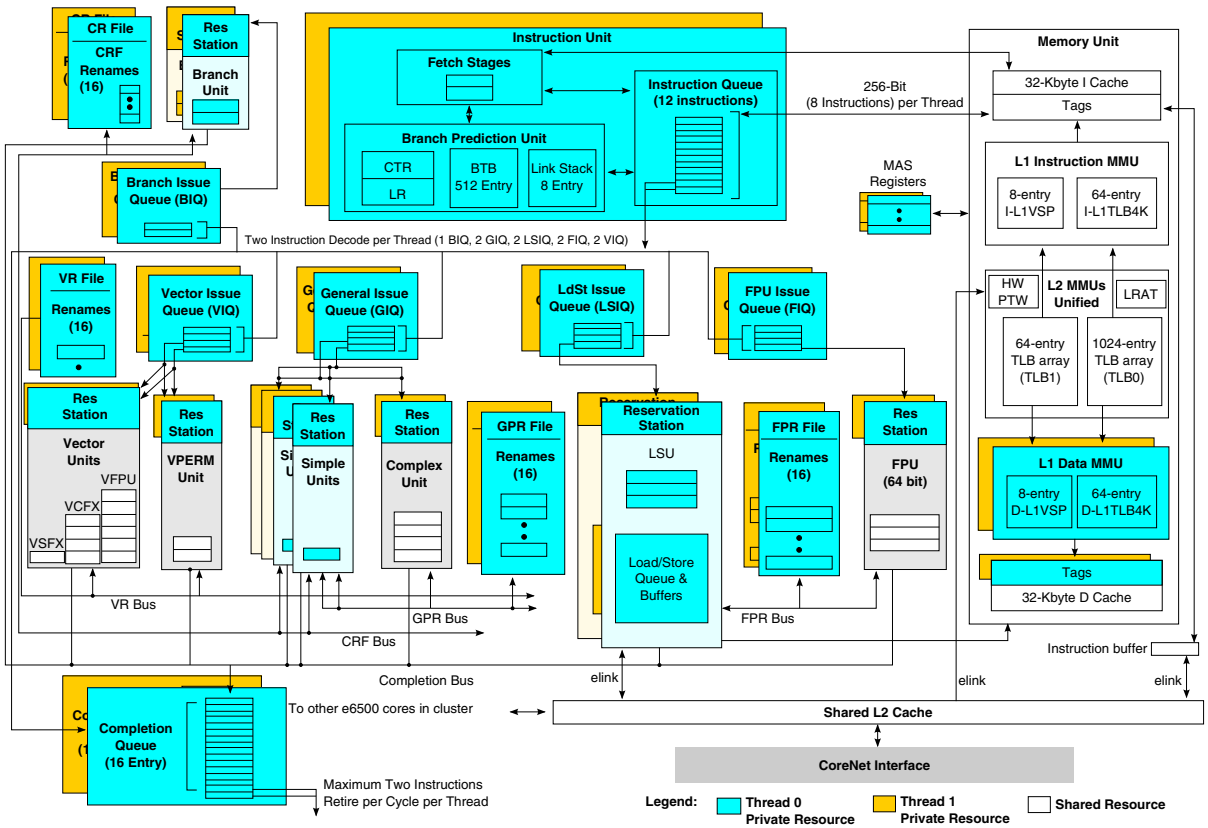


Figure 7-2. Core Block Diagram

7.3 Chip-Specific Core Implementation Details

This section describes any aspects of the implementation of the cores that are specific to the chip. Information in this section supersedes any more general descriptions provided in either *EREF: A Programmer's Reference Manual for Freescale Embedded Processors* or the *e6500 Core Reference Manual.*

7.3.1 Benefits and Flexibility of Four Dual-Threaded Cores

Each core supports two hardware threads, which software views as a virtual CPU (VCPU). These VCPUs can be combined as a fully symmetric multiprocessing system on a chip, or they can be operated with variable degrees of independence to perform asymmetric multiprocessing. Full processor independence, including the ability to independently boot and reset each core, is a defining characteristic of the chip. The ability of the cores to run different operating systems, or run without an OS, provides significant

flexibility in partitioning among control, data path, and applications processing. It also simplifies consolidation of functions previously spread across multiple discrete processors onto a single device.

The cores are arranged in a cluster of four with a shared 2-MByte L2 cache and interface to the CoreNet interconnect fabric. The cluster reduces the amount of coherency traffic on the CoreNet interface and provides faster coherent transactions among cores in a cluster.

7.3.2 CoreNet Coherency Fabric (CCF)

The CoreNet coherency fabric (CCF) provides interconnections among the core cluster, peripheral devices, and system memory in a multicore implementation. CoreNet Coherency Fabric (CCF) describes the CCF in detail, and in particular it explains how it facilitates communication between the cores, the core platform caches (CPC), and the other blocks that comprise the coherent memory domain.

7.3.3 Reset and Clocking

Reset, Clocking, and Initialization provides information about coordination between the core and the device as part of the boot process.

The following signals are important to consider regarding the integration of the cores:

- The system clock (SYSCLK), is the primary clock input and is the clock source for the core cluster group PLL. The core cluster group PLL output is assigned to the core cluster using the C1_PLL_SEL field in the RCW.
- The real time clock, RTC, may be used (optionally) to clock the time base of the cores. The RTC timing specifications are given in the hardware specifications for this chip. This signal can also be used (optionally) to clock the global timers in the programmable interrupt controller (PIC).

Core timer facilities are described in the *e6500 Core Reference Manual*.

7.3.4 Register Model Implementation Details

The device implements the core register model as defined for the core and described in the *EREF* and the *e6500 Core Reference Manual*. Details that are specific to how the core registers are implemented in the device are described in the following subsections.

7.3.4.1 Processor Version Register (PVR)

The table below lists the revision codes in the processor version register (PVR). The PVR value is source by the core processor itself. Its value is reflected in the PVR register, which is described in Processor Version Register (DCFG_CCSR_PVR). The value is also visible to software running on the core in a read-only SPR within the core, SPR287.

Table 7-1. Processor Version Register

ID	Bits	Assigned Value(Binary)	Description
PVR	[0:3]	1000	Manufacturer ID
	[4:5]	00	Reserved
	[6:9]	0001	Processor type
	[10:15]	00_0000	Processor ID
	[16:19]	0000	Process revision
	[20:23]	0000	Manufacturing revision
	[24:27]	0010	Processor major revision number
	[28:31]	0000	Processor minor revision number

7.3.4.2 System Version Register (SVR)

The SVR register, which is described in System Version Register (DCFG_CCSR_SVR), is routed to input terminals on the core as well where it is accessible as a read-only register within the core, SPR1023. This table contains the assigned values.

Table 7-2. System Version Register

ID	Bits	Assigned Value (Binary)	Description	Comments
SVR	[0:3]	1000	Manufacturer ID	Freescale
	[4:11]	0101_0011	SoC device ID	T2080
	[12]	n	Security	0 No security 1 Security enabled
	[13:15]	*nnn*	SoC variant ID	000 T2080 001 T2081
	[16:23]	0000_0000	Personality	
	[24:27]	*nnnn*	Device major revision number	For first silicon, this is set to 0001 for major revision 1.
	[28:31]	*mmmm*	Device minor revision number	For first silicon, this is set to 0000 for minor revision 0.

7.3.4.3 (Guest) Processor ID Register (PIR/GPIR)

The value in PIR/GPIR is used to distinguish processors in a system from one another. The processor control category augments this definition to make the PIR writable so software can store information specific to its needs.

In a multiprocessor system, each PIR register should be initialized at power-on-reset to a unique value. The contents of the PIR register are used as a tag for matching requested doorbell interrupts sent to the processor. Note that SPR 286 was assigned to the system version register (SVR) on some earlier devices.

Out of core reset, the PIR values for each thread are set to the following values.

NOTE

The PIR registers can be later modified if desired by Hypervisor software.

Table 7-3. PIR Values out of Reset

Physical Core	Thread	Initial PIR Value
0	0	0000_0000h
	1	0000_0001h
1	0	0000_0008h
	1	0000_0009h
2	0	0000_0010h
	1	0000_0011h
3	0	0000_0018h
	1	0000_0019h
10	0	0000_0050h

7.3.4.4 Timer Control Register (TCR)

TCR[WRC] is defined more specifically for the implementation of the core in the integrated device, as shown in the table below.

Table 7-4. TCR[WRC] Descriptions

Bits	Name	Description
34-35	WRC	See Watchdog Timer Expiration Out to Platform.

7.3.5 Cache Model Implementation Details

The device's implemented caches can be configured and locked using instructions defined by the Power architecture and described in the *EREF* and in the *e6500 Core Reference Manual*. These instructions specify the L2 cache by setting the CT operand to 2 and the CoreNet platform cache (CPC) by setting the CT operand to 1.

Caches in a system may be specific targets of *cache stashing*, an operation initiated by the processor or other device in the system, specifying a hint that specified addresses should be prefetched into a target cache. Each cache in the system that responds to such requests has a cache identifier set by system software or predefined by hardware. For the L1 data cache, the identifier is defined in L1CSR2[DCSTASHID]. For the L2 cache, the identifier is defined in L2CSR1[L2STASHID]. A cache identifier value of 0 indicates that cache stashing operations are not accepted or performed by the cache.

NOTE

The L2 cache must have ECC enabled as follows:

- L2CSR0[L2PE] = 1
- L2ERRDIS[MBECCDIS] = 0 and L2ERRDIS[SBECCDIS] = 0

7.3.6 Interrupt Model Implementation Details

The core implements the external proxy functionality, which can be used to eliminate the need for an interrupt handler to perform a software read IACK. See the *Multicore Programmable Interrupt Controller (MPIC)* chapter for information about how this functionality is implemented on the device.

The architecture defines resources to eliminate the need from an interrupt handler responding to the assertion of the external interrupt signal, *int*, from a programmable interrupt controller (PIC) in the integrated device. The Open PIC standards define a software handshake in response to the assertion of this signal that requires the handler to read the memory-mapped IACK register, which holds interrupt vector offset bits provided by the source of the interrupt, typically one of the peripheral devices incorporated in the integrated device. In addition to providing the interrupt vector, this read also informs internal logic of the PIC to treat this interrupt source as "in-service" rather than pending and it signals the PIC to deassert its internal *int* output, making it possible for the PIC to deliver an external interrupt from another peripheral device that had been configured as higher priority

The architecture defines the external proxy register, EPR, used when the interrupt is handled at the hypervisor level, and a guest EPR, GEPR, when the interrupt is handled ar the guest supervisor level. The EPR and GEPR are SPRs, and the value that appears in the PIC's IACK[VECTOR] fields is automatically loaded into the EPR (or GEPR), eliminating the cache-inhibited guarded load to the memory mapped IACK. The EPR is considered valid only from the time that the external input interrupt occurs until MSR[EE] is set as the result of a **mtmsr** or a return from interrupt instruction.

The interrupt proxy functionality can be enabled and disabled through the global configuration register in the PIC. See the PIC chapter in the reference manual for the integrated device for more information.

7.3.6.1 Watchdog Timer Expiration Out to Platform

Each core has an internal watchdog timer that can be configured to signal the platform upon "first" or "second" expiration; the see e6500 documentation for details. Such an indication out to the platform typically identifies runaway or faulty software or that something has gone wrong interacting with the platform.

There are four different actions taken in response to such a watchdog timer expiration depending on the value programmed in TCR[WRC] and EDBCR0[EDM] of the core:

- Internal interrupt sent over to the MPIC for WRC/WRS==01 while EDM core output == 0
- Internal interrupt sent over to the MPIC and automatic reset of that core for WRC/WRS==11 while EDM core output == 0
- Internal interrupt sent over to the MPIC and device output RESET_REQ_B is asserted to external world: WRC/WRS==10 while EDM core output == 0
- SoC specific debug action taken when WRC/WRS==01, 10, 11 while EDM core output == 1.

Note that the MPIC allows for watchdog timer expiration events from the cores to be delivered as either a regular, critical, or machine check type interrupt to a designated core. The watchdog timer expiration events from all cores are conceptually ORed to present one interrupt source to the MPIC. A status register in the MPIC records which of the cores has an expired watchdog timer status. See the *Multicore Programmable Interrupt Controller (MPIC)* chapter.

7.3.7 Performance Monitor Details

The performance monitor facility provides the ability to monitor and count predefined events such as processor clocks, misses in the instruction cache or data cache, types of instructions decoded, or mispredicted branches. The count of such events can be used to trigger the performance monitor interrupt or a Nexus event. Additional performance monitor registers (PMRs) similar to SPRs are used to configure and track performance monitor operations. These registers are accessed with the Move to PMR and Move from PMR instructions (mtpmr and mfpmr).

Note that the core's performance monitor is separate from the SoC-level performance monitor which is similar in its design, but uses memory-mapped performance monitor registers.

The performance monitor can be configured to trigger either a performance monitor interrupt or an event to the Nexus facility when configured conditions are met.

7.3.8 Decorated Load and Store Operations

The architecture defines a set of decorated load and store instructions that allow efficient, SoC-specific operations, such as atomically sampling and updating packet-counting statistics. The SoC defines specific semantics understood by a SoC-customized resource that requires them. See Decorated Storage for more information.

The architecture defines the a set of load/store byte, half-word, and word decorated integer operations and load/store floating-point double-word instructions, which provide the EA in **r**B and the decoration in **r**A.

The Decorated Storage Notify (dsn) instruction is an address-only operation that sends a decoration without sending data.

7.3.9 Debug Features

The core provides a comprehensive, extensively integrated set of internal and external debug features to support multicore implementations:

- Debug interrupt is a separate interrupt type
- Nexus facility

- Performance monitor now provides the ability to trigger either a performance monitor interrupt or a Nexus event
- Debug resources are extensively integrated at the SoC level. Implementation-specific details about core debug resources are described in the *EREF* and the *e6500 Core Reference Manual*.

7.3.10 Power Management

The core defines registers and instructions that provide power management that is programmable from the core. See the register and instruction model chapters of the *e6500 Core Reference Manual.* Integration details are provided in the *Run Control/Power Management Unit* chapter, which describes the run control and power management (RCPM) block, which allows the device to operate at various frequencies and execution modes to optimize power consumption.

7.3.11 Dealing with Guest OS WIMGE Aliasing in a Boundedly Undefined Manner

The e6500 core uses TLB entries to map pages of real memory. Each TLB entry contains a set of attributes which describe how the core and the system should treat core accesses to that page. These are refered to as WIMGE attributes for Write-Through Required, Caching Inhibited, Guarded, and Endianess. A complete description of these attributes can be found in EREF.

In general it is required that system software regulate how accesses are performed to memory by the cores, and are generally required to ensure that each core accesses a page with the same memory attributes.

Mixing these attributes can potentially cause incoherent accesses to the memory, or can cause memory ordering failures. In order to prevent such problems, system software should ensure that caches are flushed between any access for which the W, I, or M attribute is different. For any such mismatched accesses in which the caches are not flushed, the result is undefined (except that accesses where only the W attribute differs, from the same thread, are supported and behave as described in EREF).

In a virtualized system, where a hypervisor is running one or more guest OSes, use of the LRAT may permit a guest OS to program the TLB such that WIMGE mismatched access can occur resulting in undefined behavior. In general, this would require a misbehaving OS, or a nefarious guest OS, but the end result of the undefined behavior can result in a system deadlock, thus causing a severe denial of service to other guests (partitions) in the system.

If boundedly undefined behavior is desired for mismatched WIMGE attributes (such that failures which result from such accesses are confined to the partition which initiates the accesses), hypervisor software should do one of the following:

1. Do not use the LRAT, causing all TLB write (and page table translations) to be processed by the hypervisor and the hypervisor assures that the WIMGE bits are set correctly when writing the translation.
2. Or turn off CPC sole-data. [There needs to be a reference here to the register description in the CPC which is used to do this. The description of that bit does not yet exist.] In addition, the hypervisor should guarantee that if L2 caches are removed from the coherence fabric (through configuration or by turning clusters off), that the caches are first flushed of all modified data and invalidated prior to re-enabling them.

Chapter 8
CoreNet Platform Cache (CPC)

8.1 Introduction

8.1.1 CPC Overview

The CoreNet platform cache (CPC) is a CoreNet-compliant target device that functions as a general purpose write-back cache, I/O stash and memory mapped SRAM device, or a combination of these functions. As a general purpose cache, it manages allocations and victimizations to improve read latency and bandwidth over accesses to backing store (for example, DRAM). As an I/O stash, it can accept and allocate writes from an I/O device in order to reduce latency and improve bandwidth for future read operations to the same address. As an SRAM device, it acts as a low-latency, high-bandwidth memory that occupies a programmable address range.

The CPC connects between the and the memory controller in an inline configuration and cannot function as a lookaside cache. Therefore, the CoreNet platform cache can only cache address ranges that are present in the memory controller behind it. This limitation does not apply to SRAM, which does not have backing store.

8.1.2 Features

The CPC includes the following features:

- In-line configuration
- Support for interleaving to match memory controllers (none at defined address boundaries)
- 512-Kbyte, 16-way set associative, 64-byte coherency granule
- Configurable pseudo-least recently used (PLRU), streaming PLRU with aging, streaming PLRU without aging, and first-in/first-out (FIFO) replacement policies with programmable allocation policy and update options

- Support for individual line locking
- Write-back or write-through operation
- Low latency interface to memory controller including a dedicated 128-bit data bus in each direction
- Partial SRAM and partial I/O stash mode with programmable sizes
- Stashing support
  - Stashing of all transactions and sizes supported (not limited to 64-byte or 32-byte)
  - Explicit (CoreNet signalled) and implicit (address range based) stash allocation
- Support for decorated storage

8.2 CoreNet Platform Cache (CPC) Memory Map

The following table shows the memory map for the CoreNet platform cache (CPC) registers. The CPC registers are accessed by reading and writing to an address comprised of the base address (specified by the CCSRBAR), plus the CPC block base address, plus the offset of the specific register to be accessed.

CPC memory map

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1_0000	CPC configuration and status register 0 (CPC_CPCCSR0)	32	R/W	0000_0000h	8.2.1/313
1_0008	CPC configuration register 0 (CPC_CPCCFG0)	32	R	50B3_C008h	8.2.2/316
1_0010	CPC external write control register n (CPC_CPCEWCR0)	32	R/W	0000_0000h	8.2.3/318
1_0014	CPC external write base address register n (CPC_CPCEWBAR0)	32	R/W	0000_0000h	8.2.4/319
1_0020	CPC external write control register n (CPC_CPCEWCR1)	32	R/W	0000_0000h	8.2.3/318
1_0024	CPC external write base address register n (CPC_CPCEWBAR1)	32	R/W	0000_0000h	8.2.4/319
1_0100	CPC SRAM control register 1 (CPC_CPCSRCR1)	32	R/W	0000_0000h	8.2.5/320
1_0104	CPC SRAM control register 0 (CPC_CPCSRCR0)	32	R/W	0000_0000h	8.2.6/321
1_0200	CPC partition ID register 0 (CPC_CPCPIR0)	32	R/W	FFFF_FFFFh	8.2.7/322
1_0208	CPC partition allocation register 0 (CPC_CPCPAR0)	32	R/W	FFFF_FBFFh	8.2.8/323
1_020C	CPC partition way register 0 (CPC_CPCPWR0)	32	R/W	FFFF_FFFFh	8.2.9/324
1_0210	CPC partition ID register n (CPC_CPCPIR1)	32	R/W	0000_0000h	8.2.10/325
1_0218	CPC partition allocation register n (CPC_CPCPAR1)	32	R/W	0000_0000h	8.2.11/325
1_021C	CPC partition way register 0 (CPC_CPCPWR1)	32	R/W	0000_0000h	8.2.12/327
1_0220	CPC partition ID register n (CPC_CPCPIR2)	32	R/W	0000_0000h	8.2.10/325

Table continues on the next page...

CPC memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1_0228	CPC partition allocation register n (CPC_CPCPAR2)	32	R/W	0000_0000h	8.2.11/325
1_022C	CPC partition way register 0 (CPC_CPCPWR2)	32	R/W	0000_0000h	8.2.12/327
1_0230	CPC partition ID register n (CPC_CPCPIR3)	32	R/W	0000_0000h	8.2.10/325
1_0238	CPC partition allocation register n (CPC_CPCPAR3)	32	R/W	0000_0000h	8.2.11/325
1_023C	CPC partition way register 0 (CPC_CPCPWR3)	32	R/W	0000_0000h	8.2.12/327
1_0240	CPC partition ID register n (CPC_CPCPIR4)	32	R/W	0000_0000h	8.2.10/325
1_0248	CPC partition allocation register n (CPC_CPCPAR4)	32	R/W	0000_0000h	8.2.11/325
1_024C	CPC partition way register 0 (CPC_CPCPWR4)	32	R/W	0000_0000h	8.2.12/327
1_0250	CPC partition ID register n (CPC_CPCPIR5)	32	R/W	0000_0000h	8.2.10/325
1_0258	CPC partition allocation register n (CPC_CPCPAR5)	32	R/W	0000_0000h	8.2.11/325
1_025C	CPC partition way register 0 (CPC_CPCPWR5)	32	R/W	0000_0000h	8.2.12/327
1_0260	CPC partition ID register n (CPC_CPCPIR6)	32	R/W	0000_0000h	8.2.10/325
1_0268	CPC partition allocation register n (CPC_CPCPAR6)	32	R/W	0000_0000h	8.2.11/325
1_026C	CPC partition way register 0 (CPC_CPCPWR6)	32	R/W	0000_0000h	8.2.12/327
1_0270	CPC partition ID register n (CPC_CPCPIR7)	32	R/W	0000_0000h	8.2.10/325
1_0278	CPC partition allocation register n (CPC_CPCPAR7)	32	R/W	0000_0000h	8.2.11/325
1_027C	CPC partition way register 0 (CPC_CPCPWR7)	32	R/W	0000_0000h	8.2.12/327
1_0280	CPC partition ID register n (CPC_CPCPIR8)	32	R/W	0000_0000h	8.2.10/325
1_0288	CPC partition allocation register n (CPC_CPCPAR8)	32	R/W	0000_0000h	8.2.11/325
1_028C	CPC partition way register 0 (CPC_CPCPWR8)	32	R/W	0000_0000h	8.2.12/327
1_0290	CPC partition ID register n (CPC_CPCPIR9)	32	R/W	0000_0000h	8.2.10/325
1_0298	CPC partition allocation register n (CPC_CPCPAR9)	32	R/W	0000_0000h	8.2.11/325
1_029C	CPC partition way register 0 (CPC_CPCPWR9)	32	R/W	0000_0000h	8.2.12/327
1_02A0	CPC partition ID register n (CPC_CPCPIR10)	32	R/W	0000_0000h	8.2.10/325
1_02A8	CPC partition allocation register n (CPC_CPCPAR10)	32	R/W	0000_0000h	8.2.11/325
1_02AC	CPC partition way register 0 (CPC_CPCPWR10)	32	R/W	0000_0000h	8.2.12/327
1_02B0	CPC partition ID register n (CPC_CPCPIR11)	32	R/W	0000_0000h	8.2.10/325
1_02B8	CPC partition allocation register n (CPC_CPCPAR11)	32	R/W	0000_0000h	8.2.11/325
1_02BC	CPC partition way register 0 (CPC_CPCPWR11)	32	R/W	0000_0000h	8.2.12/327
1_02C0	CPC partition ID register n (CPC_CPCPIR12)	32	R/W	0000_0000h	8.2.10/325
1_02C8	CPC partition allocation register n (CPC_CPCPAR12)	32	R/W	0000_0000h	8.2.11/325
1_02CC	CPC partition way register 0 (CPC_CPCPWR12)	32	R/W	0000_0000h	8.2.12/327
1_02D0	CPC partition ID register n (CPC_CPCPIR13)	32	R/W	0000_0000h	8.2.10/325
1_02D8	CPC partition allocation register n (CPC_CPCPAR13)	32	R/W	0000_0000h	8.2.11/325
1_02DC	CPC partition way register 0 (CPC_CPCPWR13)	32	R/W	0000_0000h	8.2.12/327
1_02E0	CPC partition ID register n (CPC_CPCPIR14)	32	R/W	0000_0000h	8.2.10/325
1_02E8	CPC partition allocation register n (CPC_CPCPAR14)	32	R/W	0000_0000h	8.2.11/325
1_02EC	CPC partition way register 0 (CPC_CPCPWR14)	32	R/W	0000_0000h	8.2.12/327

Table continues on the next page...

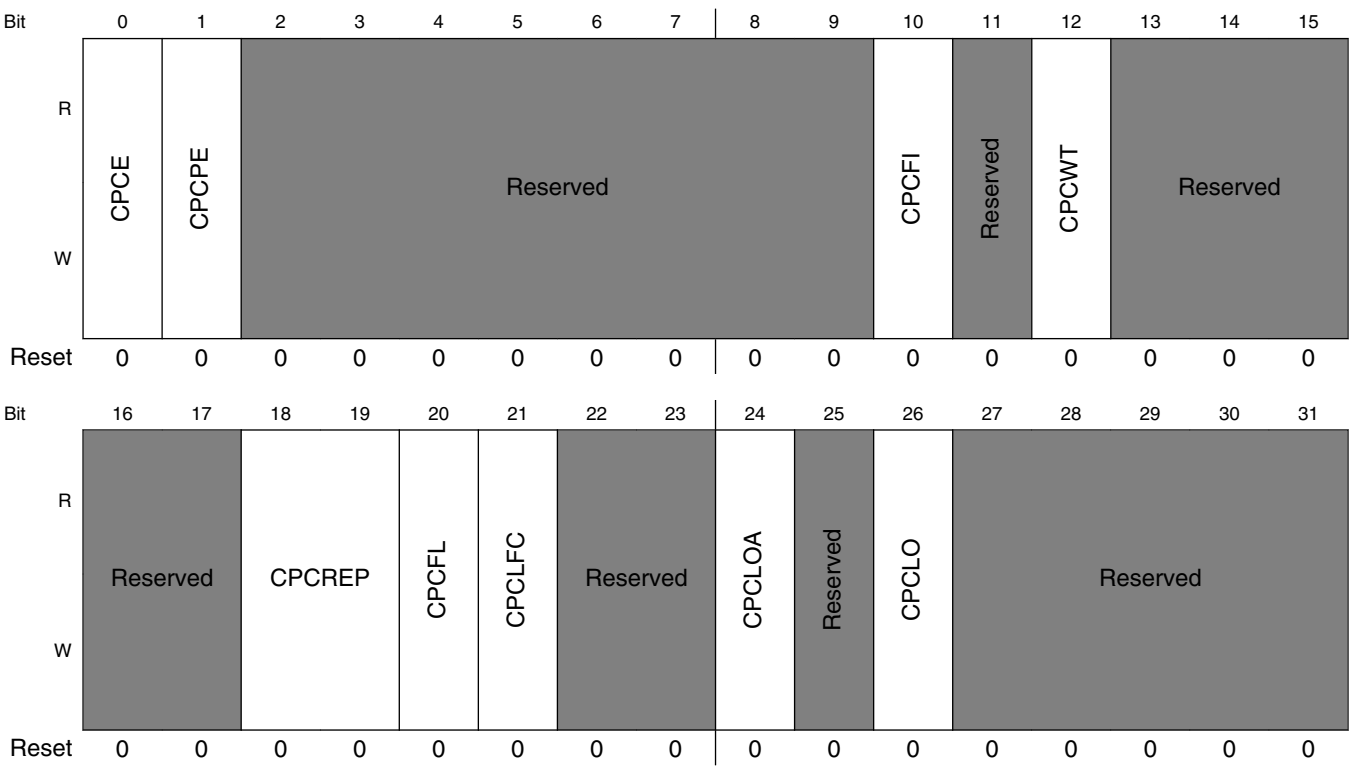
CPC memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1_02F0	CPC partition ID register n (CPC_CPCPIR15)	32	R/W	0000_0000h	8.2.10/325
1_02F8	CPC partition allocation register n (CPC_CPCPAR15)	32	R/W	0000_0000h	8.2.11/325
1_02FC	CPC partition way register 0 (CPC_CPCPWR15)	32	R/W	0000_0000h	8.2.12/327
1_0E00	CPC error injection high register (CPC_CPCERRINJHI)	32	R/W	0000_0000h	8.2.13/327
1_0E04	CPC error injection low register (CPC_CPCERRINJLO)	32	R/W	0000_0000h	8.2.14/327
1_0E08	CPC error injection control register (CPC_CPCERRINJCTL)	32	R/W	0000_0000h	8.2.15/328
1_0E20	CPC capture data high register (CPC_CPCCAPTDATAHI)	32	R	0000_0000h	8.2.16/329
1_0E24	CPC capture data low register (CPC_CPCCAPTDATALO)	32	R	0000_0000h	8.2.17/329
1_0E28	CPC capture ECC register (CPC_CPCCAPTECC)	32	R	0000_0000h	8.2.18/329
1_0E40	CPC error detect register (CPC_CPCERRDET)	32	w1c	0000_0000h	8.2.19/331
1_0E44	CPC error disable register (CPC_CPCERRDIS)	32	R/W	0000_0000h	8.2.20/333
1_0E48	CPC error interrupt enable register (CPC_CPCERRINTEN)	32	R/W	0000_0000h	8.2.21/334
1_0E50	CPC error extended address register (CPC_CPCERREADDR)	32	R/W	0000_0000h	8.2.22/335
1_0E54	CPC error address register (CPC_CPCERRADDR)	32	R/W	0000_0000h	8.2.23/335
1_0E58	CPC error control register (CPC_CPCERRCTL)	32	R/W	0000_0000h	8.2.24/336
1_0F00	CPC hardware debug control register 0 (CPC_CPCHDBCR0)	32	R/W	001E_0000h	8.2.25/336

8.2.1 CPC configuration and status register 0 (CPC_CPCCSR0)

The CPC configuration and status register (CPCCSR0) controls configuration and operation of the CPC array. The sequence for modifying CPCCSR0 can be found at Modifying CPC Control and Status Registers.

Address: 1_0000h base + 0h offset = 1_0000h



CPC_CPCCSR0 field descriptions

Field	Description
0 CPCE	CPC enable. Used to enable the CPC array (cache or memory-mapped SRAM). 0 CPC is disabled. All transactions will behave as if they miss in the cache. 1 CPC is enabled
1 CPCPE	CPC parity/ECC error checking enable. Used to enable error checking in the CPC arrays. This bit must be set for normal operation. **NOTE:** The value of CPCPE should not be changed after the CPC is enabled. CPCPE bit should only be set or cleared simultaneously with setting or clearing CPCE. 0 Error checking disabled 1 Error checking enabled
2–9 -	This field is reserved. Reserved

Table continues on the next page...

CPC_CPCCSR0 field descriptions (continued)

Field	Description
10 CPCFI	CPC flash invalidate. Invalidation occurs regardless whether the cache is enabled or not. . **NOTE:** Attempting to set CPCFI during an invalidation operation causes undefined results; attempting to clear CPCFI during an invalidation operation is ignored. **NOTE:** Note: If CPCFI and CPCLFC are set simultaneously with the same register write operation, then the CPC flash invalidate and CPC lock flash clear functions are performed in parallel. In this case, hardware clears both the valid and lock bits, as well as all the tag and LRU bits, with a single pass through all indices, writing each entry in the CPC tag, status and victim arrays. In addition, all data arrays are cleared by writing 0s to all locations. **NOTE:** Before enabling the CPC through CPCE, the CPC tags should be cleared by setting both CPCFI and CPCLFC. This insures that all entries in the cache are invalidated and unlocked. 0 Normal operation. No cache invalidation. 1 Cache flash invalidate. A cache invalidation operation is initiated by hardware. Once complete, this bit is cleared. All the lines in the CPC that are designated as cache (that is, not SRAM) will be invalidated.
11 -	This field is reserved. Reserved
12 CPCWT	CPC write-through mode. **NOTE:** This bit must not be changed once the CPC has been enabled through the CPCE bit. The value of CPCWT must be set at the same time as CPCE. 0 CPC is operating in write-back mode. 1 CPC is operating in write-through mode.
13–17 -	This field is reserved. Reserved
18–19 CPCREP	CPC line replacement algorithm. 00 SPLRU (Streaming Pseudo Least Recently Used) with Aging. 01 FIFO (First In, First Out). 10 SPLRU (Streaming Pseudo Least Recently Used). 11 PLRU (Pseudo Least Recently Used).
20 CPCFL	CPC flush. Setting this bit initiates a CPC flush operation causing all dirty lines to be written out to backing store (memory). **NOTE:** Writing a 1 while a flush operation is in progress is ignored. Writing a 0 during a flash clear operation is ignored. Writing a 1 to CPCFL is ignored if the CPC is not enabled (CPCE = 0). **NOTE:** To flush the CPC cache and ensure that no valid entries exist after the flush (that is, for the purposes of powering down or disabling) the following high level sequence of operations should be used: 1. Clear all bits in CPCPAR0-CPCPAR15 to prevent new transactions from allocating in the CPC 2. Set CPCCSR0[CPCFL] 3. Wait for CPCCSR0[CPCFL] to be cleared by hardware 4. Clear CPCCSR0[CPCE] 0 A CPC flush is not being performed. 1 Hardware initiates a CPC flush operation. This bit is cleared when the operation is complete. All lines in the CPC tag array that are dirty will be written back to backing store.
21 CPCLFC	CPC lock flash clear. Clearing of lock bits occurs regardless of the cache enable (CPCE) value. **NOTE:** Writing a 1 during a lock clear operation causes undefined results. Writing a 0 during a lock clear operation is ignored.

Table continues on the next page...

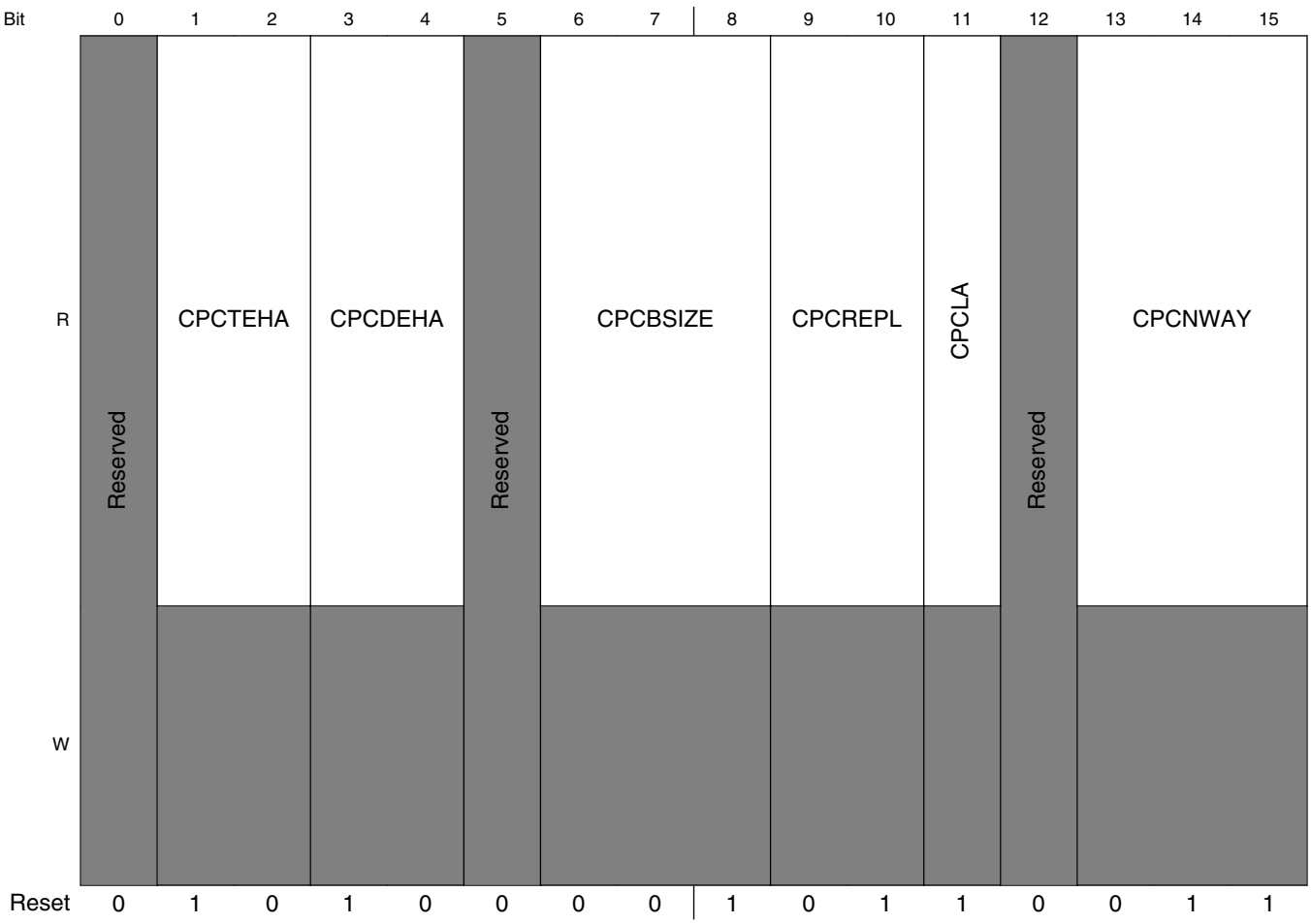
CPC_CPCCSR0 field descriptions (continued)

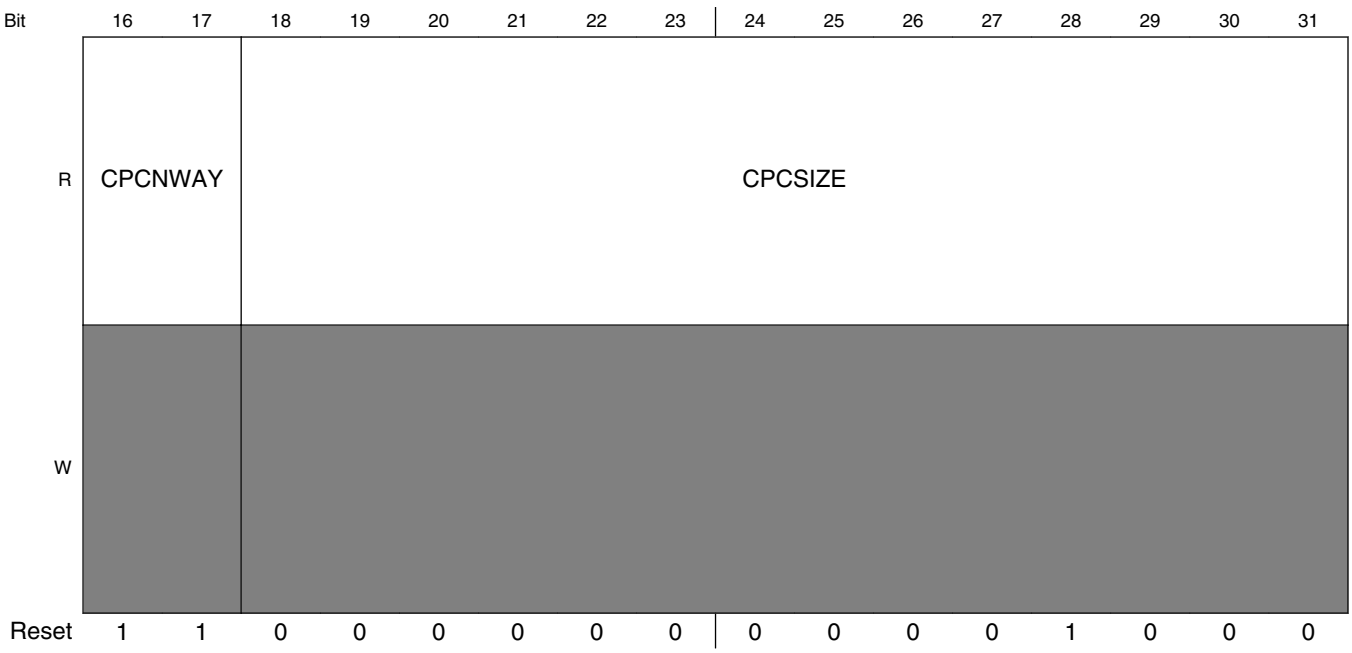
Field	Description
	NOTE: If CPCFI and CPCLFC are set simultaneously with the same register write operation, then the CPC flash invalidate and CPC lock flash clear functions will be performed in parallel. In this case, hardware will clear both the valid and lock bits, as well as all the tag and LRU bits, with a single pass through all indices, writing each entry in the CPC tag, status and victim arrays. In addition, all data arrays are cleared by writing 0's to all locations. **NOTE:** Before enabling the CPC through CPCCSR0[CPCE], the CPC tags should be cleared by setting both CPCCSR0[CPCFI] and CPCCSR0[CPCLFC]. This insures that all entries in the cache are invalidated and unlocked 0 No flash lock clear. . 1 Cache flash lock clear operation. A cache flash lock clear operation is initiated by hardware. Once complete, this bit is cleared. All the lines in the CPC that are designated as cache (that is, not SRAM) will clear their lock bits.
22–23 -	This field is reserved. Reserved
24 CPCLOA	CPC lock overflow allocate. Controls the behavior of a transaction that wishes to establish a coherency granule in the locked state. 0 Indicates a lock overflow condition will not replace an existing locked granule with the requested granule. 1 Indicates a lock overflow condition will replace an existing locked granule with the requested granule using the selected replacement algorithm.
25 -	This field is reserved. Reserved
26 CPCLO	CPC lock overflow. This sticky bit is set by hardware if a cache lock overflow situation is detected. Overflow conditions occur as the result of a transaction that wishes to establish a coherency granule in the cache in a locked state, but all available ways are already locked and as a result an existing locked coherency granule is replaced. 0 The cache has not encountered a lock overflow condition. 1 The cache has encountered a lock overflow condition.
27–31 -	This field is reserved. Reserved

8.2.2 CPC configuration register 0 (CPC_CPCCFG0)

The CPC configuration register 0 (CPCCFG0) controls the initial configuration of the CPC.

Address: 1_0000h base + 8h offset = 1_0008h





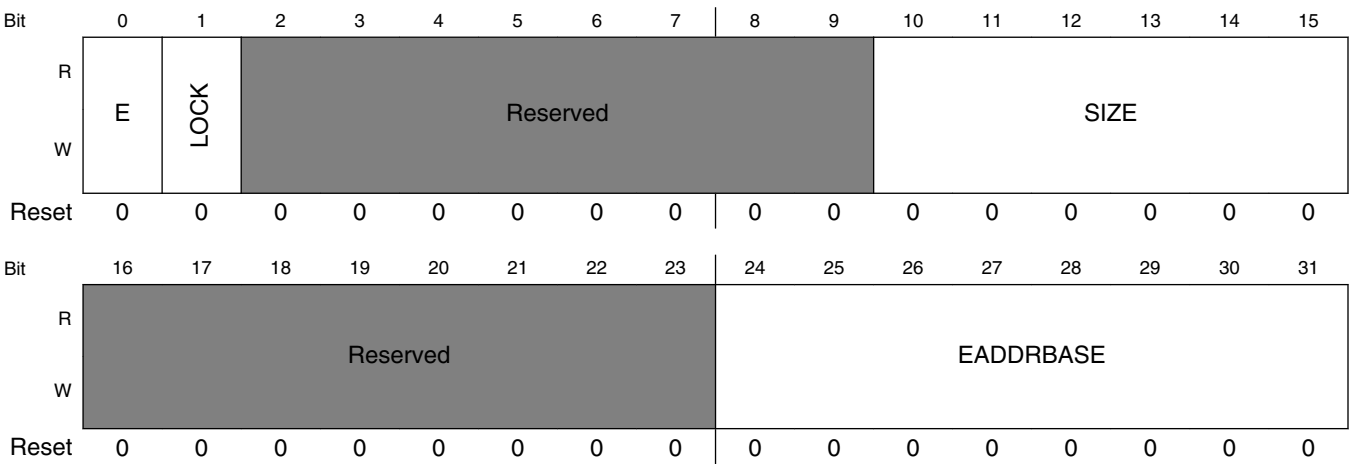
CPC_CPCCFG0 field descriptions

Field	Description
0 -	This field is reserved. Reserved
1–2 CPCTEHA	CPC tag array error handling available. 10 Single-bit ECC correction, double-bit ECC detection available
3–4 CPCDEHA	CPC data array error handling available. 10 Single-bit ECC correction, double-bit ECC detection available
5 -	This field is reserved. Reserved
6–8 CPCBSIZE	Coherency granule size. 001 64 bytes
9–10 CPCREPL	Cache default replacement policy. 01 Pseudo LRU
11 CPCLA	Cache Line Locking APU available. The CPC responds to a CT field equal to 1. 1 Available
12 -	This field is reserved. Reserved
13–17 CPCNWAY	Cache number of ways. The actual value is the number of ways - 1. 01111 16 way set associative
18–31 CPCSIZE	Cache size in 64 Kbyte increments. A value of 1 denotes a 64-Kbyte cache 00_0000_0000_1000 512 Kbytes

8.2.3 CPC external write control register n (CPC_CPCEWCR*n*)

Note that when using interleaving, the external write control registers should have the same settings.

Address: 1_0000h base + 10h offset + (16d × i), where i=0d to 1d



CPC_CPCEWCR*n* field descriptions

Field	Description
0 E	Enable
1 LOCK	Lock
2–9 -	This field is reserved. Reserved
10–15 SIZE	Size of the address range used for implicit stashing. This field is used to generate an address mask that is applied to both the transaction address and the address base formed from EADDRBASE and ADDRBASE before comparing the two. The size of the address range is equal to 2^{SIZE+6}. 000000 64 bytes 000001 128 bytes 000010 256 bytes 000011 512 bytes 000100 1 Kbytes 000101 2 Kbytes 000110 4 Kbytes 000111 8 Kbytes 001000 16 Kbytes 001001 32 Kbytes 001010 64 Kbytes 001011 128 Kbytes 001100 256 Kbytes

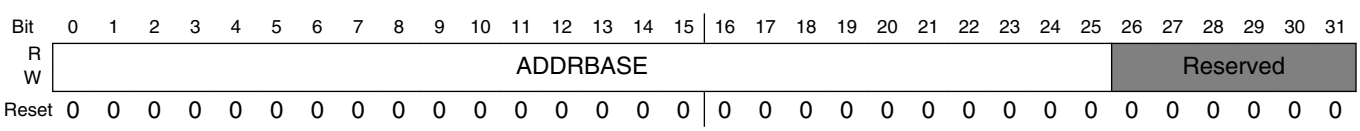
Table continues on the next page...

CPC_CPCEWCR*n* field descriptions (continued)

Field	Description
	001101 512 Kbytes 001110 1 Mbytes 001111 2 Mbytes 010000 4 Mbytes 010001 8 Mbytes 010010 16 Mbytes 010011 32 Mbytes 010100 64 Mbytes 010101 128 Mbytes 010110 256 Mbytes 010111 512 Mbytes 011000 1 Gbytes 011001 2 Gbytes 011010 4 Gbytes 011011 8 Gbytes 011100 16 Gbytes 011101 32 Gbytes 011110 64 Gbytes 011111 128 Gbytes
16–23 -	This field is reserved. Reserved
24–31 EADDRBASE	Extended base address corresponding to physical address 0:7.

8.2.4 CPC external write base address register n (CPC_CPCEWBAR*n*)

Address: 1_0000h base + 14h offset + (16d × i), where i=0d to 1d



CPC_CPCEWBAR*n* field descriptions

Field	Description
0–25 ADDRBASE	Base address corresponding to physical address 8:33.
26–31 -	This field is reserved. Reserved

8.2.5 CPC SRAM control register 1 (CPC_CPCSRCR1)

The CPC SRAM control register 1(CPCSRCR1) determines the upper base address when the CPC is used as SRAM.

Address: 1_0000h base + 100h offset = 1_0100h

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																								SRBARU							
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

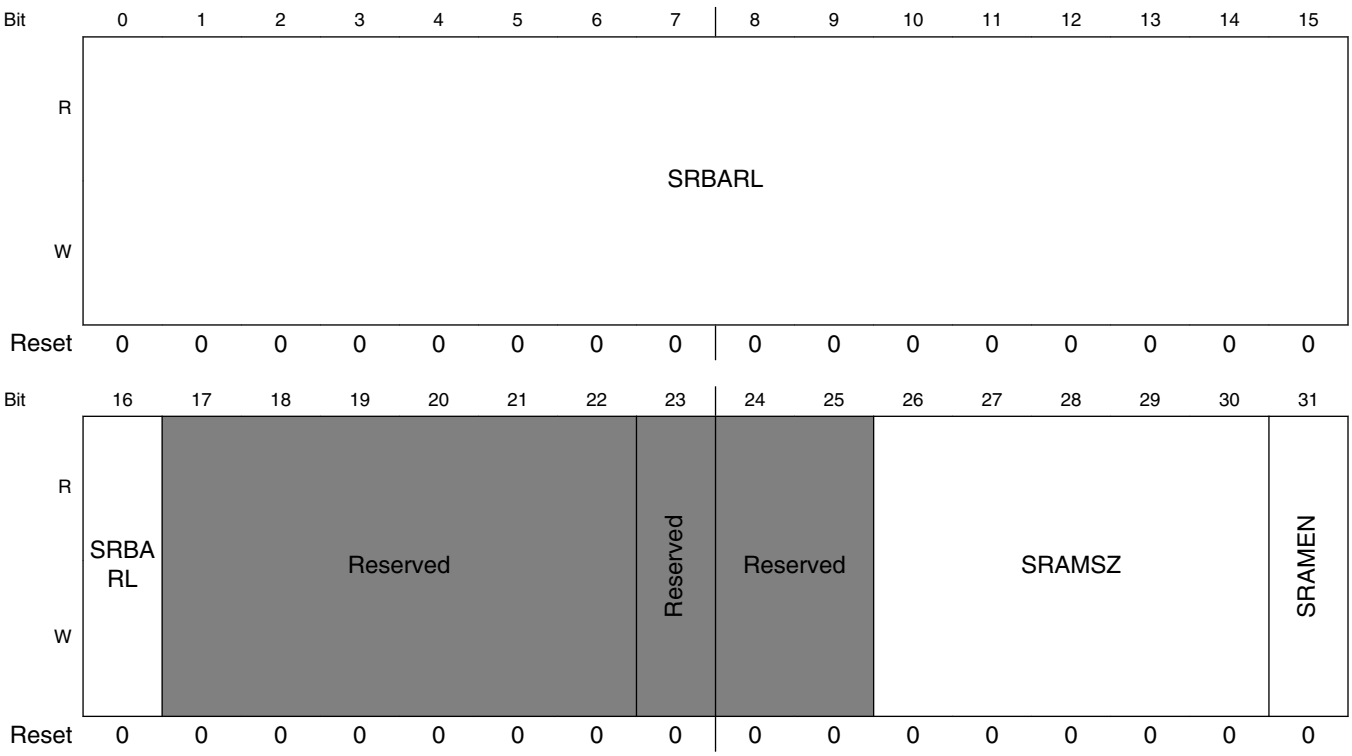
CPC_CPCSRCR1 field descriptions

Field	Description
0–23 -	This field is reserved. Reserved
24–31 SRBARU	SRAM base address high. Corresponds to physical address bits 0:7.

8.2.6 CPC SRAM control register 0 (CPC_CPCSRCR0)

The CPC SRAM control register 0 (CPCSRCR0) configures the CPC when it is used as SRAM.

Address: 1_0000h base + 104h offset = 1_0104h



CPC_CPCSRCR0 field descriptions

Field	Description
0–16 SRBARL	SRAM base address low. Corresponds to physical address bits 8:24. The base address should be aligned to the size specified in SRAMSZ.
17–22 -	This field is reserved. Reserved
23 -	This field is reserved. Reserved
24–25 -	This field is reserved. Reserved
26–30 SRAMSZ	SRAM size. This determines the size of the SRAM space . All other settings not shown are reserved. 00001 64 Kbytes,uses ways 14-15 for SRAM 00011 256 Kbytes,uses ways 8-15 for SRAM 00100 512 Kbytes uses ways 0-15 for SRAM

Table continues on the next page...

CPC_CPCSRCR0 field descriptions (continued)

Field	Description
31 SRAMEN	SRAM mode enable

8.2.7 CPC partition ID register 0 (CPC_CPCPIR0)

Address: 1_0000h base + 200h offset = 1_0200h

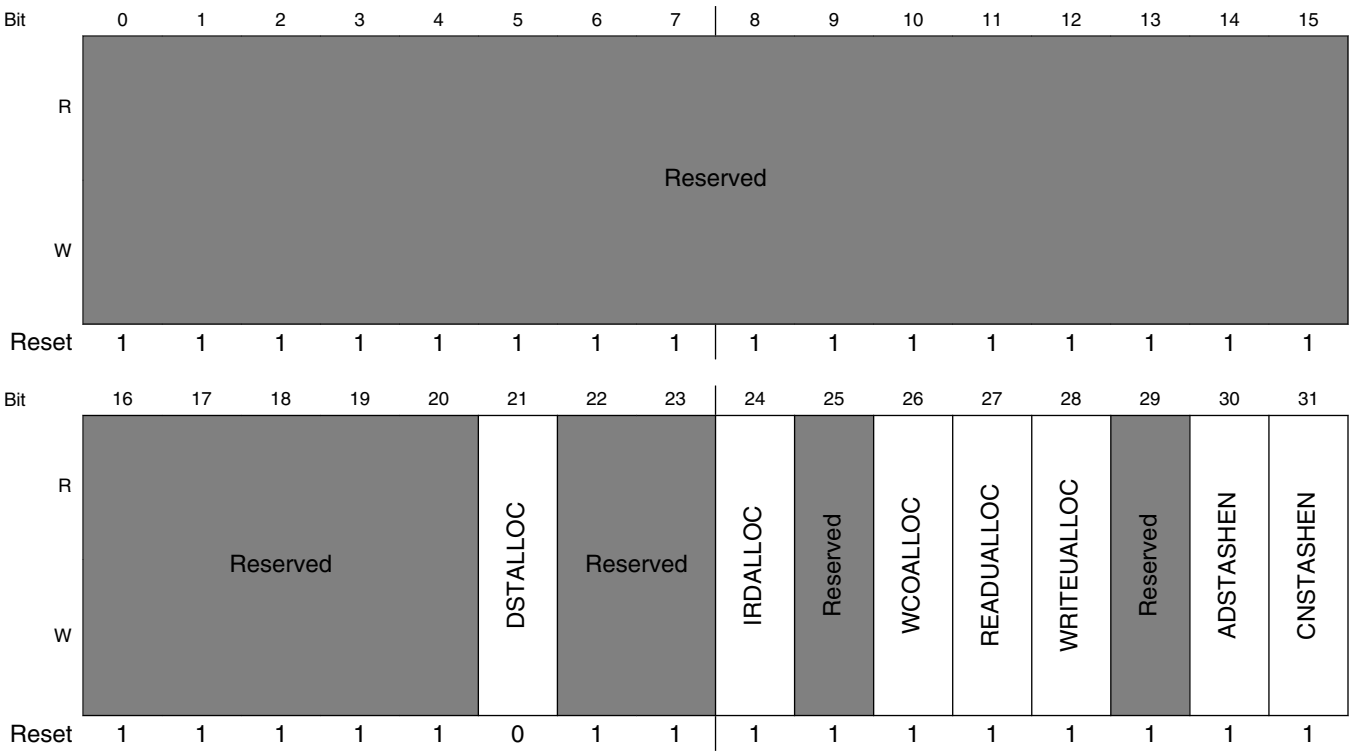
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	PID																															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

CPC_CPCPIR0 field descriptions

Field	Description
0–31 PID	Partition ID A 1 in a particular bit position indicates that the particular partition ID can match on this entry. For example, a PID of 0xF001_0000 indicates that transactions with partition IDs of 0, 1, 2, 3 and 15 can match on this table entry. Partition ID equates to the CSD_ID assigned by the LAWs. See Local Access Window (LAW) Memory Map.

8.2.8 CPC partition allocation register 0 (CPC_CPCPAR0)

Address: 1_0000h base + 208h offset = 1_0208h



CPC_CPCPAR0 field descriptions

Field	Description
0–20 -	This field is reserved. Reserved
21 DSTALLOC	Data store allocation control 0 Transactions resulting from a Store operation that missed in a core cache will not allocate. 1 Transactions resulting from a Store operation that missed in a core cache will attempt to allocate in one of the ways defined by the corresponding CPCPWR *n* .
22–23 -	This field is reserved. Reserved
24 IRDALLOC	Instruction read allocation control 0 Instruction read transactions that miss in the CPC will not allocate. 1 Instruction read transactions that miss in the CPC will attempt to allocate in one of the ways defined by the corresponding CPCPWR *n* .
25 -	This field is reserved. Reserved
26 WCOALLOC	Writes carrying cast-out data from core caches transaction allocation control **NOTE:** Setting this bit and clearing IRDALLOC and DRDALLOC will effectively make the CPC act like a victim cache for CoreNet masters that contain write-back caches.

Table continues on the next page...

CPC_CPCPAR0 field descriptions (continued)

Field	Description
	0 Writes carrying cast-out data from core caches transactions that miss in the CPC will not allocate. 1 Writes carrying cast-out data from core caches transactions that miss in the CPC will attempt to allocate in one of the ways defined by the corresponding CPCPWR *n*.
27 READUALLOC	Decorated Load operations allocation control. See Decorated Storage 0 Decorated Load operations that miss in the CPC will not allocate. 1 Decorated Load operations that miss in the CPC will attempt to allocate in one of the ways defined by the corresponding CPCPWR *n* .
28 WRITEUALLOC	Decorated Store operations allocation control. See Decorated Storage 0 Decorated Store operations that miss in the CPC will not allocate. 1 Decorated Store operations that miss in the CPC will attempt to allocate in one of the ways defined by the corresponding CPCPWR *n* .
29 -	This field is reserved. Reserved
30 ADSTASHEN	Address based stashing enabled 0 Transactions that miss in the CPC and match one of the address ranges defined by CPCEWCR *n* and CPCEWABR *n* will not allocate. 1 Transactions that miss in the CPC and match one of the address ranges defined by CPCEWCR *n* and CPCEWABR *n* will attempt to allocate in one of the ways defined by the corresponding CPCPWR *n* .
31 CNSTASHEN	Explicitly signalled stashing enabled 0 Transactions that miss in the CPC and request stashing will not allocate. 1 Transactions that miss in the CPC and request stashing will attempt to allocate in one of the ways defined by the corresponding CPCPWR *n* .

8.2.9 CPC partition way register 0 (CPC_CPCPWR0)

Address: 1_0000h base + 20Ch offset = 1_020Ch

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	WAY																															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

CPC_CPCPWR0 field descriptions

Field	Description
0–31 WAY	Partition way(s) A 1 in a bit position indicates that transactions that match the PID defined in the corresponding CPCPIR *n* register can allocate the indicated way. **NOTE:** This field in no way prevents a transaction to any partition ID from hitting to a specified way and either reading or writing data.

8.2.10 CPC partition ID register n (CPC_CPCPIR*n*)

Address: 1_0000h base + 210h offset + (16d × i), where i=0d to 14d

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	PID																															
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

CPC_CPCPIR*n* field descriptions

Field	Description
0–31 PID	Partition ID A 1 in a particular bit position indicates that the particular partition ID can match on this entry. For example, a PID of 0xF001_0000 indicates that transactions with partition IDs of 0, 1, 2, 3 and 15 can match on this table entry. Partition ID equates to the CSD_ID assigned by the LAWs. See Local Access Window (LAW) Memory Map.

8.2.11 CPC partition allocation register n (CPC_CPCPAR*n*)

Address: 1_0000h base + 218h offset + (16d × i), where i=0d to 14d

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					DSTALLOC	Reserved		IRDALLOC	Reserved	WCOALLOC	READUALLOC	WRITEUALLOC	Reserved	ADSTASHEN	CNSTASHEN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CPC_CPCPAR*n* field descriptions

Field	Description
0–20 -	This field is reserved. Reserved
21 DSTALLOC	Data store allocation control 0 Transactions resulting from a Store operation that missed in a core cache will not allocate. 1 Transactions resulting from a Store operation that missed in a core cache will attempt to allocate in one of the ways defined by the corresponding CPCPWR*n* .
22–23 -	This field is reserved. Reserved
24 IRDALLOC	Instruction read allocation control 0 Instruction read transactions that miss in the CPC will not allocate. 1 Instruction read transactions that miss in the CPC will attempt to allocate in one of the ways defined by the corresponding CPCPWR *n* .
25 -	This field is reserved. Reserved
26 WCOALLOC	Writes carrying cast-out data from core caches transaction allocation control **NOTE:** Setting this bit and clearing IRDALLOC and DRDALLOC will effectively make the CPC act like a victim cache for CoreNet masters that contain write-back caches. 0 Writes carrying cast-out data from core caches transactions that miss in the CPC will not allocate. 1 Writes carrying cast-out data from core caches transactions that miss in the CPC will attempt to allocate in one of the ways defined by the corresponding CPCPWR *n* .
27 READUALLOC	Decorated Load operations transaction allocation control. See Decorated Storage 0 Decorated Load operations transactions that miss in the CPC will not allocate. 1 Decorated Load operations transactions that miss in the CPC will attempt to allocate in one of the ways defined by the corresponding CPCPWR*n*.
28 WRITEUALLOC	Decorated Store operations transaction allocation control. See Decorated Storage 0 Decorated Store operations that miss in the CPC will not allocate. 1 Decorated Store operations that miss in the CPC will attempt to allocate in one of the ways defined by the corresponding CPCPWR *n* .
29 -	This field is reserved. Reserved
30 ADSTASHEN	Address based stashing enabled 0 Transactions that miss in the CPC and match one of the address ranges defined by CPCEWCR *n* and CPCEWABR *n* will not allocate. 1 Transactions that miss in the CPC and match one of the address ranges defined by CPCEWCR *n* and CPCEWABR *n* will attempt to allocate in one of the ways defined by the corresponding CPCPWR *n* .
31 CNSTASHEN	Explicitly signalled stashing enabled 0 Transactions that miss in the CPC and request stashing will not allocate. 1 Transactions that miss in the CPC and request stashing will attempt to allocate in one of the ways defined by the corresponding CPCPWR *n* .

8.2.12 CPC partition way register 0 (CPC_CPCPWR*n*)

Address: 1_0000h base + 21Ch offset + (16d × i), where i=0d to 14d

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	WAY																															
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

CPC_CPCPWR*n* field descriptions

Field	Description
0–31 WAY	Partition way(s) A 1 in a bit position indicates that transactions that match the PID defined in the corresponding CPCPIR *n* register can allocate the indicated way. **NOTE:** This field in no way prevents a transaction to any partition ID from hitting to a specified way and either reading or writing data.

8.2.13 CPC error injection high register (CPC_CPCERRINJHI)

Address: 1_0000h base + E00h offset = 1_0E00h

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	EIMASKHI																															
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

CPC_CPCERRINJHI field descriptions

Field	Description
0–31 EIMASKHI	Error injection mask for high word. When a bit is set and CPCERRINJCTL[DERRIEN] = 1, the corresponding bit in the data path is inverted during data array writes.

8.2.14 CPC error injection low register (CPC_CPCERRINJLO)

Address: 1_0000h base + E04h offset = 1_0E04h

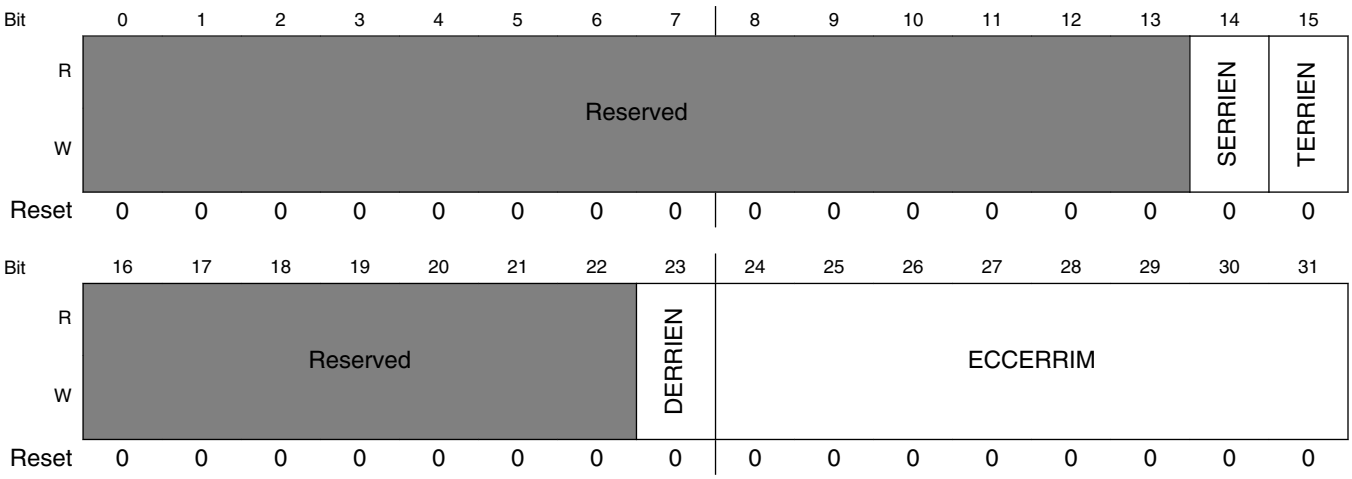
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	EIMASKLO																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CPC_CPCERRINJLO field descriptions

Field	Description
0–31 EIMASKLO	Error injection mask for low word. When a bit is set and CPCERRINJCTL[DERRIEN] = 1, the corresponding bit in the data path is inverted during data array writes.

8.2.15 CPC error injection control register (CPC_CPCERRINJCTL)

Address: 1_0000h base + E08h offset = 1_0E08h



CPC_CPCERRINJCTL field descriptions

Field	Description
0–13 -	This field is reserved. Reserved
14 SERRIEN	Status error injection enable. Status error injection is determined by CPCERRINJLO[23:31]; status ECC error injection is determined by the five least-significant bits of ECCERRIM (that is, CPCERRINJCTL[27:31]). 0 No status errors are injected 1 Subsequent entries written to the CPC status arrays have status or ECC bits inverted as specified in the status and ECC error injection masks.
15 TERRIEN	Tag error injection enable. Tag error injection is determined by CPCERRINJLO[7:31]; tag ECC error injection is determined by the seven least-significant bits of ECCERRIM (that is, CPCERRINJCTL[25:31]). 0 No tag errors are injected 1 Subsequent entries written to the CPC tag arrays have tag or ECC bits inverted as specified in the data and ECC error injection masks.
16–22 -	This field is reserved. Reserved
23 DERRIEN	CPC data array error injection enable. Data error injection is determined by CPCERRINJHI[0:31] and CPCERRINJLO[0:31]; data ECC error injection is determined by ECCERRIM. 0 No data errors are injected 1 Subsequent entries written to the CPC data arrays have data or ECC bits inverted as specified in the data and ECC error injection masks.
24–31 ECCERRIM	Error injection mask for the ECC bits. When DERRIEN=1, the eight ECCERRIM bits map to the eight ECC bits for each 64 bits of data. When TERRIEN=1, the low order seven ECCERRIM bits map to the seven ECC bits for each 25-bit tag. When SERRIEN=1, the low order five ECCERRIM bits map to the five ECC bits for each 9-bit tag.

8.2.16 CPC capture data high register (CPC_CPCCAPTDATAHI)

Address: 1_0000h base + E20h offset = 1_0E20h

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	DATAH																															
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

CPC_CPCCAPTDATAHI field descriptions

Field	Description
0–31 DATAH	If error is data ECC, then bits 0-31 are the upper order 32 bits of data associated with the error. If the error is tag ECC, then bits 0-15 are zeros and bits 16-31 are the upper order 16 bits of the tag associated with the error. If the error is status ECC, then this field is all zeros.

8.2.17 CPC capture data low register (CPC_CPCCAPTDATALO)

Address: 1_0000h base + E24h offset = 1_0E24h

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	DATAL																															
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

CPC_CPCCAPTDATALO field descriptions

Field	Description
0–31 DATAL	If error is data ECC, the bits 0-31 are the lower order 32 bits of data associated with the error. If the error is tag ECC, then bits 0-16 are the lower order 17 bits of the tag associated with the error and bits 17-31 are zeros. If the error is status ECC, then bits 0-22 are zeros and bits 23-31 are the status bits associated with the error.

8.2.18 CPC capture ECC register (CPC_CPCCAPTECC)

Address: 1_0000h base + E28h offset = 1_0E28h

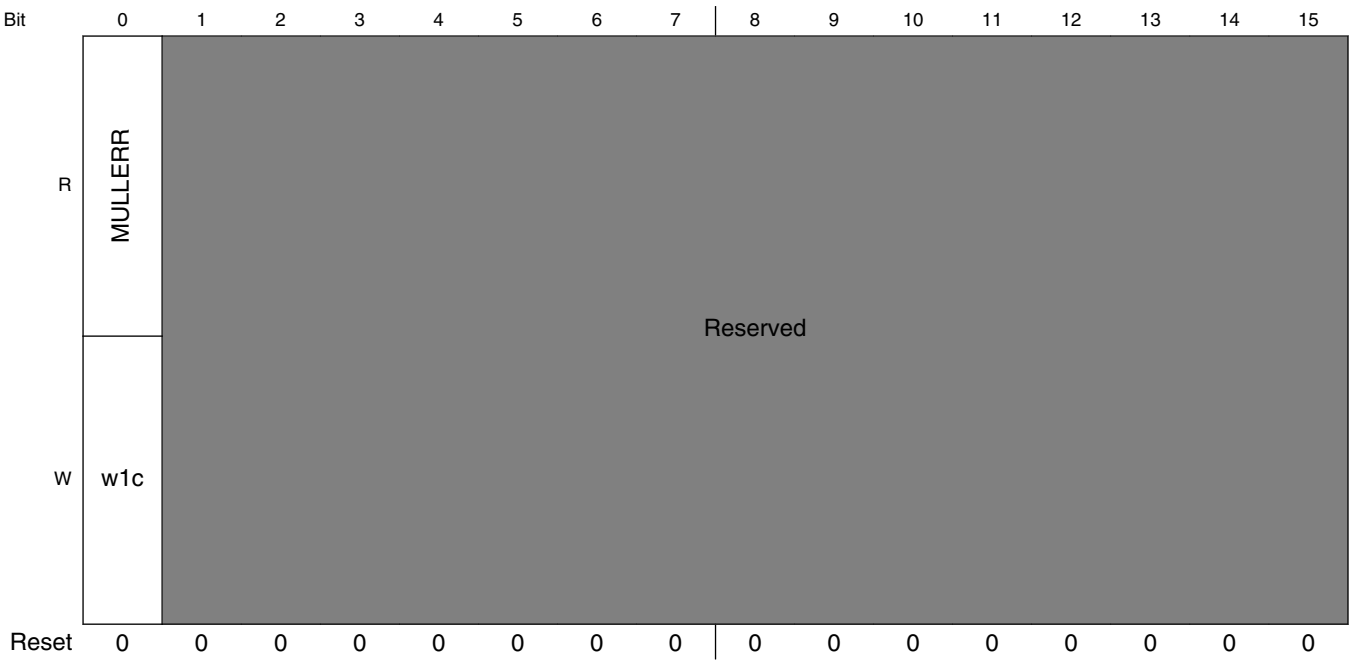
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	ECCSYND								Reserved																ECCCHKSUM							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

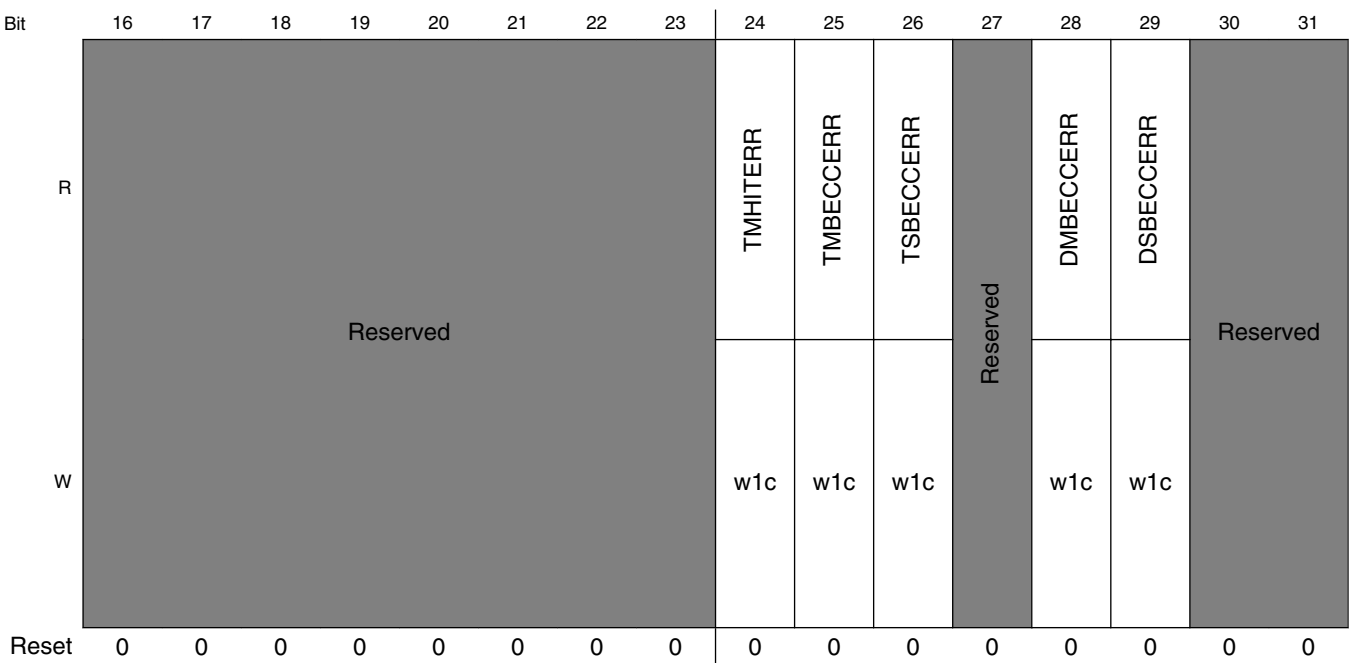
CPC_CPCCAPTECC field descriptions

Field	Description
0–7 ECCSYND	The calculated ECC syndrome of the captured error. Tag and status ECC are left padded with 1 or 3 zeros, respectively.
8–23 -	This field is reserved. Reserved
24–31 ECCCHKSUM	The stored ECC syndrome of the captured error. Tag and status ECC are left padded with 1 or 3 zeros, respectively.

8.2.19 CPC error detect register (CPC_CPCERRDET)

Address: 1_0000h base + E40h offset = 1_0E40h





CPC_CPCERRDET field descriptions

Field	Description
0 MULLERR	Multiple CPC errors

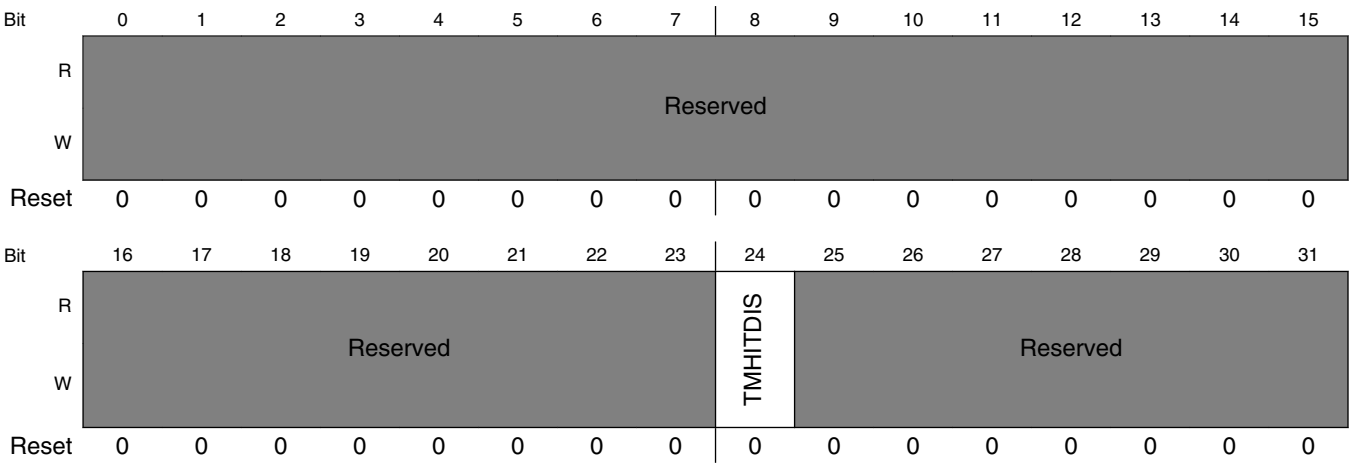
Table continues on the next page...

CPC_CPCERRDET field descriptions (continued)

Field	Description
	0 Multiple CPC errors of the same type were not detected. 1 Multiple CPC errors to the same type were detected.
1–23 -	This field is reserved. Reserved
24 TMHITERR	Tag multi-way hit 0 Tag multi-way hit error not detected. 1 Tag multi-way hit error detected.
25 TMBECCERR	Tag or status multiple-bit ECC error. CPCERRATTR[Status_Tag] indicates whether the error occurred in the tag or status. 0 Tag/status multiple-bit ECC error not detected. 1 Tag/status multiple-bit ECC error detected.
26 TSBECCERR	Tag or status single-bit ECC error. CPCERRATTR[Status_Tag] indicates whether the error occurred in the tag or status. 0 Tag/status single-bit ECC error not detected. 1 Tag/status single-bit ECC error detected.
27 -	This field is reserved. Reserved
28 DMBECCERR	Data multiple-bit ECC error 0 Data multiple-bit ECC error not detected. 1 Data multiple-bit ECC error detected.
29 DSBECCERR	Data single-bit ECC error 0 Data single-bit ECC error not detected. 1 Data single-bit ECC error detected.
30–31 -	This field is reserved. Reserved

8.2.20 CPC error disable register (CPC_CPCERRDIS)

Address: 1_0000h base + E44h offset = 1_0E44h

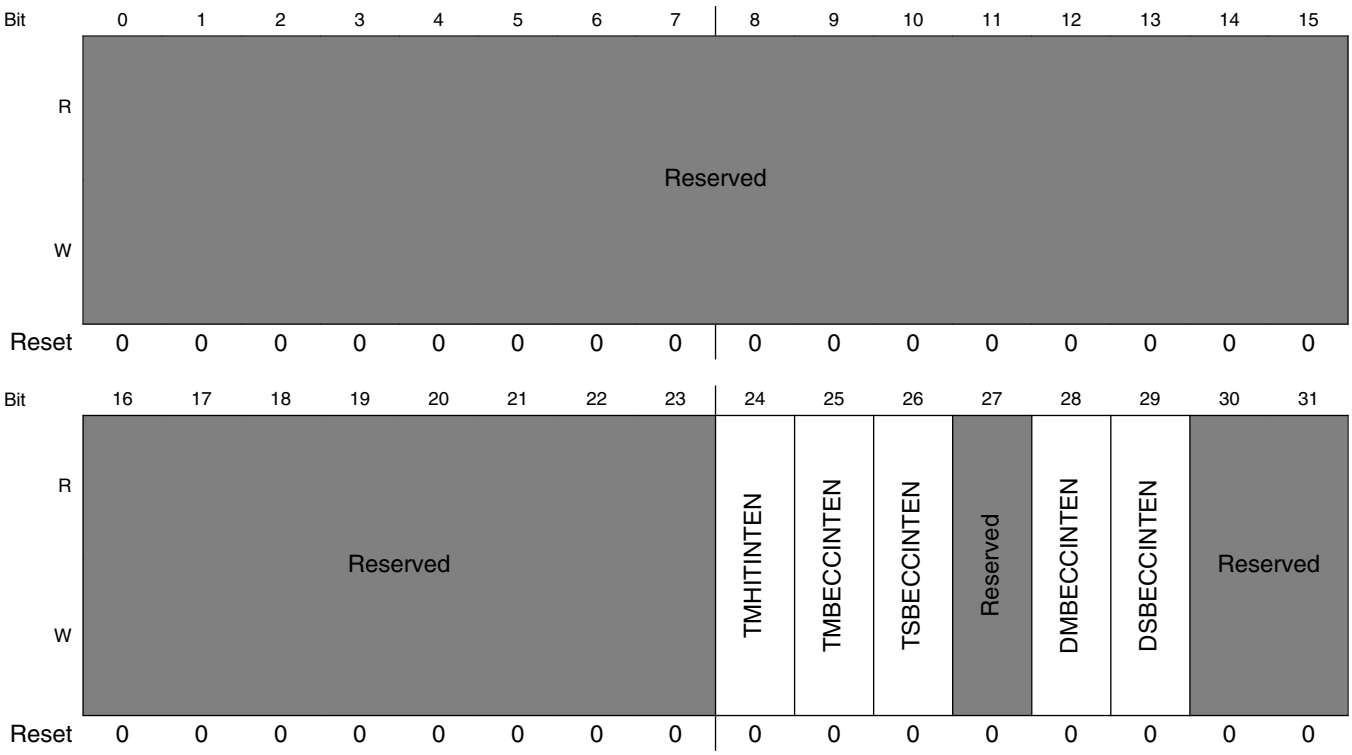


CPC_CPCERRDIS field descriptions

Field	Description
0–23 -	This field is reserved. Reserved
24 TMHITDIS	Tag multi-way hit error disable. 0 Tag multi-way hit detection enabled. 1 Tag multi-way hit detection disabled.
25–31 -	This field is reserved. Reserved

8.2.21 CPC error interrupt enable register (CPC_CPCERRINTEN)

Address: 1_0000h base + E48h offset = 1_0E48h



CPC_CPCERRINTEN field descriptions

Field	Description
0–23 -	This field is reserved. Reserved
24 TMHITINTEN	Tag multi-way hit error interrupt reporting enable. 0 Tag multi-way hit errors are not reported. 1 Tag multi-way hit errors are reported.
25 TMBECCINTEN	Tag multiple-bit ECC error interrupt reporting enable. 0 Tag multiple-bit ECC errors are not reported. 1 Tag multiple-bit ECC errors are reported.
26 TSBECCINTEN	Tag single-bit ECC error interrupt reporting enable. 0 Tag single-bit ECC errors are not reported. 1 Tag single-bit ECC errors are reported.
27 -	This field is reserved. Reserved
28 DMBECCINTEN	Data multiple-bit ECC error interrupt reporting enable. 0 Data multiple-bit ECC errors are not reported. 1 Data multiple-bit ECC errors are reported.

Table continues on the next page...

CPC_CPCERRINTEN field descriptions (continued)

Field	Description
29 DSBECCINTEN	Data single-bit ECC error interrupt reporting enable. 0 Data single-bit ECC errors are not reported. 1 Data single-bit ECC errors are reported.
30–31 -	This field is reserved. Reserved

8.2.22 CPC error extended address register (CPC_CPCERREADDR)

Address: 1_0000h base + E50h offset = 1_0E50h

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																								ADDRH							
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

CPC_CPCERREADDR field descriptions

Field	Description
0–23 -	This field is reserved. Reserved
24–31 ADDRH	High-order 8 bits of the physical address (PA[0:7]) associated with a captured error.

8.2.23 CPC error address register (CPC_CPCERRADDR)

Address: 1_0000h base + E54h offset = 1_0E54h

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	ADDRL																															
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

CPC_CPCERRADDR field descriptions

Field	Description
0–31 ADDRL	Lower-order 32 bits of the physical address (PA[8:39]) associated with a captured error.

8.2.24 CPC error control register (CPC_CPCERRCTL)

Address: 1_0000h base + E58h offset = 1_0E58h

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								CPCTHRESH								CPCTCOUNT								CPCDCOUNT							
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

CPC_CPCERRCTL field descriptions

Field	Description
0–7 -	This field is reserved. Reserved
8–15 CPCTHRESH	CPC ECC single-bit error threshold. Threshold value for the number of ECC single-bit errors that are detected before reporting an error condition. CPCTHRESH is compared to CPCTCOUNT and CPCCOUNT each time a single-bit ECC error is detected. A value of 0 in this field will cause the reporting of a single bit ECC error upon the first occurrence of such an error.
16–23 CPCTCOUNT	CPC tag ECC single-bit error count. Counts ECC single-bit errors detected in the CPC tags. If CPCTCOUNT equals the ECC single-bit error threshold (CPCTHRESH), an error is reported if single-bit error reporting for tags is enabled. Software should clear this value when such an error is reported to reset the count.
24–31 CPCDCOUNT	CPC data ECC single-bit error count. Counts ECC single-bit errors detected in the CPC data. If CPCCOUNT equals the ECC single-bit error threshold (CPCTHRESH), an error is reported if single-bit error reporting for data is enabled. Software should clear this value when such an error is reported to reset the count.

8.2.25 CPC hardware debug control register 0 (CPC_CPCHDBCR0)

Address: 1_0000h base + F00h offset = 1_0F00h

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

CPC_CPCHDBCR0 field descriptions

Field	Description
0–3 -	This field is reserved. Reserved
4 SPEC_DIS	Speculation disable 0 Read type requests speculatively pass to the memory controllers before determining whether there is a hit in the CPC. **NOTE:** This bit should be set to a 1 if the CPC SRAM is enabled and the SRAM address space does not overlap an enabled address region in the DRAM controller. 1 Read type requests must wait to determine hit in the CPC before making a memory controller request.
5–31 -	This field is reserved. Reserved

8.3 CPC Functional Description

8.3.1 Register Sections

8.3.1.1 SRAM Mode Registers

The CoreNet platform cache can be configured to use some of its ways as direct-mapped SRAM address space. The SRAM mode registers enable and configure the CPC to be used as SRAM space. SRAM mode is enabled by CPCSRCR0[SRAMEN], the size of the SRAM address space is configured by CPCSRCR0[SRAMSZ], and the base address of the SRAM address space is specified using CPCSRCR0[SRBARL] and CPCSRCR1[SRBARU].

If DRAM addresses are interleaved (as controlled through the DDR controller configuration registers), then SRAM address space can also be interleaved at the same address boundary by setting CPCSRCR0[INTLVEN]. If DDR is not interleaved, then SRAM address space cannot be interleaved. Enabling SRAM interleaving requires that all instances of the CPC in the system that participate in interleaving be programmed to the same SRAM size and base address. Additionally, the interleave granularity must be less than or equal to the size of an individual CPC's SRAM. Enabling SRAM interleaving effectively doubles the total SRAM address space.

The CPC sends speculative requests (that is, before the CPC determines hit or miss) to the memory controller. If the CPC is configured as SRAM space, the memory controller may not respond to speculative SRAM requests that don’t hit DRAM (or may generate an error). Therefore, when the CPC is configured as SRAM and the SRAM address space

does not overlap an enabled memory controller address region, disable speculative requests by setting CPCHDBCR0[SPEC_DIS]. See CPC hardware debug control register 0 (CPC_CPCHDBCR0) for more information.

8.3.1.2 Partitioning Control Registers

The CPC supports a flexible way partition/allocation control scheme. Each transaction that misses in the cache looks in a table to determine whether or not to allocate and which ways are available for allocation. Table entries are matched by comparing Partition IDs. Allocation is then controlled dependent on transaction type and address qualifiers as well as stashing control signals.

Each partition control is composed of three registers: CPCPIR*n*, CPCPAR*n* and CPCPWR*n*. There are a total of 16 triples of these registers.

8.3.1.3 Error Registers

8.3.1.3.1 Error Detection and Reporting

Error detection for tags and data are enabled by default.

When an error is detected in the CPC the appropriate bit in the CPC error detect register (CPCERRDET) is set and, if enabled by CPCERRINTEN, a machine check interrupt is generated for software to notice and handle the error. The generation of the interrupt when an error is detected is controlled by the CPC error interrupt enable register (CPCERRINTEN). Bits in this register control whether an interrupt is to be generated based on the type and location of the error in the CPC. Software may read from the CPCERRDET and may clear bits by writing a 1 into a bit position, but cannot directly write the contents of the register. CPCERRDET is described in CPC error detect register (CPC_CPCERRDET); CPCERRINTEN is described in CPC error interrupt enable register (CPC_CPCERRINTEN).

Single-bit errors that are corrected by ECC are not reported unless the appropriate bit in the CPCERRDET register is set and the number of errors reach a threshold specified in CPCERRCTL, described in CPC error control register (CPC_CPCERRCTL).This prevents normal correctable ECC errors from generating machine check exceptions, but will still cause errors to be generated when there may be a persistent faulty condition in the CPC.

The machine check exception is generated based on the CPCERRDET status bits and CPCERRINTEN enable bits. If an CPCERRDET status bit and its corresponding CPCERRINTEN enable bit are set at the same time, a machine check exception is

generated. Therefore, upon taking the machine check, software should clear CPCERRDET before setting MSR[ME], otherwise another machine check can occur when MSR[ME] is set.

8.3.1.3.2 Error Capture

When an error is detected and reported, information about the error is posted into several CPC error capture registers (CPCERRADDR, CPCERREADDR, CPCERRATTR, CPCCAPTDATAHI, CPCCAPTDATALO, and CPCCAPTECC). Only the first reported error information is saved. After software has examined the error information, it should clear CPCERRATTR[VALINFO] to allow the next detected error to have information posted. The error capture registers are as follows:

- CPCERRADDR and CPCERREADDR contain the physical address associated with the captured error. CPCERRADDR and CPCERREADDR are described in CPC error extended address register (CPC_CPCERREADDR) and CPC error address register (CPC_CPCERRADDR).
- CPCCAPTDATAHI and CPCCAPTDATALO contain data or tag value associated with the captured error and are described in CPC error injection high register (CPC_CPCERRINJHI) and CPC error injection low register (CPC_CPCERRINJLO).
- CPCCAPTECC contains ECC information from the data or tag (the syndrome and the checksum) and is described in CPC capture ECC register (CPC_CPCCAPTECC).

8.3.1.3.3 Error Injection

The CPC includes support for injecting errors into the data, data ECC, tag, tag ECC, status and status ECC. This may be used to test error recovery software by deterministically creating error scenarios. The error injection registers are as follows:

- Data error is injected into the line, according to the error injection settings in CPCERRINJHI, CPCERRINJLO, and CPCERRINJCTL, at allocation. Tag error injection is determined by CPCERRINJLO and CPCERRINJCTL, at allocation. Note that error injection enable bits in CPCERRINJCTL must be cleared by software and the CPC must be invalidated (by setting CPCCSR0[CPCFI]) before resuming CPC normal operation.
- Data, tag and status use the same error injection masks as defined by CPCERRINJHI, CPCERRINJLO and CPCERRINJCTL, but have separate error injection enables as defined in CPCERRINJCTL.
- CPCERRINJCTL, CPCERRINJLO, and CPCERRINJHI are described in CPC error injection control register (CPC_CPCERRINJCTL), CPC error injection low register

(CPC_CPCERRINJLO), and CPC error injection high register (CPC_CPCERRINJHI)."

8.3.2 Line Locking

The CoreNet platform cache supports persistent cache locking as defined by Freescale Book E Implementation Standards for Storage (EIS Storage). A single lock bit is associated with each coherency granule stored in the cache.

The CPC supports explicit lock and unlock commands carried through transactions. Processor master devices can use **dcbtls, dcbtstls, icbtls, dcblc, icblc** instructions to generate LoadEC and Unlock transactions that target the CPC through the respective instructions' CT field. I/O master devices can use a wider variety of transactions that support explicit stashing transaction attributes.

The CPC also supports implicit locking through specified address ranges as defined by the Stashing Control Registers. See CPC external write control register n (CPC_CPCEWCR*n*).

8.3.3 Cache Operation Instructions and Transactions

Even though the CoreNet platform cache is not directly tied to a processor, it does respect the traditional cache operation instructions through their CoreNet transaction. **dcbf** instructions cause Flush transactions on CoreNet and the CPC performs the appropriate action in order to flush any dirty data to memory. **dcbst** instructions cause Clean transactions on CoreNet and the CPC performs the appropriate action in order to clean any dirty data to memory, while leaving a valid copy in the cache. **dcbi** instructions cause Kill transactions on CoreNet and the CPC performs the appropriate action in order to invalidate the addressed coherency granule.

icbi instructions cause Kill transactions on CoreNet. Since the CPC is a unified cache, Kill transactions have no affect on the state of the addressed coherency granule in the CPC.

Note that cache management transactions such as Flush, Clean, and Kill have no affect to addresses configured as SRAM.

8.3.4 Decorated Storage

The *Freescale Book E Implementation Standards for Storage*, referred to as the EIS, defines an extension to the base Power Architecture instruction set called the Decorated Storage APU. This APU defines new instructions that include an attribute called a decoration that provides additional semantics for the device. The CoreNet platform cache supports the Decorated Storage APU as defined in EIS through specially-coded CoreNet transactions. Note that the decoration does not have any direct meaning to the core itself, but is interpreted by the CPC on behalf of the memory complex. A decorated operation carries up to four parameters:

- Type of access (load, store, or notify)
- Address
- Data (for stores)
- Decoration (encoding that defines the operation)

Decorated storage is only supported to addresses mapped by big-endian memory pages. In addition, the decorated storage operations are performed only on addresses that have been marked as caching-inhibited and guarded.

8.3.4.1 Decorated Load Operations

Decorated load operations read data from the memory complex (either from the CPC data arrays or from DDR), returns it to the core, and then performs a selectable update to the data in the memory complex. The overall flow through the data pipeline is similar to a read-modify-write (RMW) operation. The main difference is in how the new write data is generated. A decorated load operation can perform four different operations (clear, increment, decrement or set) on four different data types (1, 2, 4 or 8 bytes). The timing of these operations is the same as for RMW.

8.3.4.2 Decorated Store Operations

Decorated store operations are similar to decorated loads. The main difference is that decorated stores do not return data to the core and the update operations can perform an add and subtract (instead of just a increment or decrement) and minimum and maximum functions using data received from the core. The timing of these operations is the same as for RMW.

Decorated store operations require that the write data be presented in the proper byte lanes as required to perform the desired operation. Table 8-75 shows the address alignment restrictions for the specific decorated store instructions. Note that the maximum and minimum functions only support one size value for each decoration value.

If a decorated store does not carry the proper address alignment for its decoration, the decorated store writes back the existing data to the memory complex (that is, it will nop).

Table 8-75 describes the decorated storage instructions with supported decorations and the operations that are performed as a result.

Table 8-75. Decorated Storage Operations

Decorated Storage Instruction	Decoration (rA)	Address Alignment Addr[60:63][1]	Update Operation[2]	Function
lbdx	0000	-	MEM(addr[ ])[0:7] = 0x00	Clear byte
	0001	-	MEM(addr[ ])[0:7] = 0xFF	Set byte
	0010	-	MEM(addr[ ])[0:7]--	Decrement byte
	0011	-	MEM(addr[ ])[0:7]++	Increment byte
lhdx	0000	-	MEM({addr[ ], 0})[0:15] = 0x0000	Clear half-word
	0001	-	MEM({addr[ ], 0})[0:15] = 0xFFFF	Set half-word
	0010	-	MEM({addr[ ], 0})[0:15]--	Decrement half-word
	0011	-	MEM({addr[ ], 00})[0:15]++	Increment half-word
lwdx	0000	-	MEM({addr[ ], 00})[0:31] = 0x0000_0000	Clear word
	0001	-	MEM({addr[ ], 00})[0:31] = 0xFFFF_FFFF	Set word
	0010	-	MEM({addr[ ], 00})[0:31]--	Decrement word
	0011	-	MEM({addr[ ], 00})[0:31]++	Increment word
lfddx	0000	-	MEM({addr[ ], 000})[0:31] = 0x0000_0000_0000_0000	Clear doubleword
	0001	-	MEM({addr[ ], 000})[0:31] = 0xFFFF_FFFF_FFFF_FFFF	Set doubleword
	0010	-	MEM({addr[ ], 000})[0:31]--	Decrement doubleword
	0011	-	MEM({addr[ ], 000})[0:31]++	Increment doubleword
stbdx	0000	0111	MEM({addr[ ], 000})[0:63] += exts64(din[56:63])	Doubleword accumulate with signed byte
		1111	MEM({addr[ ], 000})[0:63] += exts64(din[120:127])	
	0001	0011	MEM({addr[ ], 00})[0:31] += exts32(din[24:31])	Word accumulate with signed byte
		0111	MEM({addr[ ], 00})[0:31] += exts32(din[56:63])	
		1011	MEM({addr[ ], 00})[0:31] += exts32(din[88:95])	
		1111	MEM({addr[ ], 00})[0:31] += exts32(din[120:127])	
	0010	1111	MEM({addr[ ], 0000})[0:63]++ and MEM({addr[ ], 0000})[64:127] += exts64(din[120:127])	Doubleword increment and doubleword accumulate with signed byte

Table continues on the next page...

Table 8-75. Decorated Storage Operations (continued)

Decorated Storage Instruction	Decoration (rA)	Address Alignment Addr[60:63][1]	Update Operation[2]	Function
	0011	0111	MEM({addr[ ], 000})[0:31]++ and MEM({addr[ ], 000})[32:63] += exts32(din[56:63])	Word increment and word accumulate with signed byte
		1111	MEM({addr[ ], 000})[0:31]++ and MEM({addr[ ], 000})[32:63] += exts32(din[120:127])	
sthdx	0000	0110	MEM({addr[ ], 000})[0:63] += exts64(din[48:63])	Doubleword accumulate with signed half-word
		1110	MEM({addr[ ], 000})[0:63] += exts64(din[112:127])	
	0001	0010	MEM({addr[ ], 00})[0:31] += exts32(din[16:31])	Word accumulate with signed half-word
		0110	MEM({addr[ ], 00})[0:31] += exts32(din[48:63])	
		1010	MEM({addr[ ], 00})[0:31] += exts32(din[80:95])	
		1110	MEM({addr[ ], 00})[0:31] += exts32(din[112:127])	
	0010	1110	MEM({addr[ ], 0000})[0:63]++ and MEM({addr[ ], 0000})[64:127] += exts64(din[112:127])	Doubleword increment and doubleword accumulate with signed half-word
	0011	0110	MEM({addr[ ], 000})[0:31]++ and MEM({addr[ ], 000})[32:63] += exts32(din[48:63])	Word increment and word accumulate with signed half-word
		1110	MEM({addr[ ], 000})[0:31]++ and MEM({addr[ ], 000})[32:63] += exts32(din[112:127])	
stwdx	0000	0100	MEM({addr[ ], 000})[0:63] += exts64(din[32:63])	Doubleword accumulate with signed word
		1100	MEM({addr[ ], 000})[0:63] += exts64(din[96:127])	
	0001	0000	MEM({addr[ ], 00})[0:31] += din[0:31]	Word accumulate with signed word
		0100	MEM({addr[ ], 00})[0:31] += din[32:63]	
		1000	MEM({addr[ ], 00})[0:31] += din[64:95]	
		1100	MEM({addr[ ], 00})[0:31] += din[96:127]	
	0010	1100	MEM({addr[ ], 0000})[0:63]++ and MEM({addr[ ], 0000})[64:127] += exts64(din[96:1'tw27])	Doubleword increment and doubleword accumulate with signed word
	0011	0100	MEM({addr[ ], 000})[0:31]++ and MEM({addr[ ], 000})[32:63] += din[32:63]	Word increment and word accumulate with signed word
		1100	MEM({addr[ ], 000})[0:31]++ and MEM({addr[ ], 000})[32:63] += din[96:127]	
	0101	0000	MEM({addr[ ], 00}) = unsigned_max(MEM({addr[ ], 2'b00}), DIn[0:31])	32-bit maximum with unsigned word
		0100	MEM({addr[ ], 00}) = unsigned_max(MEM({addr[ ], 00}), DIn[32:63])	
		1000	MEM({addr[ ], 00}) = unsigned_max(MEM({addr[ ], 00}), DIn[64:95])	

Table continues on the next page...

Table 8-75. Decorated Storage Operations (continued)

Decorated Storage Instruction	Decoration (rA)	Address Alignment Addr[60:63][1]	Update Operation[2]	Function
		1100	MEM({addr[ ], 00}) = unsigned_max(MEM({addr[ ], 00}), DIn[96:127])	
	0111	0000	MEM({addr[ ], 00}) = unsigned_min(MEM({addr[ ], 00}), DIn[0:31])	32-bit minimum with unsigned word
		0100	MEM({addr[ ], 00}) = unsigned_min(MEM({addr[ ], 00}), DIn[32:63])	
		1000	MEM({addr[ ], 00}) = unsigned_min(MEM({addr[ ], 00}), DIn[64:95])	
		1100	MEM({addr[ ], 00}) = unsigned_min(MEM({addr[ ], 00}), DIn[96:127])	
stfddx	0000	0000	MEM({addr[ ], 000})[0:63] += din[0:63]	Doubleword accumulate with signed Doubleword
		1000	MEM({addr[ ], 000})[0:63] += din[64:127]	
	0010	1000	MEM({addr[ ], 0000})[0:63]++ and MEM({addr[ ], 0000})[64:127] += din[64:127]	Doubleword increment and doubleword accumulate with signed doubleword
	0100	0000	MEM({addr[ ], 000}) = unsigned_max(MEM({addr[ ], 000}), DIn[0:63])	64-bit maximum with unsigned doubleword
		1000	MEM({addr[ ], 000}) = unsigned_max(MEM({addr[ ], 000}), DIn[64:127])	
	0110	0000	MEM({addr[ ], 000}) = unsigned_min(MEM({addr[ ], 000}), DIn[0:63])	64-bit minimum with unsigned doubleword
		1000	MEM({addr[ ], 000}) = unsigned_min(MEM({addr[ ], 000}), DIn[64:127])	
dsn	0000	-	MEM({addr[ ], 000})[0:63]++	Doubleword increment
	0001	-	MEM({addr[ ], 00})[0:31]++	Word increment
	0010	-	MEM({addr[ ], 000})[0:63] = 0000_0000_0000_0000	Doubleword clear
	0011	-	MEM({addr[ ], 00})[0:31] = 0000_0000	Word clear

1. Addr[60:63] in this case is the core effective address, not the physical address.
2. The update operations are given as C-code equivalents. The following notes apply: exts64() represents a sign extension of the operand to 64 bits exts32() represents a sign extension of the operand to 32 bits. unsigned_max(a,b)=(a>b) ? a : b, where a and b are unsigned integers of the appropriate size unsigned_min(a,b)=(a>b) ? b : a, where a and b are unsigned integers of the appropriate size

8.3.4.3 Notify Operations

Notify is a simplification of a decorated store operation that does not require data from the core. Because of this, only increment, decrement, set and clear functions are possible. The timing of these operations is the same as for RMW.

8.3.5 Cache/SRAM Data Clear

Upon Hard Fail condition, the security monitor drives an interrupt to the CPC that causes the CPC to immediately initiate a write of all zeros to all locations in the data arrays. These write operations have the highest priority to the data pipeline. This operation is non-coherent and the system may be non-recoverable after this has occurred.

8.4 Initialization/Application Information

8.4.1 Supported SRAM Mode Configurations

Table 8-76. Supported SRAM Mode Configurations

CPC			DDR Controller
SRAM Mode Enabled (SRAMEN)	Interleaving (INTLVEN)	SRAM Size	See Supported DDR Interleaving Configurations for more details.
No	-	-	Any supported configuration
Yes	-	64 Kbytes	Used, but memory controller interleaving disabled
		256 Kbytes	
		512 Kbyte	

8.4.2 Programming Examples

The following section gives examples of how software should configure the CPC control registers for various scenarios.

8.4.2.1 Modifying CPC Control and Status Registers

The sequence for modifying a CPC Control and Status Register is as follows:

1. Ensure that all prior instructions and memory accesses are complete and performed before modifying the CPC register.
2. Perform a store (WIMG = 01xx) to the CPC register to set its desired value.
3. Perform a load (WIMG = 01xx) to the CPC register to push the store to the CPC and ensure it has been performed.

4. Ensure that all subsequent instruction and memory accesses wait for the load to complete, ensuring that the CPC register update has been performed before any subsequent memory accesses.

An example Power Architecture instruction sequence to update CPCCSR0 from a core follows:

1. **mbar**
2. **isync**
3. **stw** (WIMG = 01xx) CCSRBAR+CPC*n* block base address+0x000
4. **lwz** (WIMG = 01xx) CCSRBAR+CPC*n* block base address+0x000
5. **mbar**

8.4.2.2 Enabling the CPC after Power-On Reset

The CPC registers have the reset values defined in the "LAW". Based on these initial register values, the following procedure is performed to invalidate, configure and enable the CPC after power-on reset:

1. Set CPCCSR0[CPCFI], CPCCSR0[CPCLFC] using the same write operation
2. Wait for hardware to clear CPCCSR0[CPCFI] and CPCCSR0[CPCLFC] using a polling loop
3. Set CPC configuration registers, including CPCSRCR1, CPCSRCR0, CPCPIR*n*, CPCPAR*n*, CPCPWR*n*, CPCERRDIS, CPCERRINTEN, and CPCERRCTL
4. Set CPCCSR0 bits to desired values (notably CPCPE, CPCWT, CPCREP and CPCLOA). Note this operation can be combined with step 6.
5. Configure the LAWs and DDR controller; this must be done before enabling the CPC.
6. Set CPCCSR0[CPCE] to enable the CPC.

This procedure needs to be performed for each CPC in the system.

8.4.2.3 Changing the Configuration of an Enabled CPC

In order to change the configuration registers of a CPC that has already been enabled, it is in generally necessary to flush and disable the cache first using the following procedure:

1. Clear all bits of CPCPAR*n* (this prevents any new transactions from allocating into the CPC) using write operation.
2. Flush the CPC using one of the two following methods:

- Harware flush: Set CPCCSR0[CPCFL] and wait for hardware to clear CPCCSR0[CPCFL] by polling
- Software flush: perform a **dcbf** operation to each coherency granule mapped to the memory target (that is, one **dcbf** for each 64 bytes).

3. Clear all lock bits by setting CPCCSR0[CPCLFC].
4. Wait for hardware to clear CPCCSR0[CPCLFC] using a polling loop.
5. Disable the CPC by clearing CPCCSR0[CPCE].
6. Reconfigure CPC as desired.
7. Enable CPC by setting CPCCSR0[CPCE].

Common operations that require this procedure include changing CPCCSR0[CPCWT], CPCCPSR0[CPCPE], CPCPIR*n*, CPCPAR*n*, and CPCPWR*n*.

8.4.2.4 Disabling the CPC

To disable the CPC once it has been enabled, follow steps 1-5 of the procedure described in Changing the Configuration of an Enabled CPC.

Chapter 9
Prefetch Manager (PMAN)

9.1 Overview

Hardware prefetching can help reduce effective memory latency for cores and IP blocks by bringing memory regions to the CoreNet platform cache before cores and IP blocks access them. PMAN communicates with CPC to identify memory regions that needed to be prefetched and send prefetch requests to CoreNet host domain. The following figure shows the interconnects of PMAN in a CoreNet based system.

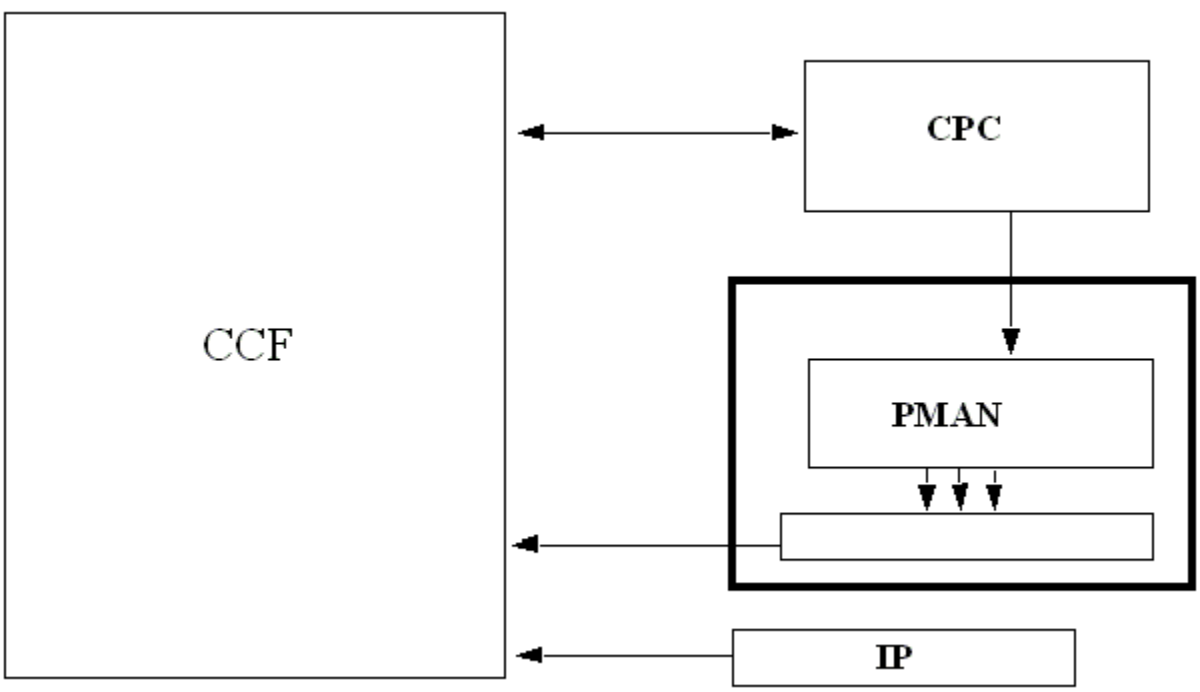


Figure 9-1. PMAN in a System

9.1.1 Features

PMAN contains the following features:

- Support for tracking multiple memory regions for the purpose of prefetching
- Separate clusters of Singleton Buffers and Stream Engine Buffers for core and I/O devices
- Changing region size for tracking
- Defining number of prefetches per bundle
- Exclude tracking based on Requester ID, transaction type, data/instruction type and master (core vs non core) type

9.1.2 Modes of Operation

PMAN operates under two modes::

- PMAN enabled: PMAN is enabled by setting the PE field of PMAN control register 1 to a value of 1.
- PMAN disabled: PMAN is disabled by setting the PE field of PMAN control register 1 to a value of 0. Coming out of reset, PMAN is disabled and software needs to write 1 to the PE field of PMAN control register 1 in order to enable PMAN. After enabling PMAN, disabling PMAN does not stop prefetches from PMAN. It only means PMAN does not track any new transactions from CPC.

9.2 PMAN Memory Map/Register Definition

PMAN registers reside in a 4KB block of CCSR space at 0x00_4000, where X represents 4,5,6. The PMAN Register Definitions are shown below.

PMAN memory map

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
48	PMAN Operation Error Address High register (PMAN_POEAH)	32	R	0000_0000h	9.2.1/351
4C	PMAN Operation Error Address Low register (PMAN_POEAL)	32	R	0000_0000h	9.2.2/351
A0	PMAN Revision register 1 (PMAN_PR1)	32	R	0950_0100h	9.2.3/352
A4	PMAN Revision register 2 (PMAN_PR2)	32	R	0000_0000h	9.2.4/352

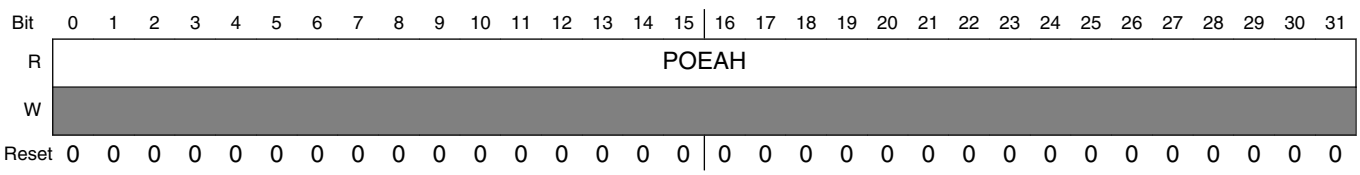
Table continues on the next page...

PMAN memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
C0	PMAN Control register 1 (PMAN_PC1)	32	R/W	0000_0000h	9.2.5/353
C4	PMAN Control register 2 (PMAN_PC2)	32	R/W	2400_0000h	9.2.6/353
C8	PMAN Control register 3 (PMAN_PC3)	32	R/W	00FF_0000h	9.2.7/354
F0	PMAN Interrupt Control and Status register (PMAN_PICS)	32	R/W	0000_0000h	9.2.8/356

9.2.1 PMAN Operation Error Address High register (PMAN_POEAH)

Address: 0h base + 48h offset = 48h

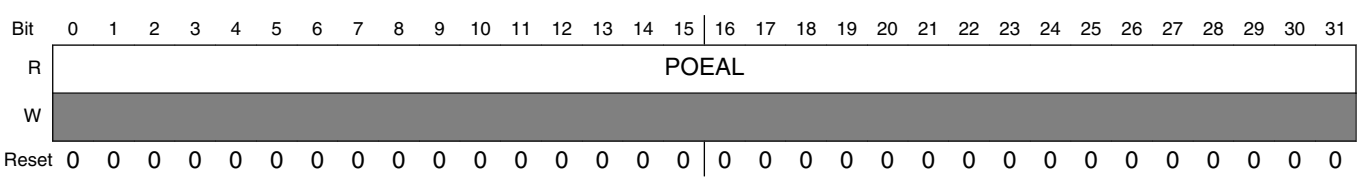


PMAN_POEAH field descriptions

Field	Description
0–31 POEAH	PMAN Operation Error Address High The POEAH is the 32 most-significant address bits that identify the address, in System Memory Space, of the prefetch request that encountered the error condition.

9.2.2 PMAN Operation Error Address Low register (PMAN_POEAL)

Address: 0h base + 4Ch offset = 4Ch



PMAN_POEAL field descriptions

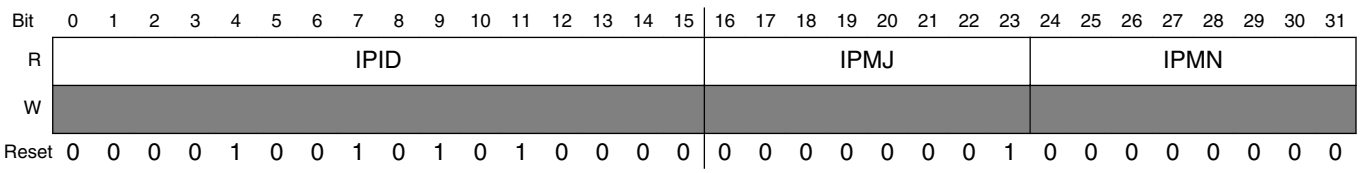
Field	Description
0–31 POEAL	PMAN Operation Error Address Low These bits define the address, in System Memory Space, of the prefetch request that encountered the error condition.

PMAN_POEAL field descriptions (continued)

Field	Description
	The POEAL is the 32 least-significant address bits of the address, in System Memory Space, that encountered the PMAN Error.

9.2.3 PMAN Revision register 1 (PMAN_PR1)

Address: 0h base + A0h offset = A0h

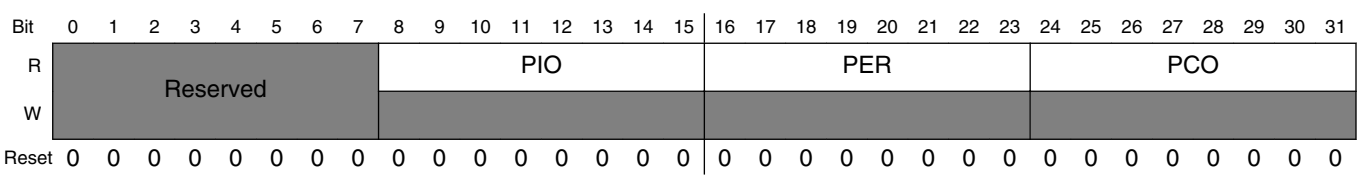


PMAN_PR1 field descriptions

Field	Description
0–15 IPID	PMAN Block ID These bits provide the IP Block ID for PMAN.
16–23 IPMJ	Major Revision These bits provide the Major Revision Number for PMAN.
24–31 IPMN	Minor Revision These bits provide the Minor Revision Number for PMAN.

9.2.4 PMAN Revision register 2 (PMAN_PR2)

Address: 0h base + A4h offset = A4h



PMAN_PR2 field descriptions

Field	Description
0–7 -	This field is reserved. Reserved (Write reserved, Read returns 0)

Table continues on the next page...

PMAN_PR2 field descriptions (continued)

Field	Description
8–15 PIO	PMAN Integration Options These bits provide information on the integration options for PMAN.
16–23 PER	PMAN ECO Revision These bits provide information on the ECO Revision Number for PMAN.
24–31 PCO	PMAN Configuration Options These bits provide information on the configuration options for PMAN.

9.2.5 PMAN Control register 1 (PMAN_PC1)

Address: 0h base + C0h offset = C0h

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

PMAN_PC1 field descriptions

Field	Description
0 PE	PMAN Enable This bit controls whether PMAN is enabled or disabled. This bit is to be set by software after coming out of reset. This bit is encoded as follows: 0 PMAN Disabled. 1 PMAN Enabled.
1–31 -	This field is reserved. Reserved (Write reserved, Read returns 0)

9.2.6 PMAN Control register 2 (PMAN_PC2)

Address: 0h base + C4h offset = C4h

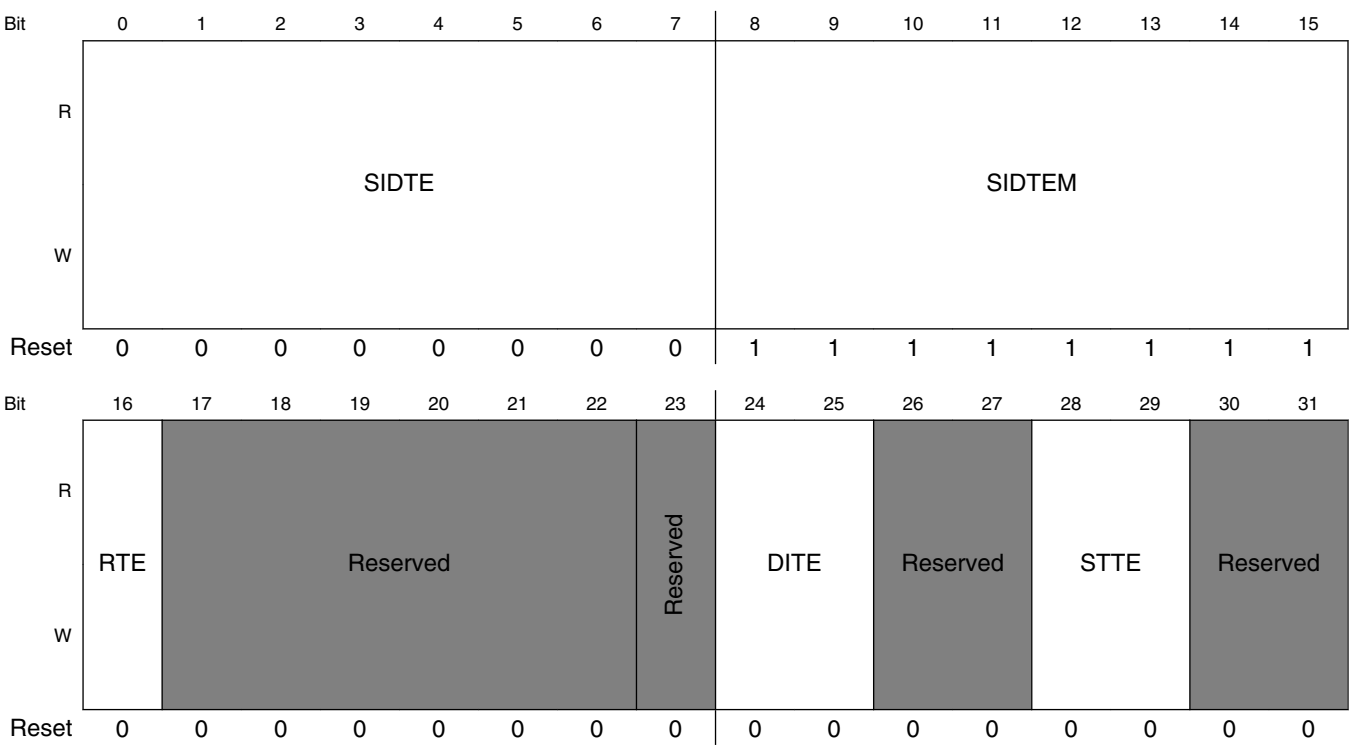
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	PRS				PBS				Reserved																							
Reset	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PMAN_PC2 field descriptions

Field	Description
0–3 PRS	PMAN Region Size These bits control the region of memory PMAN singleton buffer is tracking. These bits are encoded as follows (all other values are reserved): 0000 1KB memory region size 0010 4KB memory region size
4–7 PBS	PMAN Bundle Size These bits control the number of prefetches requested per bundle during HC4 and HC84 states These bits are encoded as follows (all other values are reserved): 0100 4 prefetches per bundle 1000 8 prefetches per bundle
8–31 -	This field is reserved. Reserved (Write reserved, Read returns 0)

9.2.7 PMAN Control register 3 (PMAN_PC3)

Address: 0h base + C8h offset = C8h



PMAN_PC3 field descriptions

Field	Description
0–7 SIDTE	Source ID Tracking Exclusion These bits in conjuction with Source ID tracking exclusion mask (SIDTEMASK) create Source ID/IDs that will be excluded from tracking by PMAN. These bits are encoded as follows: 0x00 Exclude tracking by PMAN for Source ID 0 when SIDTEMASK=0x00. For other SIDTEMASK values appropriate bit/bits will be masked to create Source ID/IDs that will be excluded from tracking. 0x01 Exclude tracking by PMAN for Source ID 1 when SIDTEMASK=0x00. For other SIDTEMASK values appropriate bit/bits will be masked to create Source ID/IDs that will be excluded from tracking. ... 0xFF Exclude tracking by PMAN for Source ID 255 when SIDTEMASK=0x00. For other SIDTEMASK values appropriate bit/bits will be masked to create Source ID/IDs that will be excluded from tracking.
8–15 SIDTEM	Source ID Tracking Exclusion Mask These bits in conjuction with Source ID tracking exclusion (SIDTE) create Source ID/IDs that will be excluded from tracking by PMAN. These bits are encoded as follows: 0x00 All bits of Source ID tracking exclusion (SIDTE) is used to create Source ID that will be excluded from tracking 0x01 Bit 7 of Source ID tracking exclusion (SIDTE) and Source ID of CPC transaction will be masked. All other bits of Source ID tracking exclusion (SIDTE) and Source ID of CPC transaction will be matched to exclude Source ID/IDs from tracking 0x02 Bit 6 of Source ID tracking exclusion (SIDTE) and Source ID of CPC transaction will be masked. All other bits of Source ID tracking exclusion (SIDTE) and Source ID of CPC transaction will be matched to exclude Source ID/IDs from tracking 0x03 Bit 6 and 7 of Source ID tracking exclusion (SIDTE) and Source ID of CPC transaction will be masked. All other bits of Source ID tracking exclusion (SIDTE) and Source ID of CPC transaction will be matched to exclude Source ID/IDs from tracking ... 0xFF No exclusion
16 RTE	Read Type Tracking Exclusion This bit directs PMAN to exclude tracking for Read transaction type This bit is encoded as follows: 0 No Read transaction type tracking exclusion 1 Exclude tracking for Read transaction type
17–22 -	This field is reserved. Reserved (Write reserved, Read returns 0)
23 -	This field is reserved. Reserved (Write reserved, Read returns 0)
24–25 DITE	Data/Instruction type Tracking Exclusion Theses bits direct PMAN to exclude tracking data or instruction transactions from core These bits are encoded as follows (all other values are reserved): 00 No exclusion

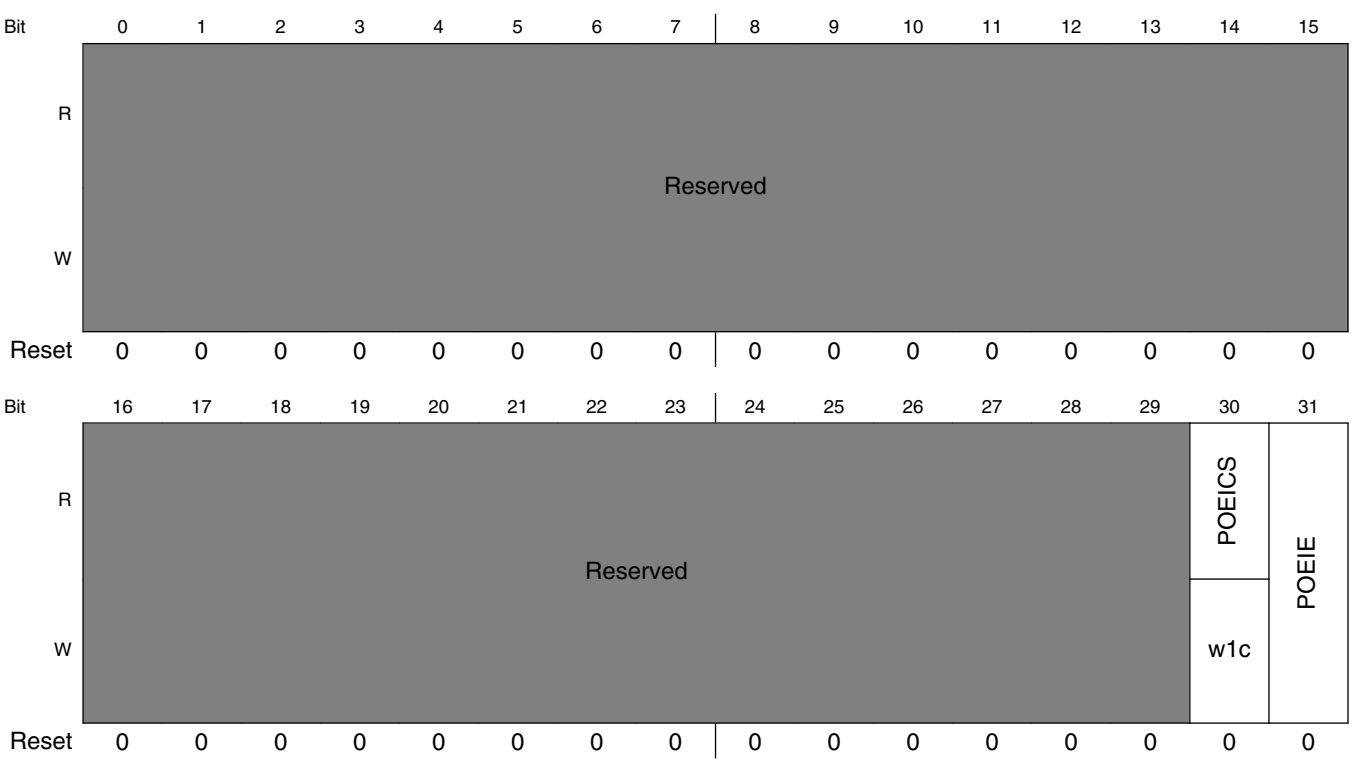
Table continues on the next page...

PMAN_PC3 field descriptions (continued)

Field	Description
	01 Exclude tracking data transactions from core 10 Exclude tracking instruction requests from core
26–27 -	This field is reserved. Reserved (Write reserved, Read returns 0)
28–29 STTE	Source Type Tracking Exclusion Theses bits direct PMAN to exclude tracking core or non core transactions These bits are encoded as follows (all other values are reserved): 00 No exclusion 01 Exclude tracking core transactions 10 Exclude tracking non core transactions
30–31 -	This field is reserved. Reserved (Write reserved, Read returns 0)

9.2.8 PMAN Interrupt Control and Status register (PMAN_PICS)

Address: 0h base + F0h offset = F0h



PMAN_PICS field descriptions

Field	Description
0–29 -	This field is reserved. Reserved (Write reserved, Read returns 0)
30 POEICS	PMAN Operation Error Interrupt Control and Status These bits provides status on whether the operation error interrupt pin is asserted and controls whether the operation error interrupt pin is to be de-asserted. The bit is coded as follows: 0 Operation error interrupt pin is de-asserted 1 Operation error interrupt pin is asserted. software must write 1 in order to clear the pin.
31 POEIE	PMAN Operation Error Interrupt Enable These bits identify if an operation error interrupt is to be asserted. The bits are coded as follows: 0 No interrupt asserted if an operation error is detected 1 Operation error interrupt pin asserted if an operation error is detected

9.3 Functional Description

The following is a brief description of the setup requirements and sequence of events involving performing prefetches by PMAN.

9.3.1 Detailed description of PMAN actions

This section describes in detail the operations performed by PMAN for performing prefetches.

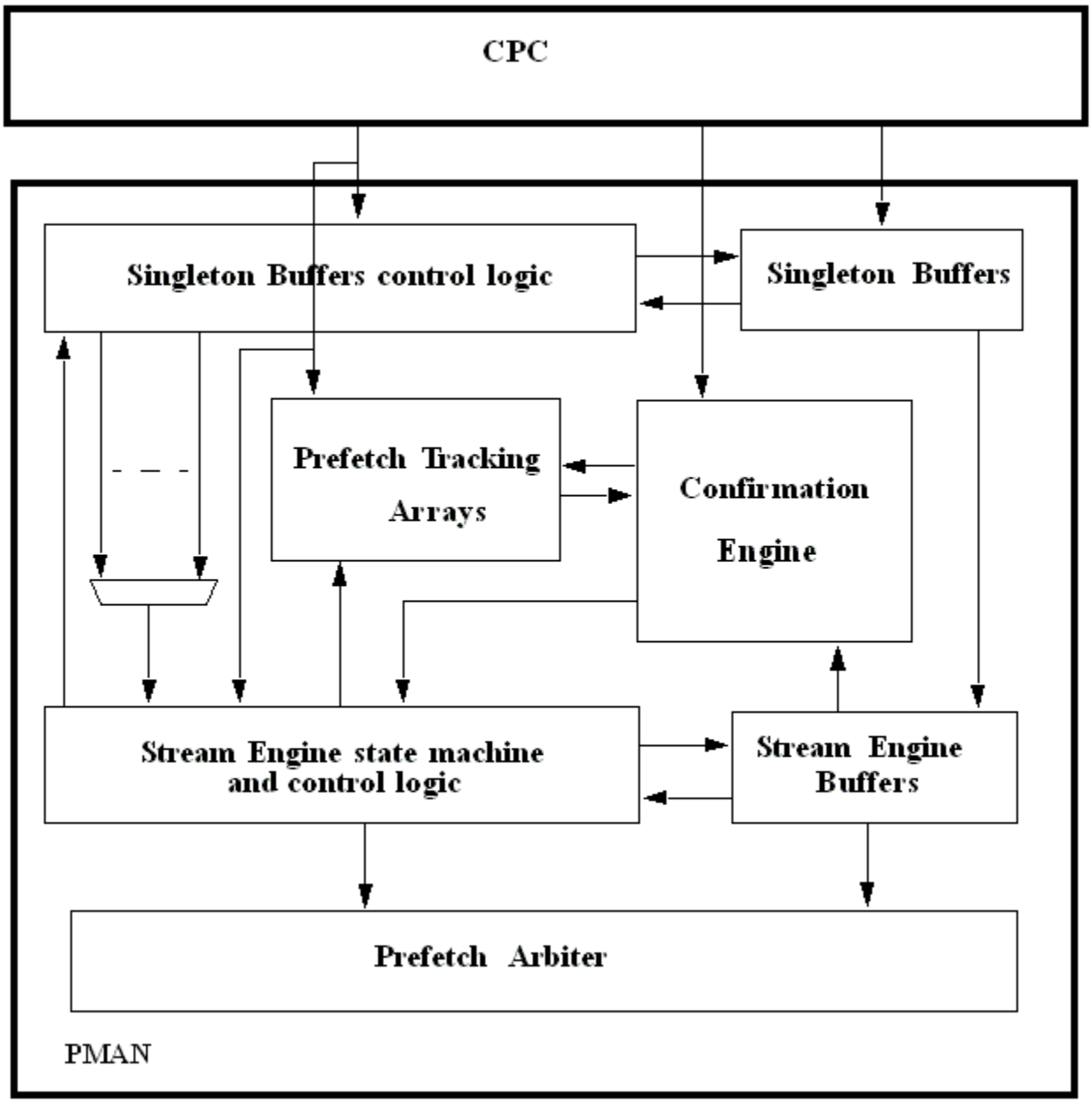


Figure 9-10. PMan Block diagram

9.3.1.1 Singleton Buffers

The singleton buffers purpose is to track a 4KB region of memory to predict a stride and thus promote the stream of accesses eligible for prefetching. The singleton buffers are a small set associative arrays with address split into a tag portion and an offset portion. Singleton buffers also has a stored stride, a valid bit and a confidence counter. Singleton buffers ignore non-prefetchable accesses such as inhibited reads, write or RWITM to CPC.

9.3.1.1.1 Allocation policy

Singleton Buffers are allocated under following condition:

- A CPC miss that does not match any of the existing singleton buffers and the operation type is either a Read or a Stash from a core or I/O. No allocation will occur on a cache inhibited Read from a core, nor on a Write or Store type operation from a core or I/O.

On allocation, the address is recorded, the stride is set to 0, the confidence counter is initialized to 0 and the valid bit is set.

9.3.1.1.2 Promotion policy

The following policy is used to promote a buffer from Singleton Buffers to Stream Engines:

- If a CPC miss is to a consecutive address (next CG or previous CG for negative sequential) of an existing Singleton Buffer
- If a CPC miss is not to a consecutive address to an existing Singleton Buffer then PMAN remembers the stride between the two addresses (stride is the distance between two addresses, it can be in any direction, positive or negative). The next CPC miss to that region is the third access to that region, since stride value is non zero after second access. If the next CPC miss follows the stored stride value than the buffer is promoted to Stream Engines.

9.3.1.1.3 Deallocation policy

The following policy is used to deallocate buffers from Singleton Buffers:

- If a third CPC miss to a region does not follow the remembered stride, the singleton buffer is deallocated.
- While waiting for Stream Engine, if a consecutive address access receives an access to a non-consecutive address.
- While waiting for Stream Engine, if a strided address access receives an access does not match the stride.
- If a prefetch stream reaches the 4k boundary, the Singleton Buffer will stop promoting the stream to stream engine and will be deallocated.
- If the confidence counter for a singleton buffer is degraded down to zero, the singleton buffer is deallocated.

9.3.1.1.4 Stride

Stride is updated under the following condition:

- On allocation of CPC miss to Singleton Buffers, the stride is set to 0.
- On the second CPC miss to a region of an existing Singleton Buffer, PMAN calculates and stores the stride.

9.3.1.1.5 Confidence counter

Confidence counter is updated under the following condition:

- On allocation to Singleton Buffers, the confidence counter is set to 1.
- If no stream engine is currently free, the PMAN will increase the confidence counter of the buffer for more consecutive accesses or more accesses based on stride of the region.
- On a CPC miss that replaces an existing singleton buffer via PLRU, the confidence counters on all outstanding singleton buffers are also decremented.

9.3.1.1.6 Arbitration

Each cycle one singleton buffer for a given core or non core can be promoted to stream engine buffers for that core or non core. A round robin algorithm is used to promote a buffer from Singleton Buffers to Stream Engine.

9.3.1.2 Stream Engine

The Stream Engine handles initiation, confirmation, and state tracking for a single stream that has already been identified by the singleton buffer as prefetchable. The goal of the stream engine is to generate high-accuracy prefetches, far enough in advance of the demand stream. Stream engine buffers contain an address, a valid bit, stride and a state machine.

9.3.1.2.1 Stream Engine Allocation policy

The following policy is used to allocate a buffer from Singleton Buffers to Stream Engine buffers:

- If a CPC miss is to a consecutive address (next CG) of an existing Singleton Buffer
- The third CPC miss to a region with a match to a strided value.

On allocation, Singleton Buffer will provide the address and the value of stride. A valid bit will be set and the state machine will be initialized to Low Confidence Prefetch1 State.

9.3.1.2.2 State Machine

The state machine consists of following states:

- Low Confidence Prefetch1 State (LC1): On allocation from Singleton Buffer, Stream Engine buffer is initialized to LC1 state if SB confidence counter value is equal or less than 4. In this state, a single prefetch is initiated from PMAN. The address is calculated based on next consecutive address or strided address.After sending the prefetch, PTA will be set for the corresponding bit. After the prefetch has consumed, state macine will change the state to MC2.
- Medium Confidence Prefetch2 State (MC2): On allocation from Singleton Buffer, Stream Engine buffer may be initialized to MC2 state if SB confidence counter value is between 5 or 6 . In this state, two prefetches should be initiated for the next appropriate addresses. PTA should be set for the corresponding bits. After the prefetches have consumed, state macine will change the state to HC4.
- High Confidence Prefetch 4 State (HC4): On allocation from Singleton Buffer, Stream Engine buffer may be initialized to HC4 state if SB confidence counter value is between 7 or 10 . In this state, four prefetches (or one bundle) should be initiated for the next appropriate addresses. PTA should be set for the corresponding bits. After the prefetches have consumed, state macine will change the state to HC84.
- High Confidence Prefetch8 State (HC84): On allocation from Singleton Buffer, Stream Engine buffer may be initialized to HC84 state if SB confidence counter value is greater than or equal to 11 . In this state, PMAN can launch minimum 8 prefetches. PMAN uses a distance counter to launch 4 prefetches at a time. When this state is entered, PMAN sends 4 prefetches and sets corresponding PTA bits and distance counter is set to a value of 8. For each consumption of a prefetch, the distance counter is decremented. If the Distance Counter reaches a threshold value of 4 or less, the next prefetch bundle for 4 prefetches is initiated and distance counter is set to 8 and PTA corresponding bits are set. PMAN will continue to send 8 prefetches at a time until the stream reaches 4K boundary or the stream is no longer consumed.

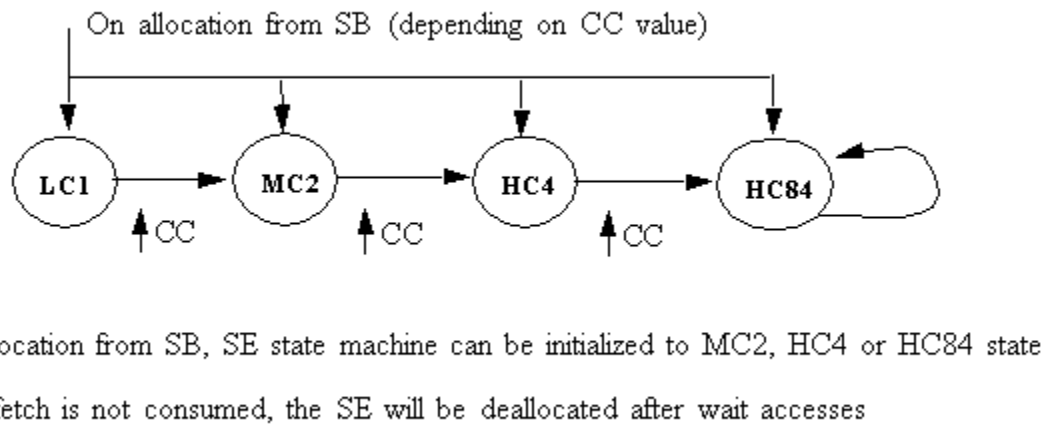


Figure 9-11. Stream Engine State machine

9.3.1.2.3 Stream Engine Deallocation policy

Deallocation policies for Stream Engine Buffers:

- If the prefetch is not observed to be consumed within a certain number of accesses, the demand stream is considered to be done. At this point, the stream engine will be deallocated.
- If a prefetch stream reaches the 4k boundary, the stream engine will stop issueing any new prefetch and will be deallocated.

9.3.1.3 Prefetch Tracking Arrays

Each Stream engine buffer has a corresponding prefetch tracking array (PTA). The PTA is cleared when Singleton Buffer allocates an entry to Stream Engine Buffers. When a Stream engine sends a prefetch, the bit corresponding to the CG are set in PTA. PTA clears the bit when it reaceives confirmation of demand accesses for that CG.

9.3.1.4 Request Arbiter

The Request Arbiter is used to service the prefetch requests generated by Stream Engine. A two-stage round-robin arbitration is used to send outstanding prefetch requests. The arbitration is also sticky in nature. Once a core/non-core wins arbitration, arbitration will allow it to send all its prefetches before moving to next core/non-core.

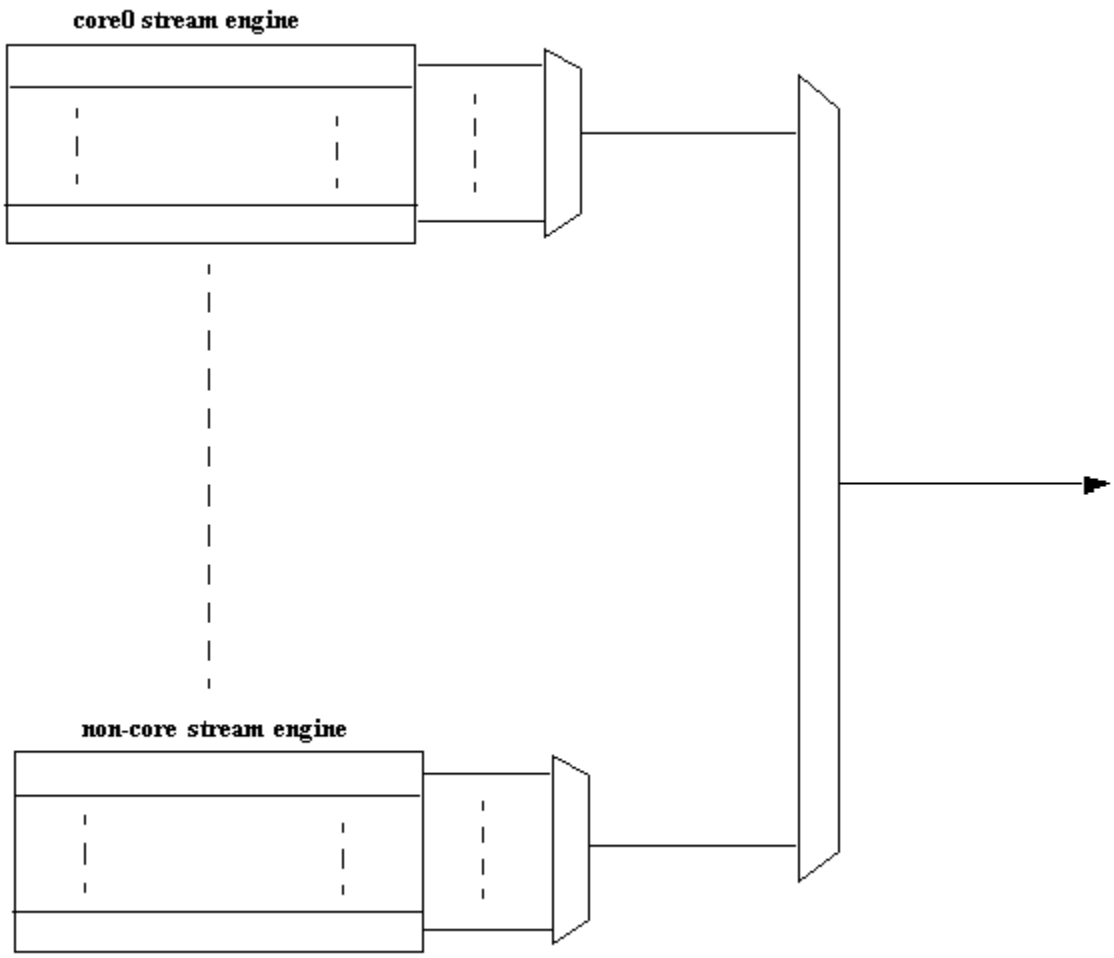


Figure 9-12. PMAN arbitration

9.4 Initialization/Application Information

This section describes system/software configuration and setup of the data structures and PMAN for system implementations. This section also describes system/software interactions of the data structures and PMAN during run-time operation.

9.4.1 System setup for PMAN

9.4.1.1 Setting up PMAN

Software must program PMAN memory map registers to initiate prefetches from PMAN.

- PMAN control registers are programmed with necessary setup. Otherwise PMAN will be run with default setup.
- PMAN Enable bit in PMAN CCSR space (PMAN control register1) must be set in order to enable PMAN to perform prefetches.

9.4.1.2 Power On Reset

Coming out of reset, PMAN must be enabled by setting PE bit in PMAN control register1. Default values will be used if no other configuration register is modified.

Chapter 10
CoreNet Coherency Fabric (CCF)

10.1 CCF Introduction

This chapter provides an overview of the CoreNet coherency fabric (CCF). It describes the components of the CCF and how they are interconnected. It also describes the salient features of the platform and its components.

The CoreNet coherency fabric (CCF) is a fabric-oriented, connectivity infrastructure that enables the implementation of coherent, multicore systems. The CCF acts as a central interconnect for cores, platform-level caches, memory subsystems, peripheral devices, and I/O host bridges in the system. The CCF natively supports Power Architecture® coherency semantics.

The figure below shows organization of a multicore system designed around the CCF.

10.1.1 CCF Features Summary

The CCF includes the following distinctive features:

- Multiple in-flight transactions with:
  - Concurrency of transactional progress through the system
  - Out of order completion
- Retry-free operation
  - Zero-loss utilization of system transaction processing bandwidth
- Sustainable bandwidth: 4 transactions/cycle
- Low latency data path for platform cache data
- Sustainable read bandwidth: 128 bytes (or two coherence granules - CGs) per cycle
- Power Architecture coherency semantics
  - Accelerated operation for non-coherent accesses
- 64 byte coherency granules
  - 1-8 bytes within an aligned double word, and aligned 16-byte, 32-byte, and 64-byte single copy atomic access semantics

- Rich transaction types and semantics to support:
  - All Power Architecture storage and synchronization operations
  - Inter-processor low-latency message send operations
  - RapidIO atomic operations
  - Reads and writes with semantic extensions for in-place atomic update of storage
- Transaction ordering support
  - Native support for Power Architecture ordering semantics
  - On-demand strong ordering for all "cache-inhibited" and "guarded" accesses from processors
  - Non-Power Architecture ordering support through host bridges
- Address map support
  - 32 local access windows (LAWs)
  - Boot window and CCSR spaces
  - DDR memory and SRAM interleaving support
- Logical partitioning support
  - Address-based, secure isolation of partitions and their resources
  - Coherency subdomain assignment and snoop limiting per LAW
  - Inter-partition sharing of address ranges
  - Peripheral access management unit (PAMU)
    - Capability to assign peripheral devices to logical partitions and limiting direct storage accesses (DSA) to owner partitions
    - Support for virtualized peripheral devices serving multiple partitions
    - Multiple windows per device
    - Window translation for accesses by devices
    - Transparent cache-coherent operation with PAMU tables in main memory
    - Transaction translation capability
    - Error isolation between partitions
- One high performance memory complex
  - Non-inclusive, in-line platform cache
    - Write-back semantics
    - Read latency optimization by keeping track of owner processors
    - I/O stashing
    - Line and way locking
    - Data delivery capability of one CG per cycle
    - Semantic extension support for cache-based atomic update of data
- Support for stashing
  - Processor cache stashing
  - Address- and attribute-based I/O stashing in platform cache
- Debug support
  - Source tracking from source to destination

- Access triggering, filtering, and tracing
- Event counts for performance monitoring

10.2 CoreNet Coherency Fabric (CCF) Memory Map

CCF memory map

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1_8200	Snoop ID 0 Port Mapping Register (CCF_SIDMR0)	32	R	FF00_8000h	10.2.1/369
1_8204	Snoop ID n Port Mapping Register (CCF_SIDMR1)	32	R/W	0000_0000h	10.2.2/370
1_8208	Snoop ID n Port Mapping Register (CCF_SIDMR2)	32	R/W	0000_0000h	10.2.2/370
1_820C	Snoop ID n Port Mapping Register (CCF_SIDMR3)	32	R/W	0000_0000h	10.2.2/370
1_8210	Snoop ID n Port Mapping Register (CCF_SIDMR4)	32	R/W	0000_0000h	10.2.2/370
1_8214	Snoop ID n Port Mapping Register (CCF_SIDMR5)	32	R/W	0000_0000h	10.2.2/370
1_8218	Snoop ID n Port Mapping Register (CCF_SIDMR6)	32	R/W	0000_0000h	10.2.2/370
1_821C	Snoop ID n Port Mapping Register (CCF_SIDMR7)	32	R/W	0000_0000h	10.2.2/370
1_8220	Snoop ID n Port Mapping Register (CCF_SIDMR8)	32	R/W	0000_0000h	10.2.2/370
1_8224	Snoop ID n Port Mapping Register (CCF_SIDMR9)	32	R/W	0000_0000h	10.2.2/370
1_8228	Snoop ID n Port Mapping Register (CCF_SIDMR10)	32	R/W	0000_0000h	10.2.2/370
1_822C	Snoop ID n Port Mapping Register (CCF_SIDMR11)	32	R/W	0000_0000h	10.2.2/370
1_8230	Snoop ID n Port Mapping Register (CCF_SIDMR12)	32	R/W	0000_0000h	10.2.2/370
1_8234	Snoop ID n Port Mapping Register (CCF_SIDMR13)	32	R/W	0000_0000h	10.2.2/370
1_8238	Snoop ID n Port Mapping Register (CCF_SIDMR14)	32	R/W	0000_0000h	10.2.2/370
1_823C	Snoop ID n Port Mapping Register (CCF_SIDMR15)	32	R/W	0000_0000h	10.2.2/370
1_8240	Snoop ID n Port Mapping Register (CCF_SIDMR16)	32	R/W	0000_0000h	10.2.2/370
1_8244	Snoop ID n Port Mapping Register (CCF_SIDMR17)	32	R/W	0000_0000h	10.2.2/370
1_8248	Snoop ID n Port Mapping Register (CCF_SIDMR18)	32	R/W	0000_0000h	10.2.2/370
1_824C	Snoop ID n Port Mapping Register (CCF_SIDMR19)	32	R/W	0000_0000h	10.2.2/370
1_8250	Snoop ID n Port Mapping Register (CCF_SIDMR20)	32	R/W	0000_0000h	10.2.2/370
1_8254	Snoop ID n Port Mapping Register (CCF_SIDMR21)	32	R/W	0000_0000h	10.2.2/370
1_8258	Snoop ID n Port Mapping Register (CCF_SIDMR22)	32	R/W	0000_0000h	10.2.2/370
1_825C	Snoop ID n Port Mapping Register (CCF_SIDMR23)	32	R/W	0000_0000h	10.2.2/370
1_8260	Snoop ID n Port Mapping Register (CCF_SIDMR24)	32	R/W	0000_0000h	10.2.2/370
1_8264	Snoop ID n Port Mapping Register (CCF_SIDMR25)	32	R/W	0000_0000h	10.2.2/370
1_8268	Snoop ID n Port Mapping Register (CCF_SIDMR26)	32	R/W	0000_0000h	10.2.2/370
1_826C	Snoop ID n Port Mapping Register (CCF_SIDMR27)	32	R/W	0000_0000h	10.2.2/370
1_8270	Snoop ID n Port Mapping Register (CCF_SIDMR28)	32	R/W	0000_0000h	10.2.2/370
1_8274	Snoop ID n Port Mapping Register (CCF_SIDMR29)	32	R/W	0000_0000h	10.2.2/370
1_8278	Snoop ID n Port Mapping Register (CCF_SIDMR30)	32	R/W	0000_0000h	10.2.2/370

Table continues on the next page...

CCF memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1_827C	Snoop ID n Port Mapping Register (CCF_SIDMR31)	32	R/W	0000_0000h	10.2.2/370
1_8600	CSD ID 0 Port Mapping Register (CCF_CIDMR0)	32	R	FF00_8000h	10.2.3/371
1_8604	CSDID n Port Mapping Register (CCF_CIDMR1)	32	R/W	0000_0000h	10.2.4/372
1_8608	CSDID n Port Mapping Register (CCF_CIDMR2)	32	R/W	0000_0000h	10.2.4/372
1_860C	CSDID n Port Mapping Register (CCF_CIDMR3)	32	R/W	0000_0000h	10.2.4/372
1_8610	CSDID n Port Mapping Register (CCF_CIDMR4)	32	R/W	0000_0000h	10.2.4/372
1_8614	CSDID n Port Mapping Register (CCF_CIDMR5)	32	R/W	0000_0000h	10.2.4/372
1_8618	CSDID n Port Mapping Register (CCF_CIDMR6)	32	R/W	0000_0000h	10.2.4/372
1_861C	CSDID n Port Mapping Register (CCF_CIDMR7)	32	R/W	0000_0000h	10.2.4/372
1_8620	CSDID n Port Mapping Register (CCF_CIDMR8)	32	R/W	0000_0000h	10.2.4/372
1_8624	CSDID n Port Mapping Register (CCF_CIDMR9)	32	R/W	0000_0000h	10.2.4/372
1_8628	CSDID n Port Mapping Register (CCF_CIDMR10)	32	R/W	0000_0000h	10.2.4/372
1_862C	CSDID n Port Mapping Register (CCF_CIDMR11)	32	R/W	0000_0000h	10.2.4/372
1_8630	CSDID n Port Mapping Register (CCF_CIDMR12)	32	R/W	0000_0000h	10.2.4/372
1_8634	CSDID n Port Mapping Register (CCF_CIDMR13)	32	R/W	0000_0000h	10.2.4/372
1_8638	CSDID n Port Mapping Register (CCF_CIDMR14)	32	R/W	0000_0000h	10.2.4/372
1_863C	CSDID n Port Mapping Register (CCF_CIDMR15)	32	R/W	0000_0000h	10.2.4/372
1_8640	CSDID n Port Mapping Register (CCF_CIDMR16)	32	R/W	0000_0000h	10.2.4/372
1_8644	CSDID n Port Mapping Register (CCF_CIDMR17)	32	R/W	0000_0000h	10.2.4/372
1_8648	CSDID n Port Mapping Register (CCF_CIDMR18)	32	R/W	0000_0000h	10.2.4/372
1_864C	CSDID n Port Mapping Register (CCF_CIDMR19)	32	R/W	0000_0000h	10.2.4/372
1_8650	CSDID n Port Mapping Register (CCF_CIDMR20)	32	R/W	0000_0000h	10.2.4/372
1_8654	CSDID n Port Mapping Register (CCF_CIDMR21)	32	R/W	0000_0000h	10.2.4/372
1_8658	CSDID n Port Mapping Register (CCF_CIDMR22)	32	R/W	0000_0000h	10.2.4/372
1_865C	CSDID n Port Mapping Register (CCF_CIDMR23)	32	R/W	0000_0000h	10.2.4/372
1_8660	CSDID n Port Mapping Register (CCF_CIDMR24)	32	R/W	0000_0000h	10.2.4/372
1_8664	CSDID n Port Mapping Register (CCF_CIDMR25)	32	R/W	0000_0000h	10.2.4/372
1_8668	CSDID n Port Mapping Register (CCF_CIDMR26)	32	R/W	0000_0000h	10.2.4/372
1_866C	CSDID n Port Mapping Register (CCF_CIDMR27)	32	R/W	0000_0000h	10.2.4/372
1_8670	CSDID n Port Mapping Register (CCF_CIDMR28)	32	R/W	0000_0000h	10.2.4/372
1_8674	CSDID n Port Mapping Register (CCF_CIDMR29)	32	R/W	0000_0000h	10.2.4/372
1_8678	CSDID n Port Mapping Register (CCF_CIDMR30)	32	R/W	0000_0000h	10.2.4/372
1_867C	CSDID n Port Mapping Register (CCF_CIDMR31)	32	R/W	0000_0000h	10.2.4/372
1_8E40	CCF Error Detect Register (CCF_ERRDET)	32	w1c	0000_0000h	10.2.5/373
1_8E44	CCF Error Disable Register (CCF_ERRDIS)	32	R/W	0000_0000h	10.2.6/375
1_8E48	CCF Error Interrupt Enable Register (CCF_ERRINTEN)	32	R/W	0000_0000h	10.2.7/376
1_8E4C	CCF Error Capture Attribute Register (CCF_CECAR)	32	w1c	0000_0000h	10.2.8/376

Table continues on the next page...

CCF memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
1_8E50	CCF Error Capture Address Register High (CCF_CECADDRH)	32	w1c	0000_0000h	10.2.9/377
1_8E54	CCF Error Capture Address Register Low (CCF_CECADDRL)	32	w1c	0000_0000h	10.2.10/377
1_8E58	CCF Error Capture Attribute Register 2 (CCF_CECAR2)	32	R/W	0000_0000h	10.2.11/378

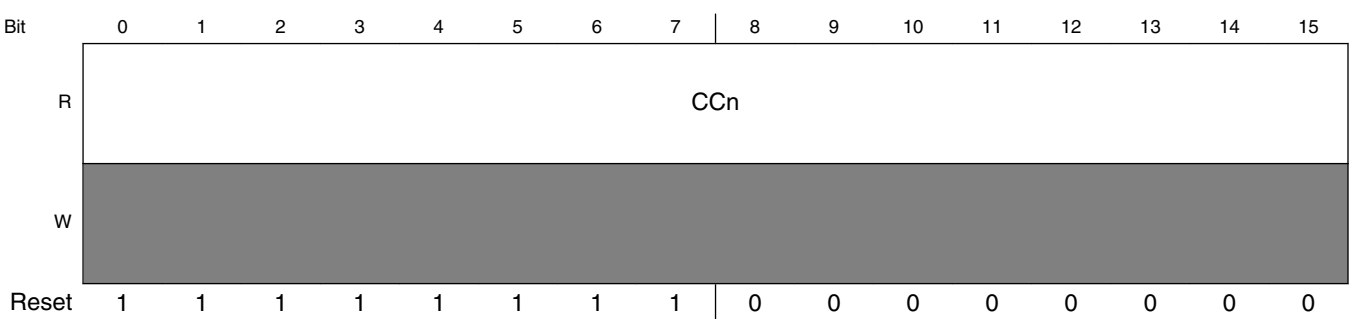
10.2.1 Snoop ID 0 Port Mapping Register (CCF_SIDMR0)

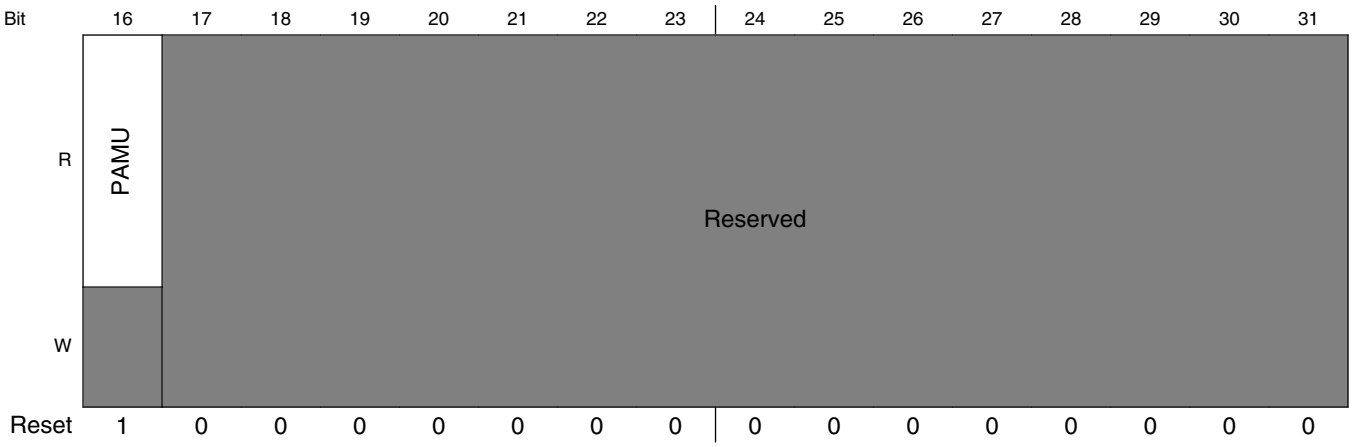
The Snoop ID n Port Mapping Registers provide a CoreNet Snoop ID to core cluster and PAMU mapping function. CoreNet transactions include an 8-bit Snoop ID [0:7]. A Snoop ID of 00h indicates an inactive Snoop ID and maps to the default SIDMR0. SIDMR0 is a read-only register. The reset value of SIDMR0 is coherent, but considered lower performing. A non-zero Snoop ID indicates a lookup into the snoop ID port mapping registers (SIDMR1-31). The Snoop ID acts as an index into these registers and extracts a core cluster and PAMU field indicating the active snoopers for this transaction. An active Snoop ID (non-zero) overrides all other implementation specific determinations for snoopers.

NOTE

Although there are 256 possible Snoop IDs, the CCF only supports bits [3:7] and thus implements 32 snoop ID port mapping registers. If any of the upper three bits (Snoop ID[0:2]) are set, the Snoop ID will alias to the 32 SIDMRs defined by Snoop ID[3:7]. If the lower bits [3:7] = 0b00000 then the default SIDMR0 is used.

Address: 1_8000h base + 200h offset = 1_8200h





CCF_SIDMR0 field descriptions

Field	Description
0–15 CCn	Core Cluster n. When set, a bit indicates which corresponding active core cluster to snoop for this transaction. Bit 0 corresponds to core cluster 0.
16 PAMU	Snoop PAMUs.When set, the PAMU field indicates that all PAMUs within the SoC need to be snooped for this transaction 0 Do not snoop the PAMUs. 1 Snoop all PAMUs.
17–31 -	This field is reserved. Reserved

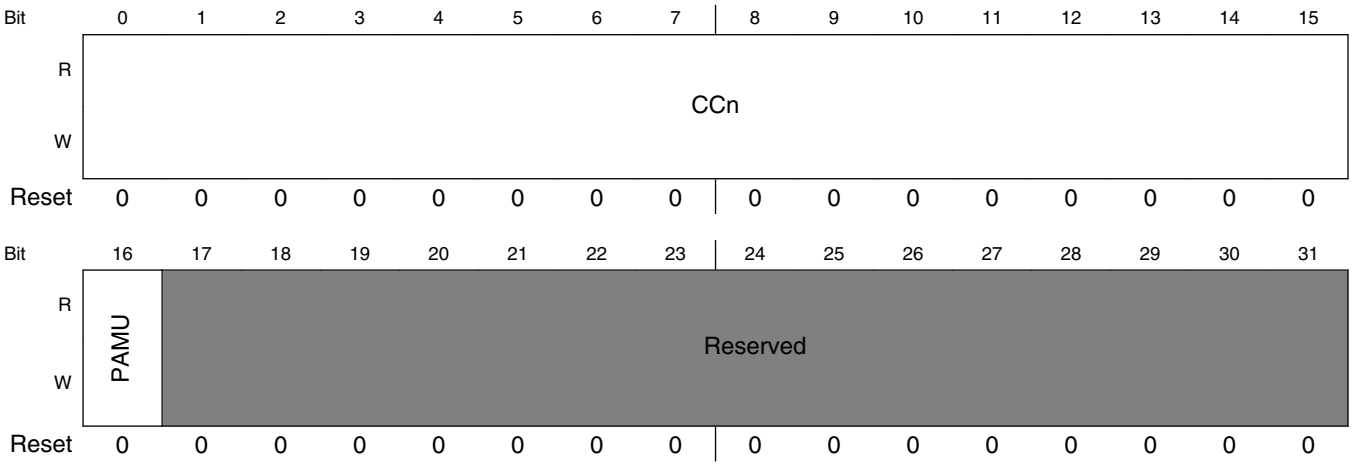
10.2.2 Snoop ID n Port Mapping Register (CCF_SIDMR*n*)

The Snoop ID n Port Mapping Registers provide a CoreNet Snoop ID to core cluster and PAMU mapping function. CoreNet transactions include an 8-bit Snoop ID [0:7]. A Snoop ID of 00h indicates an inactive Snoop ID and maps to the default SIDMR0. SIDMR0 is a read-only register. The reset value of SIDMR0 is coherent, but considered lower performing. A non-zero Snoop ID indicates a lookup into the snoop ID port mapping registers (SIDMR1-31). The Snoop ID acts as an index into these registers and extracts a core cluster and PAMU field indicating the active snoopers for this transaction. An active Snoop ID (non-zero) overrides all other implementation specific determinations for snoopers.

NOTE

Although there are 256 possible Snoop IDs, the CCF only supports bits [3:7] and thus implements 32 snoop ID port mapping registers. If any of the upper three bits (Snoop ID[0:2]) are set, the Snoop ID will alias to the 32 SIDMRs defined by Snoop ID[3:7]. If the lower bits [3:7] = 0b00000 then the default SIDMR0 is used.

Address: 1_8000h base + 204h offset + (4d × i), where i=0d to 30d



CCF_SIDMR*n* field descriptions

Field	Description
0–15 CCn	Core Cluster n. When set, a bit indicates which corresponding active core cluster to snoop for this transaction. Bit 0 corresponds to core cluster 0.
16 PAMU	Snoop PAMUs.When set, the PAMU field indicates that all PAMUs within the SoC need to be snooped for this transaction 0 Do not snoop the PAMUs. 1 Snoop all PAMUs.
17–31 -	This field is reserved. Reserved

10.2.3 CSD ID 0 Port Mapping Register (CCF_CIDMR0)

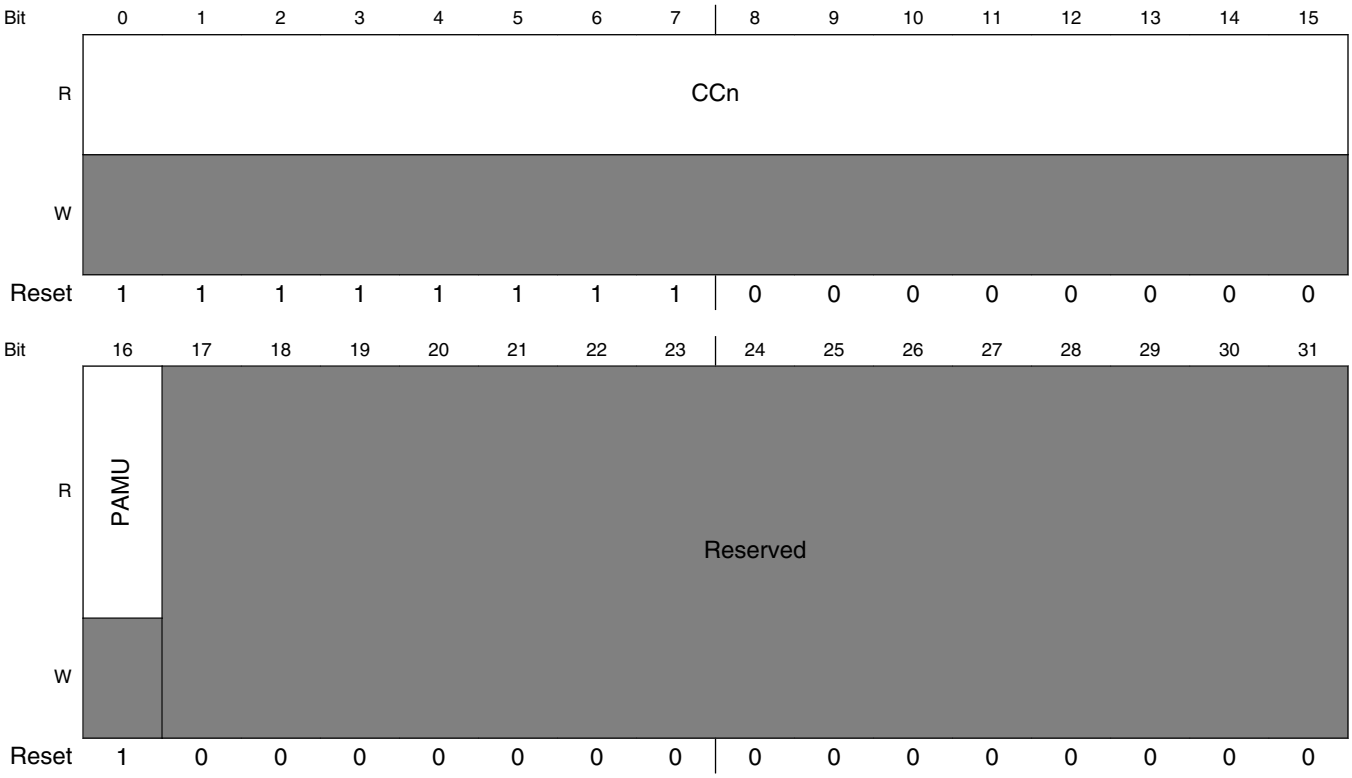
The Coherency Subdomain ID n Port Mapping Registers provide a CoreNet Coherency Subdomain ID to core cluster and PAMU mapping function. The LAW attribute registers include an 8-bit coherency subdomain ID field (LAWAR*n*[CSD_ID]). If the transaction Snoop ID is 00h, the CSD_ID obtained from the LAW hit is used to determine the valid snoopers for the transaction. The CSD_ID acts as an index into the CIDMR0-CIDMR31 registers and extracts a core cluster n field [0:15] and a PAMU field indicating the active snoopers for this transaction.

CIDMR00 is a read only register. The reset value of CIDMR0 is coherent, but considered lower performing.

NOTE

Although there are 256 possible CSDIDs, the CCF only supports 32 CIDMRs by using CSD_ID[3:7]; CSD_ID[0:2] are assumed to be inactive/zero.

Address: 1_8000h base + 600h offset = 1_8600h



CCF_CIDMR0 field descriptions

Field	Description
0–15 CCn	Core Cluster n. When set, a bit indicates which corresponding active core cluster to snoop for this transaction. Bit 0 corresponds to core cluster 0.
16 PAMU	Snoop PAMUs.When set, the PAMU field indicates that all PAMUs within the SoC need to be snooped for this transaction 0 Do not snoop the PAMUs. 1 Snoop all PAMUs.
17–31 -	This field is reserved. Reserved

10.2.4 CSDID n Port Mapping Register (CCF_CIDMR*n*)

The Coherency Subdomain ID n Port Mapping Registers provide a CoreNet Coherency Subdomain ID to core cluster and PAMU mapping function. The LAW attribute registers include an 8-bit coherency subdomain ID field (LAWAR*n*[CSD_ID]). If the transaction Snoop ID is 00h, the CSD_ID obtained from the LAW hit is used to determine the valid snoopers for the transaction. The CSD_ID acts as an index into the CIDMR0-CIDMR31 registers and extracts a core cluster n field [0:15] and a PAMU field indicating the active snoopers for this transaction.

NOTE

Although there are 256 possible CSDIDs, the CCF only supports 32 CIDMRs by using CSD_ID[3:7]; CSD_ID[0:2] are assumed to be inactive/zero.

Address: 1_8000h base + 604h offset + (4d × i), where i=0d to 30d

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

CCF_CIDMR*n* field descriptions

Field	Description
0–15 CCn	Core Cluster n. When set, a bit indicates which corresponding active core cluster to snoop for this transaction. Bit 0 corresponds to core cluster 0.
16 PAMU	Snoop PAMUs.When set, the PAMU field indicates that all PAMUs within the SoC need to be snooped for this transaction 0 Do not snoop the PAMUs. 1 Snoop all PAMUs.
17–31 -	This field is reserved. Reserved

10.2.5 CCF Error Detect Register (CCF_ERRDET)

Address: 1_8000h base + E40h offset = 1_8E40h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CAP	CTYPE					Reserved									
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

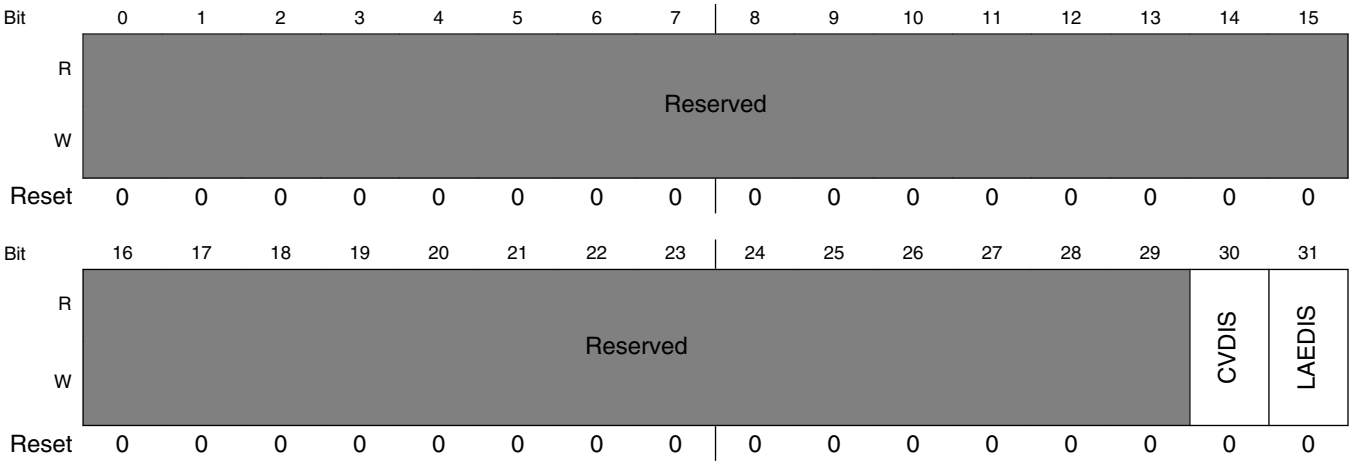
Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved														CV	LAE
W															w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CCF_ERRDET field descriptions

Field	Description
0 CAP	Capture valid. **NOTE:** To avoid capture/detect race conditions, it is recommended that software clear CAP and the detect bit for the type of error that was captured using a single write operation. 0 The error capture registers do not contain valid error attributes. The error capture registers are armed; the next detected error will be captured and this bit will be set automatically. 1 The values in the error capture registers are valid. No new errors will be captured until software re-arms capture by writing a 1 to clear this bit.
1–5 CTYPE	Captured error type. Indicates the type of error that was captured. This value is only valid when CCF_ERRDET[CAP]=1. Cleared by writing a 1 to CCF_ERRDET[CAP]. The CTYPE value encodes to the corresponding bit position. For example, 0h=local access error, 1h=coherency violation, all other encodings reserved.
6–29 -	This field is reserved. Reserved
30 CV	Coherency violation. This indicates the state of the coherency granule was found to be in violation of the coherency protocol.
31 LAE	Local Access Error Cases that can generate an LAE • Local Access Window Miss. An incoming transaction misses all LAWs (except when the PBL is active). • Unavailable target ID programmed in LAW attribute register. See LAWn attribute register (LAW_LAWAR*n*) , for more information. **NOTE:** If CV and LAE occur simultaneously then the LAE Error takes precedence. **NOTE:** If CV (Bit 30) or LAE (31) was previously set and an LAE occurs it will be ignored and bit 31 will not be set. **NOTE:** If a w1c occurs at the same time as the LAE the w1c takes precedence. 0 Local Access Error has not occurred 1 Local Access Error has occurred

10.2.6 CCF Error Disable Register (CCF_ERRDIS)

Address: 1_8000h base + E44h offset = 1_8E44h

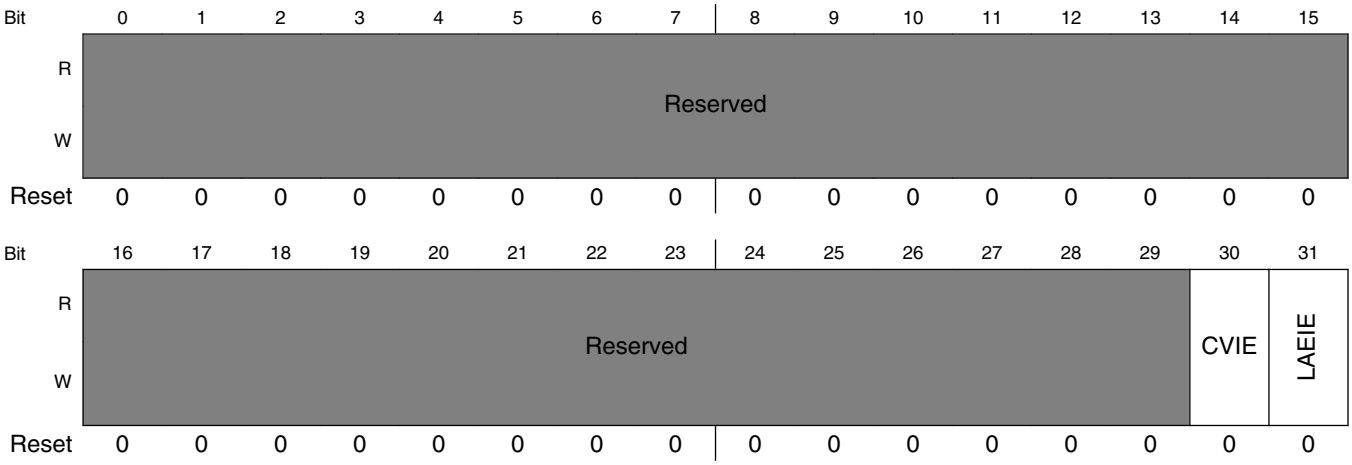


CCF_ERRDIS field descriptions

Field	Description
0–29 -	This field is reserved. Reserved
30 CVDIS	Coherency violation error disable. 0 Enable detection and capture of coherency violation errors. 1 Disable detection and capture of coherency violation errors.
31 LAEDIS	Local access error disable. 0 Enable detection and capture of local access errors. 1 Disable detection and capture of local access errors.

10.2.7 CCF Error Interrupt Enable Register (CCF_ERRINTEN)

Address: 1_8000h base + E48h offset = 1_8E48h

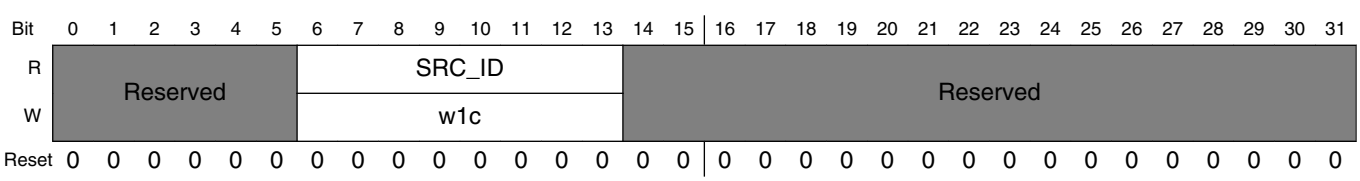


CCF_ERRINTEN field descriptions

Field	Description
0–29 -	This field is reserved. Reserved
30 CVIE	Coherency violation error interrupt enable. 0 Disable reporting of coherency violation errors as interrupts. 1 Enable reporting of coherency violation errors as interrupts.
31 LAEIE	Local access error interrupt enable. 0 Disable reporting of local access errors as interrupts. 1 Enable able reporting of local access errors as interrupts.

10.2.8 CCF Error Capture Attribute Register (CCF_CECAR)

Address: 1_8000h base + E4Ch offset = 1_8E4Ch



CCF_CECAR field descriptions

Field	Description
0–5 -	This field is reserved. Reserved

Table continues on the next page...

CCF_CECAR field descriptions (continued)

Field	Description
6–13 SRC_ID	The system transaction source ID[0:7]. See for valid encodings.
14–31 -	This field is reserved. Reserved

10.2.9 CCF Error Capture Address Register High (CCF_CECADDRH)

Address: 1_8000h base + E50h offset = 1_8E50h

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																								ADDRH							
W																									w1c							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CCF_CECADDRH field descriptions

Field	Description
0–23 -	This field is reserved. Reserved
24–31 ADDRH	ADDRH contains the high-order bits[0:7]of the captured error address.

10.2.10 CCF Error Capture Address Register Low (CCF_CECADDRL)

Address: 1_8000h base + E54h offset = 1_8E54h

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	ADDRL																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CCF_CECADDRL field descriptions

Field	Description
0–31 ADDRL	ADDRL contains the low-order bits [9:39] of the captured error address.

10.2.11 CCF Error Capture Attribute Register 2 (CCF_CECAR2)

Address: 1_8000h base + E58h offset = 1_8E58h

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

CCF_CECAR2 field descriptions

Field	Description
0–15 CCCCV	Core cache complex coherency violation. Bit 0 corresponds to core cache complex 0, bit 1 corresponds to core cache complex 1, and so on. A coherency violation causes 2 or more bits to be set among CCCCV and CPCCV.
16–22 -	This field is reserved. Reserved
23 CPCCV	CPC coherency violation. A coherency violation causes 2 or more bits to be set among CCCCV and CPCCV.
24–31 -	This field is reserved. Reserved

Chapter 11
Peripheral Access Management Unit (PAMU)

11.1 PAMU Introduction

This section provides an overview and features list for the peripheral access management unit (PAMU).

11.1.1 PAMU Overview

A multicore system, illustrated in Figure 11-1, consists of one or more Power Architecture processors, a system memory separate from other subsystems, and a number of I/O devices that may initiate transactions to system memory. The processors are linked over a host domain interconnect to each other, to the system memory and to one or more I/O devices or domains. Typically, a PAMU resides astride the boundary between the coherency and I/O domains, as shown in Figure 11-1 and enforces authorization and access control over DSA operations into the coherency domain.

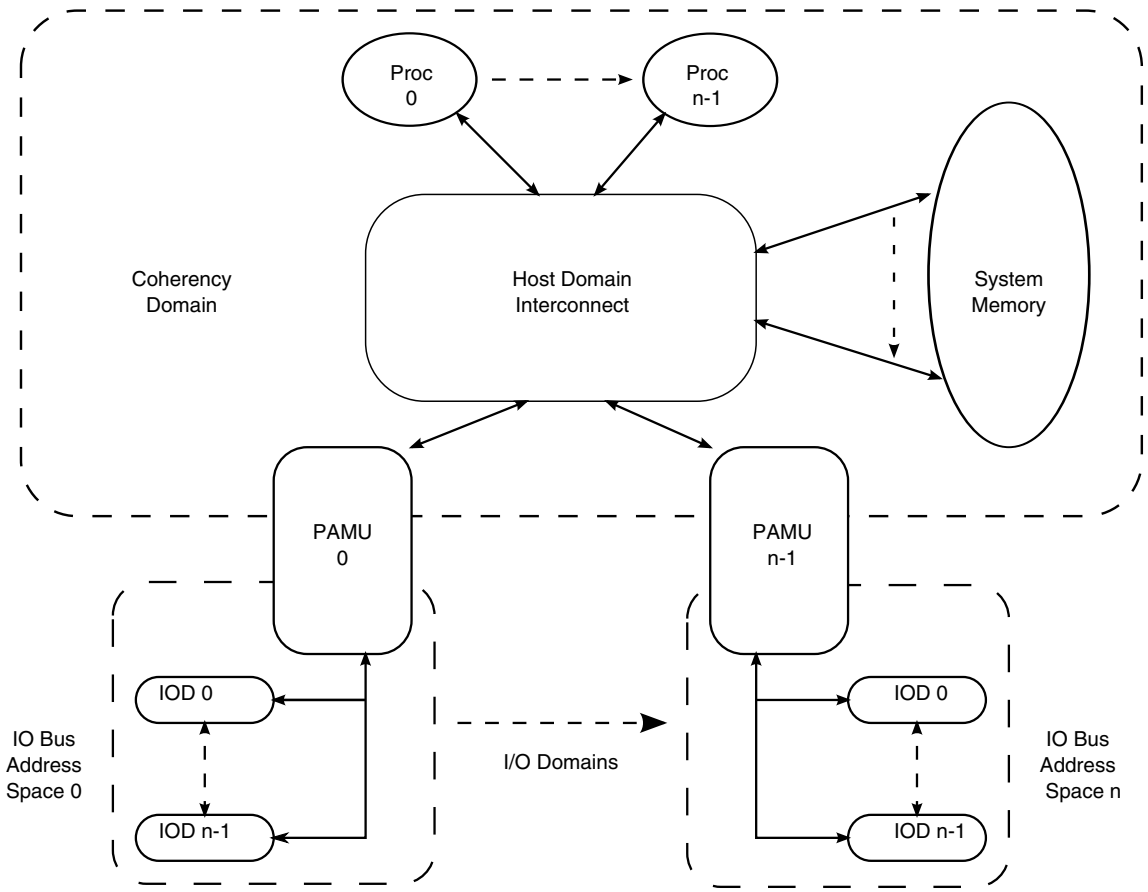


Figure 11-1. Illustration of a Multicore System

To accelerate the authorization and translation of operations, PAMU contains caches for quick lookup of memory resident data structures containing the authorization and translation attributes.

11.1.2 PAMU Features Summary

PAMU includes the following features:

- Support for authorizing DMA and peer-to-peer operations, that is, checking that the operation is originated by a valid I/O adapter, authorized to issue these operations, and that the operation is to a valid address window provided to that I/O adapter.
- Support for the access control of DMA and peer-to-peer operations, that is, allowing only the types of operations that fall within the access privileges provided to that I/O adapter.

- Support for address translation of DMA and peer-to-peer operations, that is, redirecting operations with an address in the I/O bus address space to a corresponding address in system memory space. The following address translation modes are supported:
  - No address translation
  - Window translation
- Support for operation type translation of DMA and peer-to-peer operations, that is, converting the ingress operation encoding to a corresponding egress operation encoding. The following operation translation modes are supported:
  - No Operation Translation
  - Immediate Operation Translation
  - Indexed Operation Translation for up to sixteen indexes, that is, the field is limited to OMI[0:3]
- Support for up to 4-Kbyte LIODNs, that is, signaling is limited to LIODN[4:15]
- Support for up to 256 DSA sub-windows when multiple windows are enabled, that is, the field is limited to WCE[1:3]
- Hardware coherent PAMU Look-aside caches to improve performance
  - A 32-entry, direct-mapped primary PAACT cache
  - A 128-entry, 2-way, set-associative secondary PAACT cache
  - A 4-entry, direct-mapped OMT cache
  - Support for receiving invalidating snoop operations for the PAMU look-aside caches (PLCs) from the coherency domain
  - Support for issuing PAMU fetch operations when the relevant entry is not present in the PLC
- Support for a maximum 40-bit system address space
- Support for a maximum 40-bit I/O address space
- Capability to generate interrupts
  - Access violations
  - PAMU operational errors

11.2 Data Structures Used by PAMU

The following data structures are used by the PAMU. The data structures are located in system memory and are accessed by the PAMU as necessary. These data structures are programmed by the hypervisor controlling the system.

11.2.1 Overview of Data Structures

11.2.1.1 Peripheral Access Authorization and Control Tables (PAACTs)

PAACTs are the primary data structures used by PAMU . A PAACT is a table of peripheral access authorization and control entries (PAACE). Each PAACE defines the range of I/O bus address space that is accessible by the LIOD and the associated access capabilities.

There are two types of PAACTs: primary PAACT (PPAACT) and secondary PAACT (SPAACT).

A given physical I/O device may be able to act as one or more independent logical I/O devices (LIODs). Each such logical I/O device is assigned an identifier called logical I/O device number (LIODN).

A LIOD is allocated a contiguous portion of the I/O bus address space called the DSA window for performing DSA operations. The DSA window may optionally be divided into multiple sub-windows, each of which may be used to map to a region in system storage space. The first sub-window is referred to as the primary sub-window and the remaining are called secondary sub-windows.

A DSA window is 2^n bytes in size. Sub-windows are declared in a two-level hierarchy: a DSA window and 2^m ($n > m > 0$) sub-windows within it. There are means available to recover unused secondary windows and unused address sub-ranges (See Recovering Address Space, for more information.). The sub-windows are labeled 0 through 2^F-1. Each sub-window is of size 2^{n-m} bytes. A sub-range of size 2^k within the sub-window may be specified for mapping. The sub-range can be narrower than the default sub-window size (2^{n-m} bytes).

The access attributes of the DSA window and the primary sub-window are declared using the primary PAACE (PPAACE) and those for the secondary sub-windows are declared using secondary PAACEs (SPAACEs). Only the primary PAACE is placed in the PPAACT. The secondary PAACEs are placed consecutively in the SPAACT in the order of their labels. A pointer in the PPAACE points to the placement of the first of the SPAACEs in the SPAACT. See Figure 11-2 and Figure 11-3.

DSA operations from a LIOD may be redirected to different locations in system storage space via the mechanism of address translation via a PAACE. The attributes that define whether address translation is enabled and the type of address translation, if enabled, are contained in the PAACE.

A LIOD may have restrictions as to the type of DSA operation it is allowed to perform. The attributes that define the access permissions for that LIOD are contained in the PAACE.

Furthermore, DSA operations from a LIOD may be translated to different operation types in system storage space via the mechanism of operation translation via a PAACE. The attributes that define whether operation translation is enabled and the type of operation translation, if enabled, are contained in the PAACE. The translation may optionally refer to a data structure called operation mapping table (See Operation Mapping Table, for more information.).

A LIOD presents its LIODN along with every DSA operation issued by it. The PAMU uses the LIODN as a means to identify the relevant PPAACE and then if necessary the SPAACEs and other data structures that specify the access capabilities provided to that LIOD.

11.2.1.2 Operation Mapping Table

DSA operations from a LIOD may optionally be converted to different operation types via the mechanism of operation translation. The conversion of operation types may be due to differences between the source and destination interconnect protocols of the DSA operation. The attributes that define operation translation is also contained in PAACE. Additional operation mapping attributes for that LIOD may be optionally contained in a operation mapping entry (OME) in the operation mapping table (OMT).

11.2.1.3 Access Capabilities Across LIODs

LIODs may be provided with a subset of the access capabilities described above as necessary. Certain LIODs may utilize a narrower set of the mechanisms provided while other LIODs simultaneously utilize a broader or the full set of mechanisms. Hypervisor may also choose to vary, over time, the provisions for an LIOD, as application requirements and the capabilities of the system vary.

11.2.2 Structure and Contents of PAACTs

Figure 11-2 displays the placement of the PPAACT within system memory space and the arrangement of PPAACEs within the PPAACT.

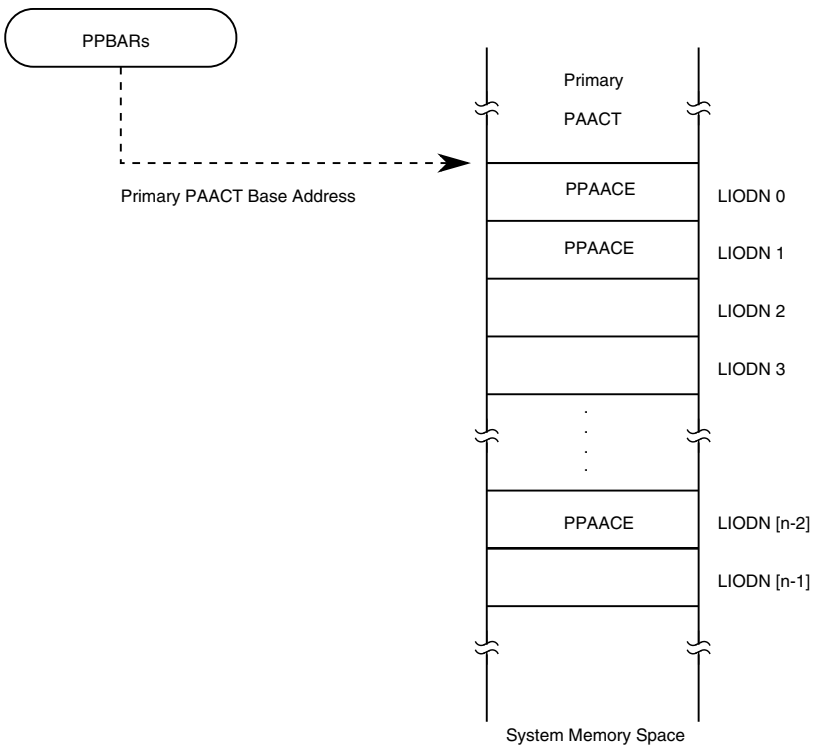


Figure 11-2. Arrangement of Primary PAACEs in System Memory Space

- The LIODN is used to index into the PPAACT to identify the PPAACE corresponding to the LIOD.
- The PPAACE located at the LIODN index of PPAACT contains fields indicating the base address and size of the DSA window for that LIOD (see Table 11-1). If the DSA window has properties that cannot be expressed in a single PAACE, multiple PAACEs are created. Fields in the PPAACE located at the LIODN index of PPAACT indicate the presence and number of multiple PAACEs for that LIOD:
  - The DSA window is divided into non-overlapping sub-windows of equal size with a PAACE containing the attributes of a sub-window.
  - The entry located at the LIODN index is called the primary PAACE. The primary PAACE contains the attributes describing the entire DSA window as well as the primary sub-window.
  - The SPAACEs are contained in the SPAACT starting at the first secondary PAACE index (FSPI) provided in the PPAACE and are arranged in linear order corresponding to the linear address order of the sub-window (see example in Figure 11-3).
  - A SPAACE describes the properties of a secondary DSA sub-window (see Table 11-2).

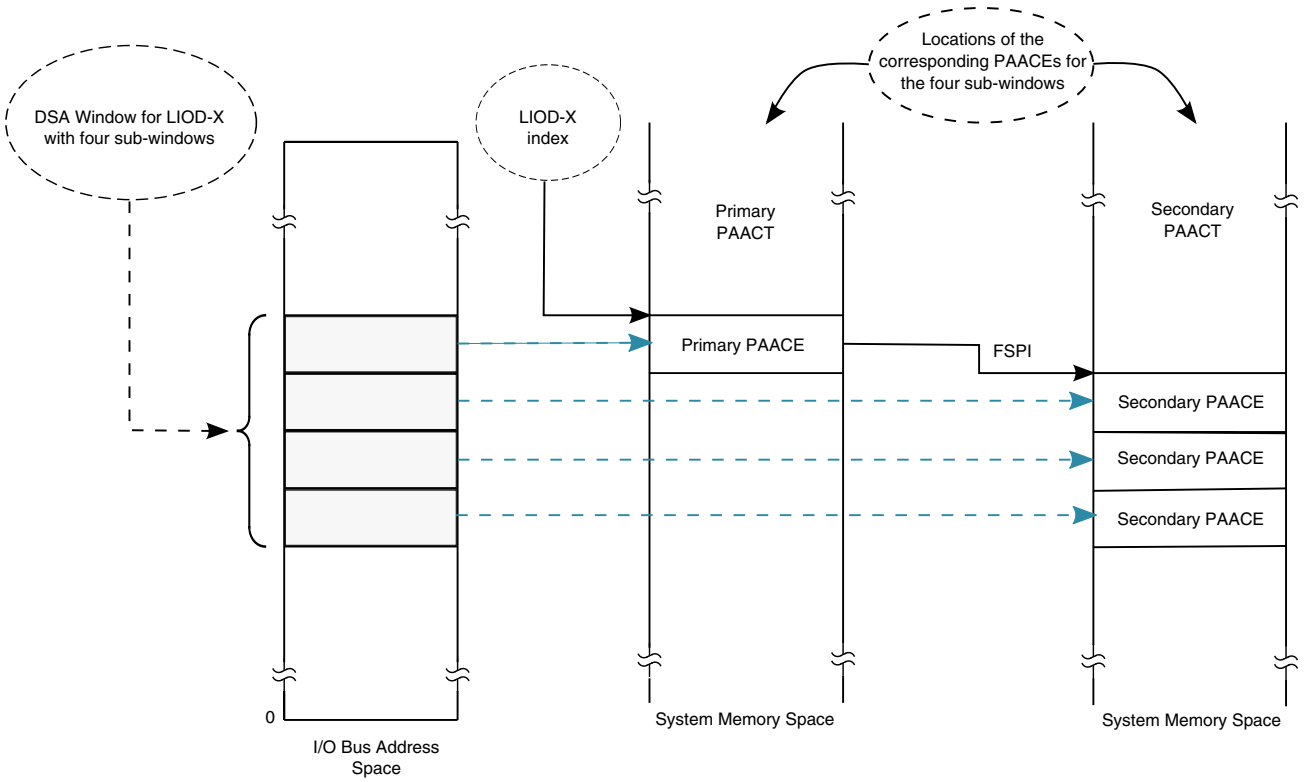


Figure 11-3. Example of DSA Sub-Windows and their Corresponding PAACEs

11.2.3 Peripheral Access Authorization and Control Entry (PAACE)

This section describes the structure and contents of a PAACE. The format of the primary PAACE data structure is shown in Table 11-1. Fields valid in only the primary PAACE apply to the entire DSA window defined by the primary PAACE. Fields valid in both the primary and secondary PAACE apply to the DSA sub-window defined by that PAACE.

Table 11-1. Primary PAACE Format

Index Offset (hex)	Bits 0:31	Bits 32:39	Bits 40:47	Bits 48:51	Bits 52:55	Bits 56:57	Bit 58	Bit 59	Bit 60	Bit 61	Bit 62	Bit 63
00	WBA				WSE		MW	AP		-	PT	V
08	DA	IA				WCE			ATM		OTM	
10	TWBA				SWSE		reserved					

Table continues on the next page...

Table 11-1. Primary PAACE Format (continued)

Index Offset (hex)	Bits 0:31	Bits 32:39	Bits 40:47	Bits 48:51	Bits 52:55	Bits 56:57	Bit 58	Bit 59	Bit 60	Bit 61	Bit 62	Bit 63
18	FSPI	IOEA	MOEA	IOEB		MOEB						
		reserved		OMI								
20-38	reserved											

The format of the secondary PAACE data structure is shown in Table 11-2.

Table 11-2. Secondary PAACE Format

Index Offset (hex)	Bits 0:31	Bits 32:39	Bits 40:47	Bits 48:51	Bits 52:55	Bits 56:57	Bit 58	Bit 59	Bit 60	Bit 61	Bit 62	Bit 63
00	reserved	LIODN		reserved				AP		-	PT	V
08	DA	IA				reserved			ATM		OTM	
10	TWBA				SWSE		reserved					
18	reserved	IOEA	MOEA	IOEB		MOEB						
	reserved			OMI								
20-38	reserved											

11.2.3.1 PAACE Offset 0x00

Table 11-3. PAACE Offset 0x00 Field Definitions

Bits	Name	PPAACE (P) and/or SPAACE (S) Field	Description
0-51	See Notes		Notes: If primary PAACE: bits 0-51 of PAACE offset 0x00 contain the WBA. If secondary PAACE: bits 32-47 of PAACE offset 0x00 contain the LIODN. The rest of the bits are reserved.
0-51	WBA	P	Window Base Address

Table continues on the next page...

Table 11-3. PAACE Offset 0x00 Field Definitions (continued)

Bits	Name	PPAACE (P) and/or SPAACE (S) Field	Description
			These bits describe the base address of the DSA window in I/O address space. The WBA is the 52 most significant address bits of a 64-bit address aligned to the window size as described in the WSE field. The WBA field is 52 bits wide to allow for a 64-bit address aligned to the minimum 4 Kbyte window size. These bits are valid in the primary PAACE. In certain platforms, all these bits may not be required. However, enough bits must be implemented to match the largest system storage address in the platform. Reads to unimplemented high order bits must return 0.
32-47	LIODN	S	Logical I/O Device Number These bits define the LIODN for which the DSA sub-window is being described. These bits are valid in the secondary PAACE.
52-57	WSE	P	Window Size Encoding These bits identify the size of the DSA window starting from the window base address. These bits are valid in the primary PAACE and are reserved in a secondary PAACE. DSA window size is $2^{(WSE+1)}$ bytes: 0x00-0x0A Reserved (Write reserved, read returns 0) 0x0B 4 Kbytes 0x0C 8 Kbytes . . . $2^{(WSE+1)}$ bytes 0x1F 4 Gbytes 0x20 8 Gbytes . . . $2^{(WSE+1)}$ bytes 0x3F 16 Ebytes
58	MW	P	Multiple Windows This bit indicates whether the LIOD has multiple DSA sub-windows. These bits are valid in the primary PAACE and are reserved in a secondary PAACE. It is coded as follows: 0 A single DSA window exists for the LIOD. 1 Multiple DSA sub-windows exist for the LIOD. The number of sub-windows is indicated in the WCE field of the PPAACE.
59-60	AP	P and S	Access Permissions. These bits define whether access to the window described by this PAACE is permitted and if permitted, the type of permitted access. In the field description, a Query Access typically refers to read type operations where the effect of the operation does not change the values of locations or state of the system; an Update Access typically refers to write type operations where the effect of the operation changes the values of locations or state of the system. 00 Access denied

Table continues on the next page...

Table 11-3. PAACE Offset 0x00 Field Definitions (continued)

Bits	Name	PPAACE (P) and/or SPAACE (S) Field	Description
			01 Query access only 10 Update access only 11 Access permitted for all operation types **NOTE:** If MW = 0, then AP applies to the DSA wIndow described by this PAACE; if MW = 1, then AP applies to the DSA sub-window described by this PAACE.
61	-	P and S	Reserved
62	PT	P and S	PAACE Type This bit indicates whether it is a primary or secondary PAACE. 0 Primary PAACE 1 Secondary PAACE
63	V	P and S	Valid PAACE This bit defines whether this PAACE is valid. 0 Invalid PAACE. Other PAACE fields do not contain valid information. 1 Valid PAACE. Other PAACE fields may be referenced.

11.2.3.2 PAACE Offset 0x08

Table 11-4. PAACE Offset 0x08 Field Definitions

Bits	Name	PPAACE (P) and/or SPAACE (S) Field	Description
0-31	DA	P and S	Domain Attributes These bits define operation attributes specific to the destination domain of the operation. The contents of the DA field are further described in PAACE Domain Attributes .
32-55	IA	P and S	Implementation Attributes These bits define operation attributes that are implementation specific. For details see Implementation Attributes
56-59	WCE	P	Window Count Encoding These bits identify the number of DSA sub-windows for this LIOD. These bits are valid in the primary PAACE when MW is enabled and are reserved in a secondary PAACE. Window count is $2^{(WCE+1)}$ windows: 0000 2 DSA sub-windows 0001 4 DSA sub-windows . . .

Table continues on the next page...

Table 11-4. PAACE Offset 0x08 Field Definitions (continued)

Bits	Name	PPAACE (P) and/or SPAACE (S) Field	Description
			1111 65536 DSA sub-windows
60-61	ATM	P and S	Address Translation Mode These bits define whether address translation is enabled for the window described by this PAACE and if enabled, the type of translation to perform. 00 No address translation. The ingress and egress operation address are the same. 01 Window address translation 10 Reserved 11 Reserved
62-63	OTM	P and S	Operation Translation Mode These bits define whether operation type translation is enabled for the window described by this PAACE and if enabled, the type of translation to perform. 00 No operation translation. The ingress and egress operation type are the same. 01 Operation translation enabled with immediate operation translation. 10 Operation translation enabled with indexed operation translation. 11 Reserved

11.2.3.2.1 PAACE Domain Attributes

Table 11-5 describes the domain attribute (DA) field of PAACE at offset 0x08-0x0B.

Table 11-5. PAACE Domain Attributes

Bits	Name	PPAACE (P) and/or SPAACE (S) Field	Description
0-15	-	-	Reserved (Write reserved; read returns 0)
16-23	SnpID	P and S	Snoop ID This field may be used to identify a specific snooper or set of snoopers required for the maintenance of hardware coherency for the window described by this PAACE. This field is valid if the M bit for the PAACE is 1. If the SnpID field contains a pattern of all 0s, then all snoopers in the coherency domain must participate in the maintenance of hardware coherency. A SnpID pattern of all 0s indicates no specific snooper or set of snoopers is being specified. If the SnpID pattern is non-zero, only the snoopers

Table continues on the next page...

Table 11-5. PAACE Domain Attributes (continued)

Bits	Name	PPAACE (P) and/or SPAACE (S) Field	Description
			defined by the SnpID field must participate in the maintenance of hardware coherency. The snoopers indicated by the non-zero SnpID pattern is implementation dependent.
24	M	P and S	Memory Coherence Required This bit defines if hardware cache coherency is to be maintained in the coherency domain for the window described by this PAACE. 0 Non-coherent memory, that is, hardware cache coherency mechanisms not required. 1 Coherent memory, that is, hardware cache coherency mechanisms required.
25-31	-	-	Reserved (Write reserved; read returns 0)

11.2.3.3 PAACE Offset 0x10

Table 11-6. PAACE Offset 0x10 Field Definitions

Bits	Name	PPAACE (P) and/or SPAACE (S) Field	Description
0-51	TWBA	P and S	Translated Window Base Address These bits describe the translated base address in system storage space of the window described by this PAACE. The TWBA is the 52 most significant address bits of the 64-bit address, aligned to the size of the window described by this PAACE. The TWBA field is 52 bits wide to allow for a 64-bit address aligned to the minimum 4 Kbyte size of a window described by this PAACE. This field is referenced when ATM[0:1] = 01 for this PAACE.
52-57	SWSE	P and S	Sub-Window Sub-Range Encoding These bits identify the size of the sub-range of addresses within the DSA sub-window described by this PAACE, that is eligible for access and translation. This eligible sub-range starts from the starting address of this DSA sub-window. This field is only referenced when the MW bit is 1. The sub-range size is $2^{(SWSE+1)}$ bytes: 0x00-0x0A Reserved (Write reserved; read returns 0) 0x0B 4 Kbyte 0x0C 8 Kbyte . . . $2^{(SWSE+1)}$ bytes 0x1F 4 Gbyte 0x20 8 Gbyte . . . $2^{(SWSE+1)}$ bytes

Table continues on the next page...

Table 11-6. PAACE Offset 0x10 Field Definitions (continued)

Bits	Name	PPAACE (P) and/or SPAACE (S) Field	Description
			0x3F 16Ebytes
58-63	-	-	Reserved (Write reserved; read returns 0)

11.2.3.4 PAACE Offset 0x18

Table 11-7. PAACE Offset 0x18 Field Definitions

Bits	Name	PPAACE (P) and/or SPAACE (S) Field	Description
0-31	FSPI	P	First Secondary PAACE Index These bits identify the index location in the SPAACT of the secondary PAACE that describes the second DSA sub-window for this LIOD. These bits are valid in the primary PAACE when MW is set and are reserved in a secondary PAACE.
32-63	See Notes		**Notes:** 1. If OTM[0:1] == 01, bits 32-63 of PAACE Offset 0x18 contain IOEA, MOEA, IOEB, and MOEB respectively. If OTM[0:1] == 10, bits 48-63 of PAACE Offset 0x18 contain the OMI, bits 32-47 are reserved. 2. The following apply to ingress operation encodings: bit[0] is a qualifier indicating the operation type. bit[0] = 0 is the qualifier value reserved for all query operation types. bit[0] = 1 is the qualifier value reserved for all update operation types. The operation encoding in IOE[1:7] is implementation dependent. 3. The following apply to egress operation encodings: bit[0] is a qualifier indicating a valid operation type. The operation encoding in EOE[1:7] is implementation dependent.
48-63	OMI	P and S	Operation Mapping Index If OTM[0:1] == 10, these bits are used to determine the relevant operation mapping entry (OME) of the OMT to be referenced for operation translation.
32-39	IOEA	P and S	Ingress Operation Encoding A If OTM[0:1] == 01, these bits define the ingress operation encoding A for which the corresponding egress encoding will be provided.
40-47	MOEA	P and S	Mapped Operation Encoding A

Table continues on the next page...

Table 11-7. PAACE Offset 0x18 Field Definitions (continued)

Bits	Name	PPAACE (P) and/or SPAACE (S) Field	Description
			These bits define the mapped operation encoding, including a portion of the translated egress operation encoding A (EOEA) for the ingress operation encoding A. MOEA[0] is a valid bit MOEA[1:7] contains the corresponding bits of the egress operation encoding, that is, EOEA[1:7].
48-55	IOEB	P and S	Ingress Operation Encoding B If OTM[0:1] == 01, these bits define the ingress operation encoding B for which the corresponding egress encoding will be provided.
56-63	MOEB	P and S	Mapped Operation Encoding B These bits define the mapped operation encoding, including a portion of the translated egress operation encoding B (EOEB) for the ingress operation encoding B. MOEB[0] is a valid bit MOEB[1:7] contains the corresponding bits of the egress operation encoding, that is, EOEB[1:7].

11.3 PAMU Memory Map/Register Definitions

PAMU registers reside in a 4-Kbyte block of CCSR space starting at 0x02_*n*000, where *n* is the instance of PAMU being referenced.

11.4 PAMU Memory Map

PAMU memory map

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
2_0000	Primary PAACT Base Address High register (PAMU1_PPBAH)	32	R/W	0000_0000h	11.4.1/396
2_0004	Primary PAACT Base Address Low register (PAMU1_PPBAL)	32	R/W	0000_0000h	11.4.2/397
2_0008	Primary PAACT Limit Address High register (PAMU1_PPLAH)	32	R/W	0000_0000h	11.4.3/397
2_000C	Primary PAACT Limit Address Low register (PAMU1_PPLAL)	32	R/W	0000_0000h	11.4.4/397

Table continues on the next page...

PAMU memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
2_0010	Secondary PAACT Base Address High register (PAMU1_SPBAH)	32	R/W	0000_0000h	11.4.5/398
2_0014	Secondary PAACT Base Address Low register (PAMU1_SPBAL)	32	R/W	0000_0000h	11.4.6/398
2_0018	Secondary PAACT Limit Address High register (PAMU1_SPLAH)	32	R/W	0000_0000h	11.4.7/399
2_001C	Secondary PAACT Limit Address Low register (PAMU1_SPLAL)	32	R/W	0000_0000h	11.4.8/399
2_0020	OMT Base Address High register (PAMU1_OBAH)	32	R/W	0000_0000h	11.4.9/400
2_0024	OMT Base Address Low register (PAMU1_OBAL)	32	R/W	0000_0000h	11.4.10/400
2_0028	OMT Limit Address High register (PAMU1_OLAH)	32	R/W	0000_0000h	11.4.11/401
2_002C	OMT Limit Address Low register (PAMU1_OLAL)	32	R/W	0000_0000h	11.4.12/401
2_0030	PAMU Address Capabilities Register 1 (PAMU1_PAC1)	32	R	0000_000Fh	11.4.13/402
2_0034	PAMU Address Capabilities Register 2 (PAMU1_PAC2)	32	R	0000_00FFh	11.4.14/402
2_0040	PAMU Operation Error Status register 1 (PAMU1_POES1)	32	R/W	0000_0000h	11.4.15/403
2_0044	PAMU Operation Error Status register 2 (PAMU1_POES2)	32	R/W	0000_0000h	11.4.16/404
2_0048	PAMU Operation Error Address High register (PAMU1_POEAH)	32	R/W	0000_0000h	11.4.17/404
2_004C	PAMU Operation Error Address Low register (PAMU1_POEAL)	32	R/W	0000_0000h	11.4.18/405
2_0050	Access Violation Status register 1 (PAMU1_AVS1)	32	R/W	0000_0000h	11.4.19/406
2_0054	Access Violation Status register 2 (PAMU1_AVS2)	32	R/W	0000_0000h	11.4.20/408
2_0058	Access Violation Address High register (PAMU1_AVAH)	32	R/W	0000_0000h	11.4.21/409
2_005C	Access Violation Address Low register (PAMU1_AVAL)	32	R/W	0000_0000h	11.4.22/409
2_0060	ECC Error Control Register (PAMU1_EECTL)	32	R/W	0000_0000h	11.4.23/409
2_0068	ECC Error Interrupt Enable Register (PAMU1_EEINTEN)	32	R/W	0000_0000h	11.4.24/410
2_006C	ECC Error Detect Register (PAMU1_EEDET)	32	w1c	0000_0000h	11.4.25/411
2_0070	ECC Error Attributes Register (PAMU1_EEATTR)	32	R/W	0000_0000h	11.4.26/412
2_0074	ECC Error Address High (PAMU1_EEAHI)	32	R/W	0000_0000h	11.4.27/413
2_0078	ECC Error Address Low (PAMU1_EEALO)	32	R/W	0000_0000h	11.4.28/414
2_007C	ECC Error Data High (PAMU1_EEDHI)	32	R/W	0000_0000h	11.4.29/414
2_0080	ECC Error Data Low (PAMU1_EEDLO)	32	R/W	0000_0000h	11.4.30/415
2_0090	Unauthorized device access detection register (PAMU1_UDAD)	32	R/W	0000_0000h	11.4.31/415
2_0BF8	PAMU Revision register 1 (PAMU1_PR1)	32	R	0901_0101h	11.4.32/416
2_0BFC	PAMU Revision register 2 (PAMU1_PR2)	32	R	0000_0000h	11.4.33/416
2_0C00	PAMU Capabilities register 1 (PAMU1_PC1)	32	R	0000_000Fh	11.4.34/417
2_0C04	PAMU Capabilities register 2 (PAMU1_PC2)	32	R	0FFF_00FFh	11.4.35/418
2_0C08	PAMU Capabilities register 3 (PAMU1_PC3)	32	R	007E_1E00h	11.4.36/419
2_0C0C	PAMU Capabilities register 4 (PAMU1_PC4)	32	R	0000_0CE1h	11.4.37/421

Table continues on the next page...

PAMU memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
2_0C10	PAMU Control register (PAMU1_PC)	32	R/W	0000_1111h	11.4.38/422
2_0C1C	PAMU Interrupt Control and Status register (PAMU1_PICS)	32	R/W	0000_0000h	11.4.39/423
2_1000	Primary PAACT Base Address High register (PAMU2_PPBAH)	32	R/W	0000_0000h	11.4.1/396
2_1004	Primary PAACT Base Address Low register (PAMU2_PPBAL)	32	R/W	0000_0000h	11.4.2/397
2_1008	Primary PAACT Limit Address High register (PAMU2_PPLAH)	32	R/W	0000_0000h	11.4.3/397
2_100C	Primary PAACT Limit Address Low register (PAMU2_PPLAL)	32	R/W	0000_0000h	11.4.4/397
2_1010	Secondary PAACT Base Address High register (PAMU2_SPBAH)	32	R/W	0000_0000h	11.4.5/398
2_1014	Secondary PAACT Base Address Low register (PAMU2_SPBAL)	32	R/W	0000_0000h	11.4.6/398
2_1018	Secondary PAACT Limit Address High register (PAMU2_SPLAH)	32	R/W	0000_0000h	11.4.7/399
2_101C	Secondary PAACT Limit Address Low register (PAMU2_SPLAL)	32	R/W	0000_0000h	11.4.8/399
2_1020	OMT Base Address High register (PAMU2_OBAH)	32	R/W	0000_0000h	11.4.9/400
2_1024	OMT Base Address Low register (PAMU2_OBAL)	32	R/W	0000_0000h	11.4.10/400
2_1028	OMT Limit Address High register (PAMU2_OLAH)	32	R/W	0000_0000h	11.4.11/401
2_102C	OMT Limit Address Low register (PAMU2_OLAL)	32	R/W	0000_0000h	11.4.12/401
2_1030	PAMU Address Capabilities Register 1 (PAMU2_PAC1)	32	R	0000_000Fh	11.4.13/402
2_1034	PAMU Address Capabilities Register 2 (PAMU2_PAC2)	32	R	0000_00FFh	11.4.14/402
2_1040	PAMU Operation Error Status register 1 (PAMU2_POES1)	32	R/W	0000_0000h	11.4.15/403
2_1044	PAMU Operation Error Status register 2 (PAMU2_POES2)	32	R/W	0000_0000h	11.4.16/404
2_1048	PAMU Operation Error Address High register (PAMU2_POEAH)	32	R/W	0000_0000h	11.4.17/404
2_104C	PAMU Operation Error Address Low register (PAMU2_POEAL)	32	R/W	0000_0000h	11.4.18/405
2_1050	Access Violation Status register 1 (PAMU2_AVS1)	32	R/W	0000_0000h	11.4.19/406
2_1054	Access Violation Status register 2 (PAMU2_AVS2)	32	R/W	0000_0000h	11.4.20/408
2_1058	Access Violation Address High register (PAMU2_AVAH)	32	R/W	0000_0000h	11.4.21/409
2_105C	Access Violation Address Low register (PAMU2_AVAL)	32	R/W	0000_0000h	11.4.22/409
2_1060	ECC Error Control Register (PAMU2_EECTL)	32	R/W	0000_0000h	11.4.23/409
2_1068	ECC Error Interrupt Enable Register (PAMU2_EEINTEN)	32	R/W	0000_0000h	11.4.24/410
2_106C	ECC Error Detect Register (PAMU2_EEDET)	32	w1c	0000_0000h	11.4.25/411
2_1070	ECC Error Attributes Register (PAMU2_EEATTR)	32	R/W	0000_0000h	11.4.26/412
2_1074	ECC Error Address High (PAMU2_EEAHI)	32	R/W	0000_0000h	11.4.27/413
2_1078	ECC Error Address Low (PAMU2_EEALO)	32	R/W	0000_0000h	11.4.28/414
2_107C	ECC Error Data High (PAMU2_EEDHI)	32	R/W	0000_0000h	11.4.29/414

Table continues on the next page...

PAMU memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
2_1080	ECC Error Data Low (PAMU2_EEDLO)	32	R/W	0000_0000h	11.4.30/415
2_1090	Unauthorized device access detection register (PAMU2_UDAD)	32	R/W	0000_0000h	11.4.31/415
2_1BF8	PAMU Revision register 1 (PAMU2_PR1)	32	R	0901_0101h	11.4.32/416
2_1BFC	PAMU Revision register 2 (PAMU2_PR2)	32	R	0000_0000h	11.4.33/416
2_1C00	PAMU Capabilities register 1 (PAMU2_PC1)	32	R	0000_000Fh	11.4.34/417
2_1C04	PAMU Capabilities register 2 (PAMU2_PC2)	32	R	0FFF_00FFh	11.4.35/418
2_1C08	PAMU Capabilities register 3 (PAMU2_PC3)	32	R	007E_1E00h	11.4.36/419
2_1C0C	PAMU Capabilities register 4 (PAMU2_PC4)	32	R	0000_0CE1h	11.4.37/421
2_1C10	PAMU Control register (PAMU2_PC)	32	R/W	0000_1111h	11.4.38/422
2_1C1C	PAMU Interrupt Control and Status register (PAMU2_PICS)	32	R/W	0000_0000h	11.4.39/423
2_2000	Primary PAACT Base Address High register (PAMU3_PPBAH)	32	R/W	0000_0000h	11.4.1/396
2_2004	Primary PAACT Base Address Low register (PAMU3_PPBAL)	32	R/W	0000_0000h	11.4.2/397
2_2008	Primary PAACT Limit Address High register (PAMU3_PPLAH)	32	R/W	0000_0000h	11.4.3/397
2_200C	Primary PAACT Limit Address Low register (PAMU3_PPLAL)	32	R/W	0000_0000h	11.4.4/397
2_2010	Secondary PAACT Base Address High register (PAMU3_SPBAH)	32	R/W	0000_0000h	11.4.5/398
2_2014	Secondary PAACT Base Address Low register (PAMU3_SPBAL)	32	R/W	0000_0000h	11.4.6/398
2_2018	Secondary PAACT Limit Address High register (PAMU3_SPLAH)	32	R/W	0000_0000h	11.4.7/399
2_201C	Secondary PAACT Limit Address Low register (PAMU3_SPLAL)	32	R/W	0000_0000h	11.4.8/399
2_2020	OMT Base Address High register (PAMU3_OBAH)	32	R/W	0000_0000h	11.4.9/400
2_2024	OMT Base Address Low register (PAMU3_OBAL)	32	R/W	0000_0000h	11.4.10/400
2_2028	OMT Limit Address High register (PAMU3_OLAH)	32	R/W	0000_0000h	11.4.11/401
2_202C	OMT Limit Address Low register (PAMU3_OLAL)	32	R/W	0000_0000h	11.4.12/401
2_2030	PAMU Address Capabilities Register 1 (PAMU3_PAC1)	32	R	0000_000Fh	11.4.13/402
2_2034	PAMU Address Capabilities Register 2 (PAMU3_PAC2)	32	R	0000_00FFh	11.4.14/402
2_2040	PAMU Operation Error Status register 1 (PAMU3_POES1)	32	R/W	0000_0000h	11.4.15/403
2_2044	PAMU Operation Error Status register 2 (PAMU3_POES2)	32	R/W	0000_0000h	11.4.16/404
2_2048	PAMU Operation Error Address High register (PAMU3_POEAH)	32	R/W	0000_0000h	11.4.17/404
2_204C	PAMU Operation Error Address Low register (PAMU3_POEAL)	32	R/W	0000_0000h	11.4.18/405
2_2050	Access Violation Status register 1 (PAMU3_AVS1)	32	R/W	0000_0000h	11.4.19/406
2_2054	Access Violation Status register 2 (PAMU3_AVS2)	32	R/W	0000_0000h	11.4.20/408

Table continues on the next page...

PAMU memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
2_2058	Access Violation Address High register (PAMU3_AVAH)	32	R/W	0000_0000h	11.4.21/409
2_205C	Access Violation Address Low register (PAMU3_AVAL)	32	R/W	0000_0000h	11.4.22/409
2_2060	ECC Error Control Register (PAMU3_EECTL)	32	R/W	0000_0000h	11.4.23/409
2_2068	ECC Error Interrupt Enable Register (PAMU3_EEINTEN)	32	R/W	0000_0000h	11.4.24/410
2_206C	ECC Error Detect Register (PAMU3_EEDET)	32	w1c	0000_0000h	11.4.25/411
2_2070	ECC Error Attributes Register (PAMU3_EEATTR)	32	R/W	0000_0000h	11.4.26/412
2_2074	ECC Error Address High (PAMU3_EEAHI)	32	R/W	0000_0000h	11.4.27/413
2_2078	ECC Error Address Low (PAMU3_EEALO)	32	R/W	0000_0000h	11.4.28/414
2_207C	ECC Error Data High (PAMU3_EEDHI)	32	R/W	0000_0000h	11.4.29/414
2_2080	ECC Error Data Low (PAMU3_EEDLO)	32	R/W	0000_0000h	11.4.30/415
2_2090	Unauthorized device access detection register (PAMU3_UDAD)	32	R/W	0000_0000h	11.4.31/415
2_2BF8	PAMU Revision register 1 (PAMU3_PR1)	32	R	0901_0101h	11.4.32/416
2_2BFC	PAMU Revision register 2 (PAMU3_PR2)	32	R	0000_0000h	11.4.33/416
2_2C00	PAMU Capabilities register 1 (PAMU3_PC1)	32	R	0000_000Fh	11.4.34/417
2_2C04	PAMU Capabilities register 2 (PAMU3_PC2)	32	R	0FFF_00FFh	11.4.35/418
2_2C08	PAMU Capabilities register 3 (PAMU3_PC3)	32	R	007E_1E00h	11.4.36/419
2_2C0C	PAMU Capabilities register 4 (PAMU3_PC4)	32	R	0000_0CE1h	11.4.37/421
2_2C10	PAMU Control register (PAMU3_PC)	32	R/W	0000_1111h	11.4.38/422
2_2C1C	PAMU Interrupt Control and Status register (PAMU3_PICS)	32	R/W	0000_0000h	11.4.39/423

11.4.1 Primary PAACT Base Address High register (PAMUx_PPBAH)

A write to the primary PAACT Base Address High register invalidates the contents of Primary PAACT cache.

Address: Base address + 0h 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	PPBAH																															
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

PAMUx_PPBAH field descriptions

Field	Description
0–31 PPBAH	Primary PAACT Base Address High The PPBAH is the 32 most significant bits of the 64-bit base address of the primary PAACT. **NOTE:** The most significant bits of PPBAH should have a value of 0 to be consistent with the MSA capability field as described in PAMU Capabilities register 1 (PAMU_PC1).

11.4.2 Primary PAACT Base Address Low register (PAMUx_PPBAL)

A write to the primary PAACT Base Address Low register invalidates the contents of Primary PAACT cache.

Address: Base address + 4h 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	PPBAL																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

PAMUx_PPBAL field descriptions

Field	Description
0–19 PPBAL	Primary PAACT Base Address Low The PPBAL is the 20 next-significant of the 64 bits that identify the primary PAACT base address.
20–31 -	This field is reserved. Reserved. These bits are the 12 least significant base address bits of the primary PAACT base address which must be aligned to a 4-Kbyte page boundary.

11.4.3 Primary PAACT Limit Address High register (PAMUx_PPLAH)

Address: Base address + 8h 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	PPLAH																															
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

PAMUx_PPLAH field descriptions

Field	Description
0–31 PPLAH	Primary PAACT Limit Address High The PPLAH is the 32 most significant of the 64 address bits that identify the limit of the primary PAACT. **NOTE:** The most significant bits of PPLAH should have a value of 0 to be consistent with the MSA capability field as described in PAMU Capabilities register 1 (PAMU_PC1).

11.4.4 Primary PAACT Limit Address Low register (PAMUx_PPLAL)

Address: Base address + Ch 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	PPLAL																										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

PAMUx_PPLAL field descriptions

Field	Description
0–25 PPLAL	Primary PAACT Limit Address Low The PPLAL is the 26 next-significant of the 64 bits that identify the limit of the primary PAACT. **NOTE:** The primary PAACT limit is the address of the byte following the last byte of the PPAACT.
26–31 -	This field is reserved. Reserved. These bits are the 6 least significant address bits representing the 64-byte PAACE size.

11.4.5 Secondary PAACT Base Address High register (PAMUx_SPBAH)

A write to the secondary PAACT Base Address High register invalidates the contents of secondary PAACT cache.

Address: Base address + 10h 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	SPBAH																															
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

PAMUx_SPBAH field descriptions

Field	Description
0–31 SPBAH	Secondary PAACT Base Address High The SPBAH is the 32 most-significant of the 64 bits that identify the secondary PAACT base address. **NOTE:** The most significant bits of SPBAH should have a value of 0 to be consistent with the MSA capability field as described in PAMU Capabilities register 1 (PAMU_PC1).

11.4.6 Secondary PAACT Base Address Low register (PAMUx_SPBAL)

A write to the secondary PAACT Base Address Low register invalidates the contents of secondary PAACT cache.

Address: Base address + 14h 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	SPBAL																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

PAMUx_SPBAL field descriptions

Field	Description
0–19 SPBAL	Secondary PAACT Base Address Low The SPBAL is the 20 next-significant of the 64 bits that identify the secondary PAACT base address.
20–31 -	This field is reserved. Reserved. These bits are the 12 least significant base address bits of the secondary PAACT which must be aligned to a 4-Kbyte page boundary.

11.4.7 Secondary PAACT Limit Address High register (PAMUx_SPLAH)

Address: Base address + 18h 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	SPLAH																															
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

PAMUx_SPLAH field descriptions

Field	Description
0–31 SPLAH	Secondary PAACT Limit Address High The SPLAH is the 32 most-significant of the 64 bits that identify the limit of the secondary PAACT. **NOTE:** The most significant bits of SPLAH should have a value of 0 to be consistent with the MSA capability field as described in PAMU Capabilities register 1 (PAMU_PC1).

11.4.8 Secondary PAACT Limit Address Low register (PAMUx_SPLAL)

Address: Base address + 1Ch 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	SPLAL																										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

PAMUx_SPLAL field descriptions

Field	Description
0–25 SPLAL	Secondary PAACT Limit Address Low The SPLAL is the 26 next-significant of the 64 bits that identify the limit of the secondary PAACT. **NOTE:** The secondary PAACT limit is the address of the byte following the last byte of the SPAACT.
26–31 -	This field is reserved. Reserved. These bits are the 6 least significant address bits representing the 64-byte PAACE size.

11.4.9 OMT Base Address High register (PAMUx_OBAH)

A write to OMT Base Address High register invalidates the contents of OMT cache.

Address: Base address + 20h 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	OBAH																															
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

PAMUx_OBAH field descriptions

Field	Description
0–31 OBAH	OMT Base Address High The OBAH is the 32 most significant address bits that identify the base of the OMT window. **NOTE:** The most significant bits of OBAH should have a value of 0 to be consistent with the MSA capability field as described in PAMU Capabilities 1.

11.4.10 OMT Base Address Low register (PAMUx_OBAL)

A write to OMT Base Address Low register invalidates the contents of OMT cache.

Address: Base address + 24h 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	OBAL																				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

PAMUx_OBAL field descriptions

Field	Description
0–19 OBAL	OMT Base Address Low The OBAL is the next 20 least significant address bits that identify the base of the OMT window above the 12 least significant address bits representing the minimum 4-Kbyte page boundary for the base address.
20–31 -	This field is reserved. Reserved. These bits are the 12 least significant address bits a 4-Kbyte page boundary.

11.4.11 OMT Limit Address High register (PAMUx_OLAH)

Address: Base address + 28h 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	OLAH																															
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

PAMUx_OLAH field descriptions

Field	Description
0–31 OLAH	OMT Limit Address High The OLAH is the 32 most significant address bits that identify the limit of the OMT window. **NOTE:** The most significant bits of OLAH should have a value of 0 to be consistent with the MSA capability field as described in PAMU Capabilities 1.

11.4.12 OMT Limit Address Low register (PAMUx_OLAL)

Address: Base address + 2Ch 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	OLAL																									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

PAMUx_OLAL field descriptions

Field	Description
0–24 OLAL	OMT Limit Address Low The OLAL is the next 25 least significant address bits that identify the limit of the OMT window above the 7 least significant address bits representing the 128-byte size of an OME.
25–31 -	This field is reserved. Reserved. These bits are the 7 least significant address bits representing the 128-byte size of an OME.

11.4.13 PAMU Address Capabilities Register 1 (PAMUx_PAC1)

The PAMU address capabilities registers denote implementation parameters for a given hardware instance of the PAMU. Software must read these registers to discover the specific parameter values implemented in this instance of PAMU and utilize the information when programming the PAMU, its data structures, and related system resources.

Address: Base address + 30h 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	MSA																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

PAMUx_PAC1 field descriptions

Field	Description
0–31 MSA	Maximum System Storage Address Capability. These bits define the maximum system storage address capability of this PAMU implementation. The MSA identifies the maximum value of the 32 most-significant address bits above the 32 least-significant address bits representing 4 Gbytes. This PAMU implements only the least significant MSA number of bits of the TWBA field within a PAACE. **NOTE:** These bits do not define the maximum address capability of the system. A PAMU implementation can be used in a system where system address capability is less than or equal to maximum system-storage address capability of the PAMU. 0x0000_00FF Maximum 40-bit System Address Space

11.4.14 PAMU Address Capabilities Register 2 (PAMUx_PAC2)

The PAMU address capabilities registers denote implementation parameters for a given hardware instance of the PAMU. Software must read these registers to discover the specific parameter values implemented in this instance of PAMU and utilize the information when programming the PAMU, its data structures, and related system resources.

Address: Base address + 34h offset

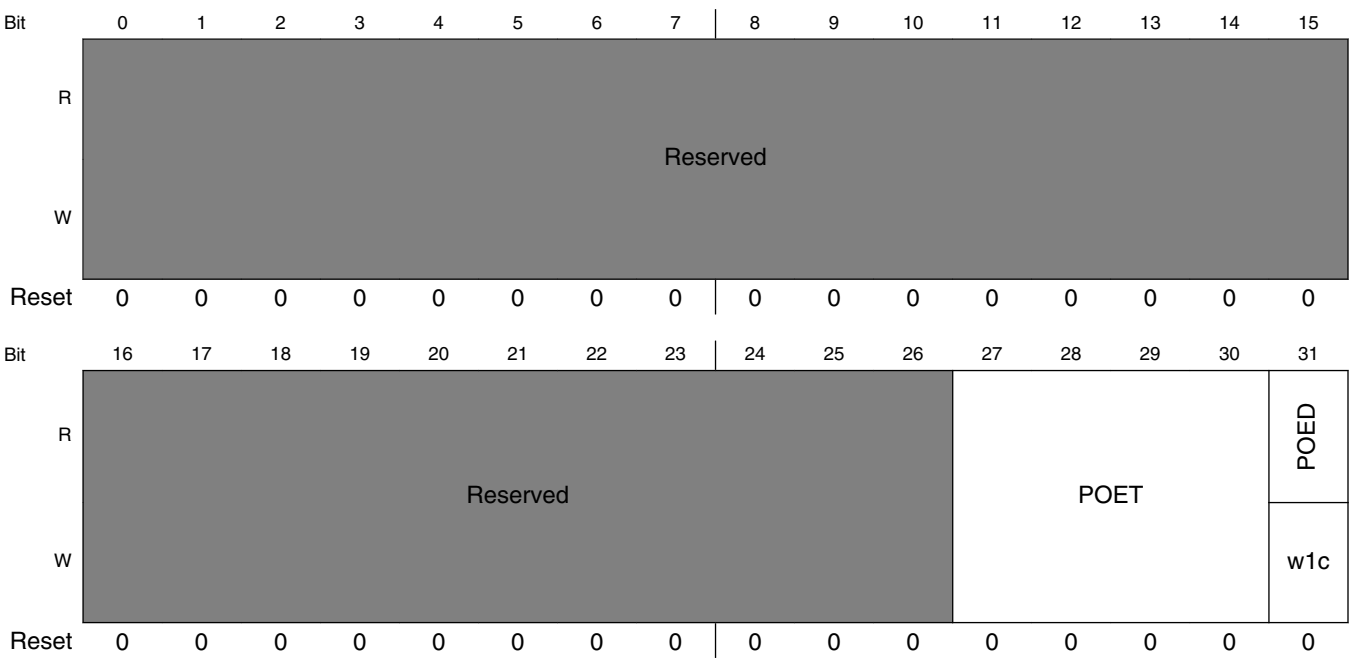
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	MIA																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

PAMUx_PAC2 field descriptions

Field	Description
0–31 MIA	Maximum I/O Address Capability. These bits define the maximum I/O bus address capability of this PAMU implementation. The MIA identifies the maximum value of the 32 most-significant address bits above the 32 least-significant address bits representing 4 Gbytes. This PAMU implements only the least significant MIA number of bits of the WBA field within a PAACE. **NOTE:** In no-address-translation mode, the I/O bus address of the DSA operation is also the system storage address. Since there is no address translation, the DSA Window must be contained within the system storage space. 0x0000_00FF Maximum 40-bit I/O bus address space

11.4.15 PAMU Operation Error Status register 1 (PAMUx_POES1)

Address: Base address + 40h offset



PAMUx_POES1 field descriptions

Field	Description
0–26 -	This field is reserved. Reserved
27–30 POET	PAMU Operation Error Type These bits identify the type of error encountered by the PAMU. The bits are coded as follows: 0000 PAMU fetch: bad data received 0001-0111 Reserved 1000 Unsupported ATM code requested in PAACE

Table continues on the next page...

PAMUx_POES1 field descriptions (continued)

Field	Description
	1001 Unsupported OTM code requested in PAACE 1010-1111 Reserved
31 POED	PAMU Operation Error Detected This bit is set when a hardware error is encountered by PAMU. If the bit is set, the rest of the contents of the Error Status register and the contents of the Error Status and Error Address registers are valid. It is coded as follows: 0 No PAMU error 1 PAMU error detected. Software should write a 1 to clear this bit.

11.4.16 PAMU Operation Error Status register 2 (PAMUx_POES2)

Address: Base address + 44h 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																															
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

PAMUx_POES2 field descriptions

Field	Description
0–31 -	This field is reserved. Reserved

11.4.17 PAMU Operation Error Address High register (PAMUx_POEAH)

Address: Base address + 48h 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	POEAH																															
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

PAMUx_POEAH field descriptions

Field	Description
0–31 POEAH	PAMU Operation Error Address High The POEAH is the 32 most significant address bits that identify the address, in system memory space, of the PAACT, OMT location that encountered the error condition. **NOTE:** POEAH[0:27] has a value of 0x000_0000 and is consistent with the MSA capability field as described in PAMU Address Capabilities Register 1.

11.4.18 PAMU Operation Error Address Low register (PAMUx_POEAL)

Address: Base address + 4Ch offset

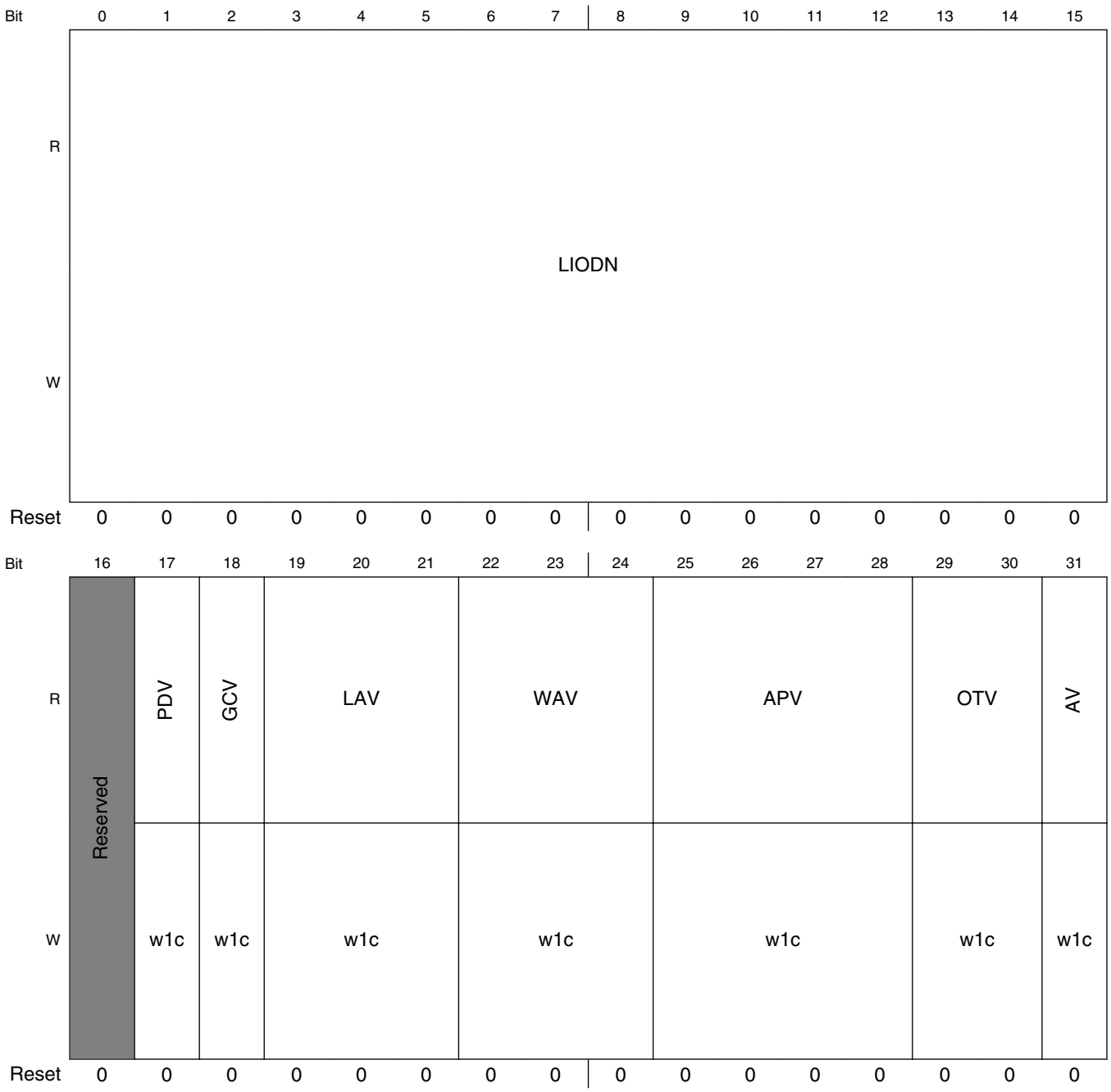
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	POEAL																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PAMUx_POEAL field descriptions

Field	Description
0–31 POEAL	PAMU Operation Error Address Low These bits define the address, in system memory space, of the PAACT or OMT location that encountered the error condition. The POEAL is the 32 least significant address bits of the address, in system memory space, that encountered the PAMU error. The granularity of the POEAL varies depending on the type of data structure that encountered the error: • For a PAACT location, the 6 least significant address bits of the POEAL has a value of 0x00, representing the PAACE size.

11.4.19 Access Violation Status register 1 (PAMUx_AVS1)

Address: Base address + 50h offset



PAMUx_AVS1 field descriptions

Field	Description
0–15 LIODN	Logical I/O Device Number These bits identify the LIODN of the operation that encountered the error.

Table continues on the next page...

PAMUx_AVS1 field descriptions (continued)

Field	Description
	NOTE: LIODN[0:3] has a value of 0x0 to be consistent with the ML capability field as described in PAMU Capabilities 2.
16 - Reserved	This field is reserved.
17 PDV	PAMU Disable Violation This bit identifies an access violation while the PAMU enable (PE) bit is not set (and the PAMU gate closed signal is deasserted). 0 No access violation while the PAMU enable (PE) bit is not set (and the PAMU gate closed signal is deasserted) 1 Access violation while the PAMU enable (PE) bit is not set (and the PAMU gate closed signal is deasserted)
18 GCV	Gate Closed Violation This bit identifies an access violation during assertion of PAMU gate closed bit. 0 No access violation during PAMU gate closed 1 Access violation during PAMU gate closed
19–21 LAV	LIODN Access Violation These bits identify if a LIODN access violation has occurred or the type of LIODN access violation encountered. 000 No LIODN access violation 001 LIODN access violation-LIODN index not within PPAACT 010 LIODN access violation-Secondary PAACE index not within SPAACT 011-111 reserved
22–24 WAV	Window Access Violation These bits identify if a window access violation has occurred or the type of window access violation encountered by the logical device. 000 No window access violation 001 Window access violation-Address not within valid window 010 Sub-window access violation-Address not within valid sub-range of the sub-window 011 Sub-window access violation-LIODN does not match secondary PAACE 100-111 Reserved
25–28 APV	Access Permission Violation These bits identify if an access permission violation has occurred or the type of access permission violation encountered by the logical device. **NOTE:** The APV encoding of 0001 can only be used when the MW field of the relevant PAACE is not set. Similarly, the APV encoding of 0010 can only be used when the MW field of the relevant PAACE is set. 0000 No access permission violation 0001 Window access permission violation-access denied to the DSA window 0010 Sub-window access permission violation-access denied to the DSA sub-window 0011 Reserved 0100 Window access type violation-illegal access type to the DSA window 0101 Sub-window access type violation-illegal access type to the DSA sub-window

Table continues on the next page...

PAMUx_AVS1 field descriptions (continued)

Field	Description
	0110-0111 Reserved 1000 Sub-window Access Type Violation - Access to a DSA sub-window with invalid SPAACE 1001-1111 Reserved
29–30 OTV	Operation Type Violation These bits identify if an operation type violation has occurred or the type of operation type violation encountered by the logical device. 00 No operation type violation 01 Immediate operation translation violation-IOE does not match IOA or IOB 10 Immediate operation translation violation-matched MOE valid bit not set (MOEA[0]=0 or MOEB[0]=0) 11 Indexed operation translation violation (MOEn[0]=0)
31 AV	Access Violation This bit is set when an access violation has been detected by PAMU. If the bit is set, the rest of the contents of the access violation register and the contents of the access violation address registers are valid. 0 No access violation 1 Access violation detected.

11.4.20 Access Violation Status register 2 (PAMUx_AVS2)

Address: Base address + 54h 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							OIV	IOE							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PAMUx_AVS2 field descriptions

Field	Description
0–22 -	This field is reserved. Reserved
23 OIV	Operation Index Violation OMI in PAACE refers to Operation Mapping Entry (OME) beyond the OMT limit address
24–31 IOE	Ingress Operation Encoding These bits identify the ingress operation encoding of the operation that encountered the error.

11.4.21 Access Violation Address High register (PAMUx_AVAH)

Address: Base address + 58h 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	AVAH																															
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

PAMUx_AVAH field descriptions

Field	Description
0–31 AVAH	Access Violation Address High The AVAH is the 32 most significant address bits that identify the address, in I/O address space, of the operation that encountered the error condition.

11.4.22 Access Violation Address Low register (PAMUx_AVAL)

Address: Base address + 5Ch 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	AVAL																				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

PAMUx_AVAL field descriptions

Field	Description
0–19 AVAL	Access Violation Address Low AVAL is the 20 next-significant of the 64 bits that identify the address, in I/O address space, of the operation that encountered the error condition.
20–31 -	This field is reserved. Reserved. The 12 of the 64 bits that identify the address, in I/O address space, of the operation that encountered the error condition are not reported.

11.4.23 ECC Error Control Register (PAMUx_EECTL)

Address: Base address + 60h 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								THRESH								Reserved								COUNT							
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

PAMUx_EECTL field descriptions

Field	Description
0–7 -	This field is reserved. Reserved
8–15 THRESH	PAMU Threshold Threshold value for the number of ECC single-bit error that are detected before reporting an error condition. After setting the threshold value, single bit ECC error reporting has to be enabled to report an error condition. If single bit ECC error reporting is enabled with a threshold value of 0, an error condition is reported on the first occurrence of single bit ECC error.
16–23 -	This field is reserved. Reserved
24–31 COUNT	PAMU Count Counts ECC single-bit errors detected. If COUNT is greater than or equal to THRESH then an error is reported, provided single-bit error reporting is enabled. Single bit ECC error is counted across a cache line, as a result one cache access can lead to only one single bit ECC error. Primary PAACT and secondary PAACT accesses in the same cycle can lead to a COUNT update by more than 1. Software must write zeros in bits 24-31 in order to clear the COUNT bits.

11.4.24 ECC Error Interrupt Enable Register (PAMUx_EEINTEN)

Address: Base address + 68h 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												MBECCINTEN	SBECCINTEN	Reserved	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PAMUx_EEINTEN field descriptions

Field	Description
0–27 -	This field is reserved. Reserved
28 MBECCINTEN	Multiple-bit ECC error reporting enable. 0 Multiple-bit ECC error reporting disabled. 1 Multiple-bit ECC error reporting enabled.

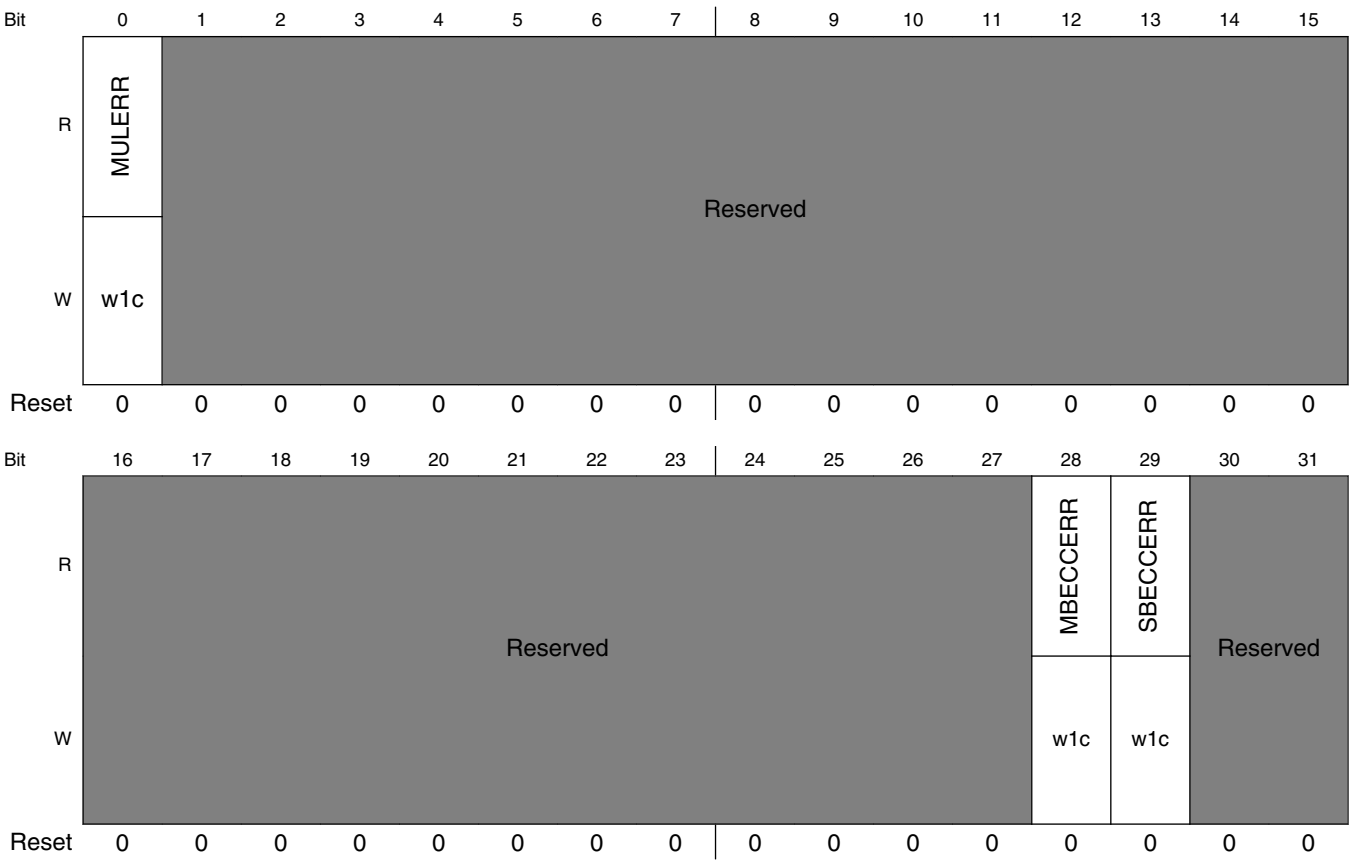
Table continues on the next page...

PAMUx_EEINTEN field descriptions (continued)

Field	Description
29 SBECCINTEN	Single-bit ECC error reporting enable. 0 Single-bit ECC error reporting disabled. 1 Single-bit ECC error reporting enabled.
30–31 -	This field is reserved. Reserved

11.4.25 ECC Error Detect Register (PAMUx_EEDET)

Address: Base address + 6Ch offset



PAMUx_EEDET field descriptions

Field	Description
0 MULERR	Multiple errors. This bit is set, when multiple-bit ECC error is set and PAMU gets another multiple-bit ECC error or single-bit ECC error is set and PAMU gets another single-bit ECC error. The bit does not get set for a Primary PAACT and secondary PAACT access in the same cycle and both accesses have same type of errors. 0 Multiple errors of the same type (Multiple-bit ECC/single-bit ECC) were not detected 1 Multiple errors of the same type (Multiple-bit ECC/single-bit ECC) were detected.

Table continues on the next page...

PAMUx_EEDET field descriptions (continued)

Field	Description
1–27 -	This field is reserved. Reserved
28 MBECCERR	Multiple-bit ECC error 0 Multiple-bit ECC error was not detected. 1 Multiple-bit ECC error was detected.
29 SBECCERR	Single-bit ECC error 0 Single-bit ECC error was not detected 1 Single-bit ECC error was detected.
30–31 -	This field is reserved. Reserved

11.4.26 ECC Error Attributes Register (PAMUx_EEATTR)

If ECC errors are detected for the primary PAACT cache and secondary PAACT cache in the same cycle, the primary PAACT cache information is captured in the EEATTR, EEAHI, EEALO, EEDHI, and EEDLO registers.

Address: Base address + 70h offset

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EELIODN															
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	EEPEL		DWNUM			Reserved										VALINFO
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PAMUx_EEATTR field descriptions

Field	Description
0–15 EELIODN	Logical I/O Device Number These bits identify the LIODN of the operation that encountered the ECC error. **NOTE:** LIODN[0:3] has a value of 0x0 to be consistent with the ML capability field as described in PAMU Capabilities 2.
16–17 EEPEL	PAMU Error Location. These bits identify the location of the error encountered by the PAMU. For cases, where an error is detected in the primary PAACT cache and secondary PAACT cache in the same cycle, EEPEL will be set to 00. 00 Error detected in primary PAACT cache 01 Error detected in secondary PAACT cache 10 Reserved 11 Error detected in OMT cache
18–20 DWNUM	Double word number. Indicates the double-word number within the cache line where the ECC error was detected. Example: 0x1: Indicates double word 1
21–30 -	This field is reserved. Reserved
31 VALINFO	PAMU ECC error attribute register valid 0 PAMU ECC error attribute register contains no valid information or no enabled errors were detected. 1 PAMU ECC error attribute register contains information of the first detected error that has reporting enabled. Software must clear this bit to unfreeze error capture so error detection hardware can overwrite the capture address/data/attributes for a newly detected error.

11.4.27 ECC Error Address High (PAMUx_EEAHI)

If ECC errors are detected for the primary PAACT cache and secondary PAACT cache in the same cycle, the primary PAACT cache information is captured in the EEATTR, EEAHI, EEALO, EEDHI, and EEDLO registers.

Address: Base address + 74h 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	ECCEAH																															
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

PAMUx_EEAHI field descriptions

Field	Description
0–31 ECCEAH	ECC Error Address High The ECCEAH is the 32 most significant address bits that identify the address that encountered an ECC error while reading from the cache.

11.4.28 ECC Error Address Low (PAMUx_EEALO)

If ECC errors are detected for the primary PAACT cache and secondary PAACT cache in the same cycle, the primary PAACT cache information is captured in the EEATTR, EEAHI, EEALO, EEDHI, and EEDLO registers.

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	ECCEAL																										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

PAMUx_EEALO field descriptions

Field	Description
0–25 ECCEAL	ECC Error Address Low ECCEAL is the next 26 least significant address bits that identify the address that encountered an ECC error while reading from the cache.
26–31 -	This field is reserved. Reserved

11.4.29 ECC Error Data High (PAMUx_EEDHI)

If ECC errors are detected for the primary PAACT cache and secondary PAACT cache in the same cycle, the primary PAACT cache information is captured in the EEATTR, EEAHI, EEALO, EEDHI, and EEDLO registers.

Address: Base address + 7Ch 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	DATAHI																															
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

PAMUx_EEDHI field descriptions

Field	Description
0–31 DATAHI	ECC Error Data High High-order bits for data presenting an ECC error. Data[0:31].

11.4.30 ECC Error Data Low (PAMUx_EEDLO)

If ECC errors are detected for the primary PAACT cache and secondary PAACT cache in the same cycle, the primary PAACT cache information is captured in the EEATTR, EEAHI, EEALO, EEDHI, and EEDLO registers.

Address: Base address + 80h 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	DATALO																															
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

PAMUx_EEDLO field descriptions

Field	Description
0–31 DATALO	ECC Error Data Low Low-order bits for data presenting an ECC error. Data[32:63].

11.4.31 Unauthorized device access detection register (PAMUx_UDAD)

Address: Base address + 90h offset

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	LIODN															
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															UDAD
W																w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PAMUx_UDAD field descriptions

Field	Description
0–15 LIODN	Logical I/O device number. These bits identify the LIODN of the operation that encountered the unauthorized device access. **NOTE:** LIODN[0:3] has a value of 0x0 to be consistent with the ML capability field as described in PAMU Capabilities 2.
16–30 -	This field is reserved. Reserved
31 UDAD	Unauthorized device access detected. This bit is set when an unauthorized device access has been detected by the PAMU (LIODN index not pointing to a valid PPAACE). If the bit is set, the rest of the contents of the Unauthorized Device Access Detection register are valid. 0 No Unauthorized Device Access Detected 1 Unauthorized Device Access Detected.Software must write 1 in order to clear the pin.

11.4.32 PAMU Revision register 1 (PAMUx_PR1)

Address: Base address + BF8h 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	IPID																IPMJ								IPMN							
W																																
Reset	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

PAMUx_PR1 field descriptions

Field	Description
0–15 IPID	IP block ID. These bits provide the IP block ID. For PAMU, the value is 0x0901.
16–23 IPMJ	Major revision. These bits provide the major revision number for PAMU.
24–31 IPMN	Minor revision. These bits provide the minor revision number for PAMU.

11.4.33 PAMU Revision register 2 (PAMUx_PR2)

Address: Base address + BFCh 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								PIO								PER								PCO							
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

PAMUx_PR2 field descriptions

Field	Description
0–7 -	This field is reserved. Reserved
8–15 PIO	PAMU Integration Options. These bits provide information on the integration options for PAMU.
16–23 PER	PAMU ECO Revision. These bits provide information on the ECO Revision Number for PAMU.
24–31 PCO	PAMU Configuration Options. These bits provide information on the configuration options for PAMU.

11.4.34 PAMU Capabilities register 1 (PAMUx_PC1)

The fields in the PAMU capabilities registers denote implementation parameters for a given hardware instance of PAMU. Software must read these registers to discover the specific parameter values implemented in this instance of PAMU and utilize the information when programming the PAMU, its data structures, and related system resources.

Address: Base address + C00h 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																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

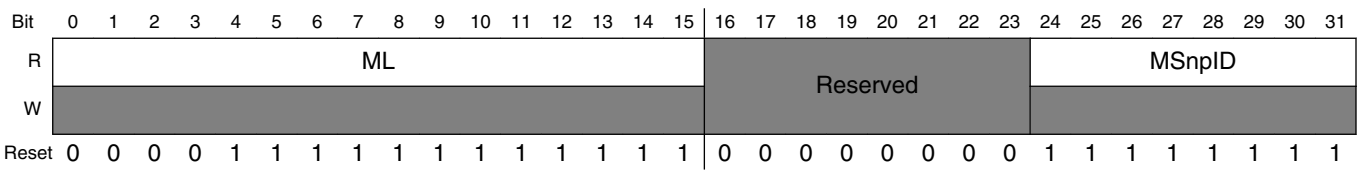
PAMUx_PC1 field descriptions

Field	Description
0–31 -	Reserved

11.4.35 PAMU Capabilities register 2 (PAMUx_PC2)

The fields in the PAMU capabilities registers denote implementation parameters for a given hardware instance of PAMU. Software must read these registers to discover the specific parameter values implemented in this instance of PAMU and utilize the information when programming the PAMU, its data structures, and related system resources.

Address: Base address + C04h offset



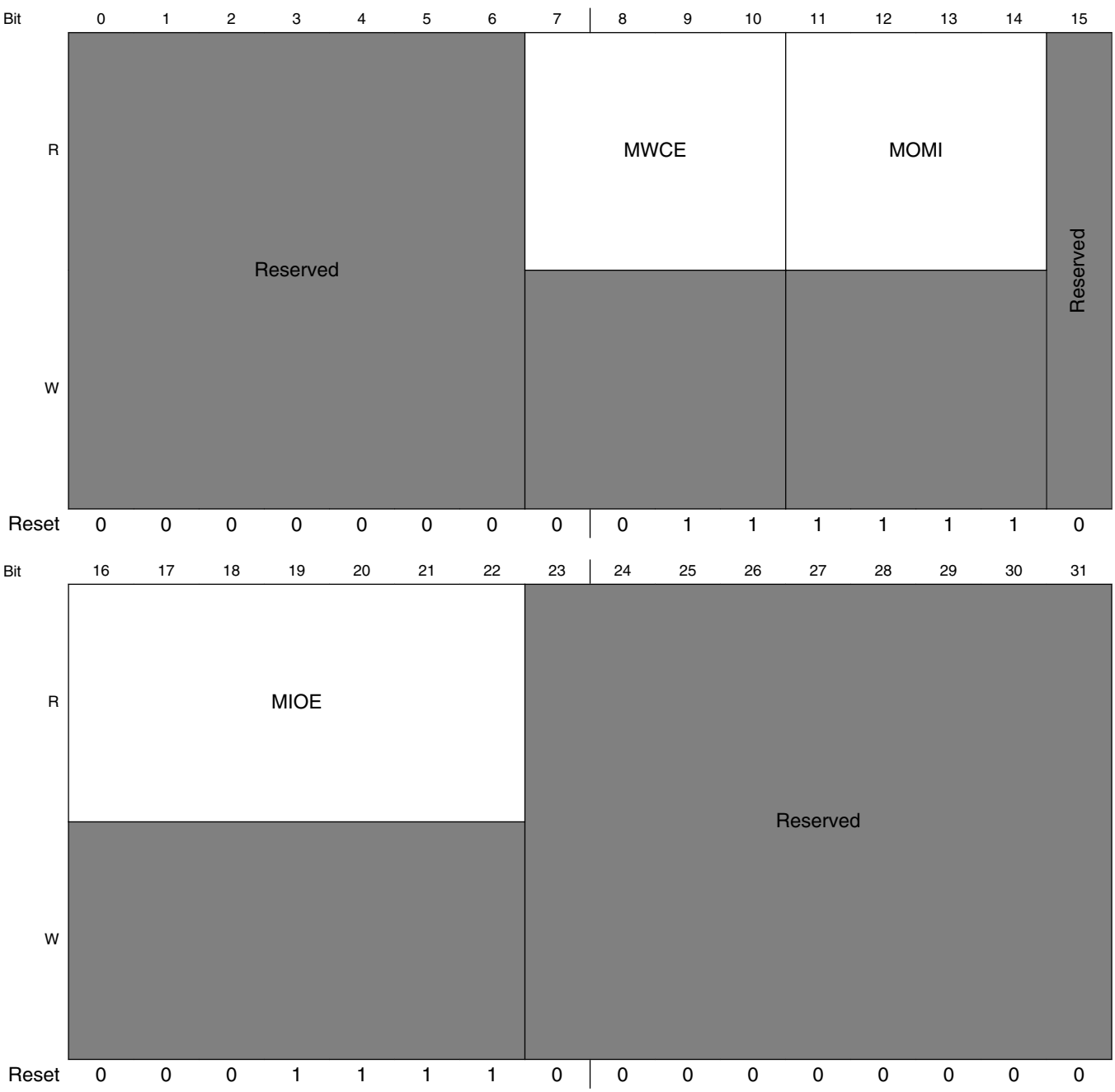
PAMUx_PC2 field descriptions

Field	Description
0–15 ML	Maximum LIODN. These bits define the maximum number of bits of LIODN supported by this PAMU instance. This field is specified as a 16-bit vector, with the sub-vector of the supported number of bits set to 1. Examples: 0x00FF Maximum 8-bit LIODN capability 0x0FFF Maximum 12-bit LIODN capability
16–23 -	This field is reserved. Reserved
24–31 MSnpID	Maximum Snoop ID. These bits define the maximum Snoop ID capability of the system or the maximum Snoop ID that PAMU can signal to the Host/Coherency Domain. This field is specified as a 8-bit vector, with the sub-vector of the supported number of bits set to 1. The bits are coded as follows: 0x0F Maximum 4-bit Snoop ID or the capability for up to 16 unique Snoop IDs 0x1F Maximum 5-bit Snoop ID or the capability for up to 32 unique Snoop IDs 0xFF Maximum 8-bit Snoop ID or the capability for up to 256 unique Snoop IDs

11.4.36 PAMU Capabilities register 3 (PAMUx_PC3)

The fields in the PAMU capabilities registers denote implementation parameters for a given hardware instance of PAMU. The software must read these registers to discover the specific parameter values implemented in this instance of PAMU and utilize the information when programming PAMU, its data structures, and related system resources.

Address: Base address + C08h offset



PAMUx_PC3 field descriptions

Field	Description
0–6 -	This field is reserved. Reserved
7–10 MWCE	Maximum Window Count Encoding. These bits define the maximum number of DSA sub-windows available to a LIODN. The window count identifies the maximum number of secondary PAACEs that the PAMU accesses to perform authorization and access control services. The maximum window count is $2^{(MWCE+1)}$ windows. **NOTE:** MWCE affects the maximum value of the WCE field in a PAACE that can be handled by this PAMU instance. Example: 0x3 Maximum 16 window count capability
11–14 MOMI	Maximum Operation Mapping Index. These bits define the number of least significant bits of the OMI field in a PAACE that are accessed and used as OMT index by this instance of PAMU. This field is specified as a 4-bit vector, with the sub-vector of the supported number of bits set to 1. The field thus defines the maximum number of operation mapping entries available to a LIODN for indexed operation type translation. **NOTE:** MOMI affects the maximum index value of the OMI field of a PAACE that can be handled by this PAMU instance. The bits are coded as follows: 0001 Indicates a maximum 2 OME capability 0011 Indicates a maximum 4 OME capability 0111 Indicates a maximum 8 OME capability 1111 Indicates a maximum 16 OME capability
15 -	This field is reserved. Reserved
16–22 MIOE	Maximum Ingress Operation Encoding. These bits define the number of least significant bits of an ingress operation encoding (IOE) that are used as an index into an OME to obtain an EOE mapping by this instance of PAMU. This field is specified as a 7-bit vector, with the sub-vector of the supported number of bits set to 1. **NOTE:** The field defines the maximum number of IOEs available to an LIODN for operation type translation through this PAMU instance. Example: 0x0F Indicates a maximum 16 IOE capability
23–31 -	This field is reserved. Reserved

11.4.37 PAMU Capabilities register 4 (PAMUx_PC4)

The fields in the PAMU capabilities registers denote implementation parameters for a given hardware instance of PAMU. The software must read these registers to discover the specific parameter values implemented in this instance of PAMU and utilize the information when programming PAMU, its data structures, and related system resources.

Address: Base address + C0Ch offset

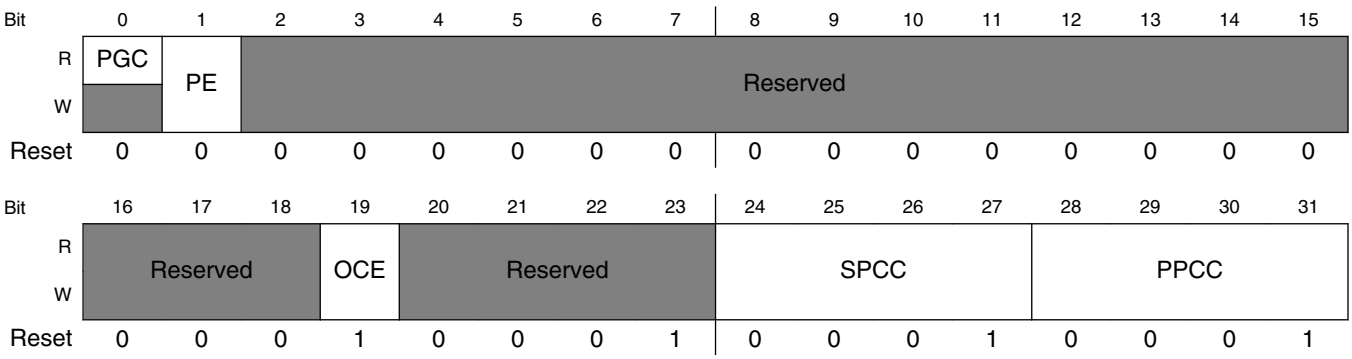
Bit	0–19	20–23	24–27	28–31
R	Reserved	ATMS	OTMS	ALS
W				
Reset	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	1 1 0 0	1 1 1 0	0 0 0 1

PAMUx_PC4 field descriptions

Field	Description
0–19 -	This field is reserved. Reserved
20–23 ATMS	ATM Support. These bits identify the address translation mode(s) supported by this instance of PAMU. The bits are coded as follows: Bit 20, when set, indicates that no address translation mode is supported by this PAMU. Bit 21, when set, indicates that window address translation mode is supported by this PAMU. Bits 22-23 are reserved. Example: 1100 Indicates PAMU support for the following address translation modes: No Address Translation. Window Address Translation.
24–27 OTMS	OTM Support. These bits identify the operation translation mode(s) supported by this instance of PAMU. The bits are coded as follows: Bit 24, when set, indicates that no operation translation mode is supported by this PAMU. Bit 25, when set, indicates that immediate operation translation mode is supported by this PAMU. Bit 26, when set, indicates that indexed operation translation mode is supported by this PAMU. Bit 27 is reserved. Example: 1110 Indicates PAMU support for the following operation translation modes: No Operation Translation Immediate Operation Translation Indexed Operation Translation
28–31 ALS	Architecture Level Support. These bits identify the version(s) of the PAMU architecture supported by this revision of the PAMU implementation. 0001 Indicates PAMU support for PAMU architecture Version 1.0 only

11.4.38 PAMU Control register (PAMUx_PC)

Address: Base address + C10h offset



PAMUx_PC field descriptions

Field	Description
0 PGC	PAMU Gate Closed 0 PAMU gate open. PAMU may allow peripheral accesses, subject to authorization and access control. 1 PAMU gate closed. PAMU blocks all peripheral accesses.
1 PE	PAMU Enable This bit controls whether PAMU performs authorization and translation functions. This is to be set by software after setting up the authorization and translation data structures, PAMU and devices (including the setting of LIODNs). **NOTE:** • If PGC is set, or if PE is cleared, all peripheral accesses are blocked. • If PGC is cleared and PE is set, peripheral accesses are allowed based on the LIODN signaled with the access capabilities permitted for the LIOD. 0 PAMU Disabled 1 PAMU Enabled
2–18 -	This field is reserved. Reserved
19 OCE	OMT Cache Enable These bits define the OMT cache enable bit. 0 OMT cache disabled 1 OMT cache enabled
20–23 -	This field is reserved. Reserved
24–27 SPCC	Secondary PAACT Cache Control These bits, SPCC[0:3], define the properties of the secondary PAACT cache. SPCC[0:2] are reserved. SPCC[3] is the cache enable bit.

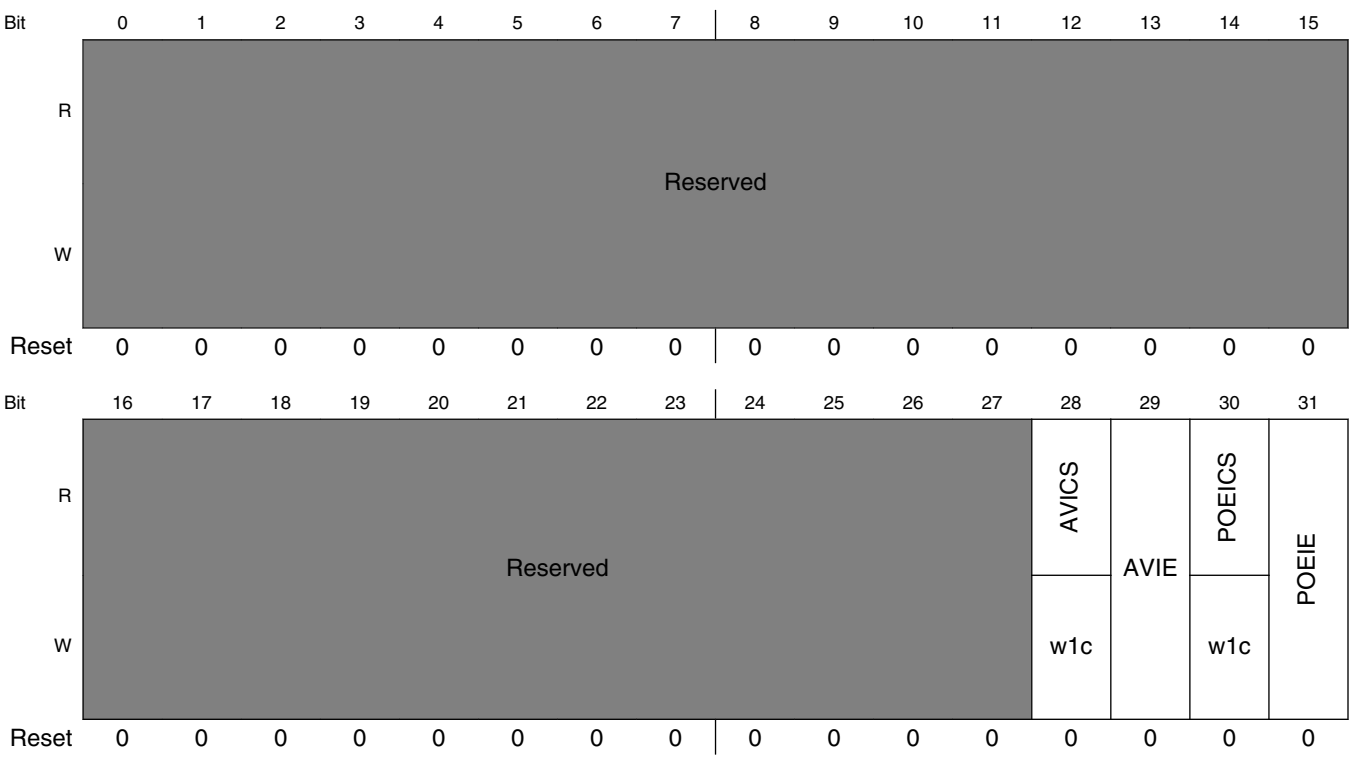
Table continues on the next page...

PAMUx_PC field descriptions (continued)

Field	Description
	0 Secondary PAACT cache disabled 1 Secondary PAACT cache enabled
28–31 PPCC	Primary PAACT Cache Control These bits, PPCC[0:3], define the properties of the primary PAACT cache. PPCC[0:2] are reserved. PPCC[3] is the cache enable bit. 0 Primary PAACT cache disabled 1 Primary PAACT cache enabled

11.4.39 PAMU Interrupt Control and Status register (PAMUx_PICS)

Address: Base address + C1Ch offset



PAMUx_PICS field descriptions

Field	Description
0–27 -	This field is reserved. Reserved
28 AVICS	Access Violation Interrupt Control and Status This bit provides status on whether the access violation interrupt pin is asserted and controls whether the access violation interrupt pin is to be deasserted.

Table continues on the next page...

PAMUx_PICS field descriptions (continued)

Field	Description
	0 Access violation interrupt pin is deasserted 1 Access violation interrupt pin is asserted.
29 AVIE	Access Violation Interrupt Enable This bit identifies if an access violation interrupt is to be asserted. 0 No interrupt asserted if an access violation is detected 1 Access violation interrupt pin asserted if an access violation is detected
30 POEICS	PAMU Operation Error Interrupt Control and Status This bit provides status on whether the operation error interrupt pin is asserted and controls whether the operation error interrupt pin is to be deasserted. 0 Operation error interrupt pin is deasserted 1 Operation error interrupt pin is asserted.
31 POEIE	PAMU Operation Error Interrupt Enable This bit identifies if an operation error interrupt is to be asserted. 0 No interrupt asserted if an operation error is detected 1 Operation error interrupt pin asserted if an operation error is detected

11.5 PAMU Functional Description

The following is a brief description of the set-up requirements and sequence of events involving processing of DSA operations by PAMU.

11.5.1 System Set-Up for PAMU Operation

System software, for example, a Hypervisor, enumerates these physical I/O devices with one or more Logical I/O Device Numbers (LIODN).

Hypervisor then creates PPAACT and if necessary SPAACT and OMT in system memory (see Data Structures Used by PAMU).

After setting up the data structures, PAMU CCSRs are programmed, including the locations of the above data structures. After a PAMU is programmed, it is enabled to process DSA operations.

NOTE

Depending on reset configuration, the PAMUs may default to PAMU bypass mode. The PAMU bypass enable register in the device configuration module (DCFG_PAMUBYPENR) controls enabling and disabling PAMU bypass mode for each

PAMU. PAMU bypass mode must be disabled for PAMU to authorize and translate transactions. See PAMU Bypass Mode and No Operation Translation Mode for more information about PAMU bypass mode and PAMUBYPENR for more information about disabling PAMU bypass mode.

11.5.2 Steps in Processing of DSA Operations by PAMU

Figure 11-160 illustrates the data structures and actions associated with processing of a DSA operation by PAMU.

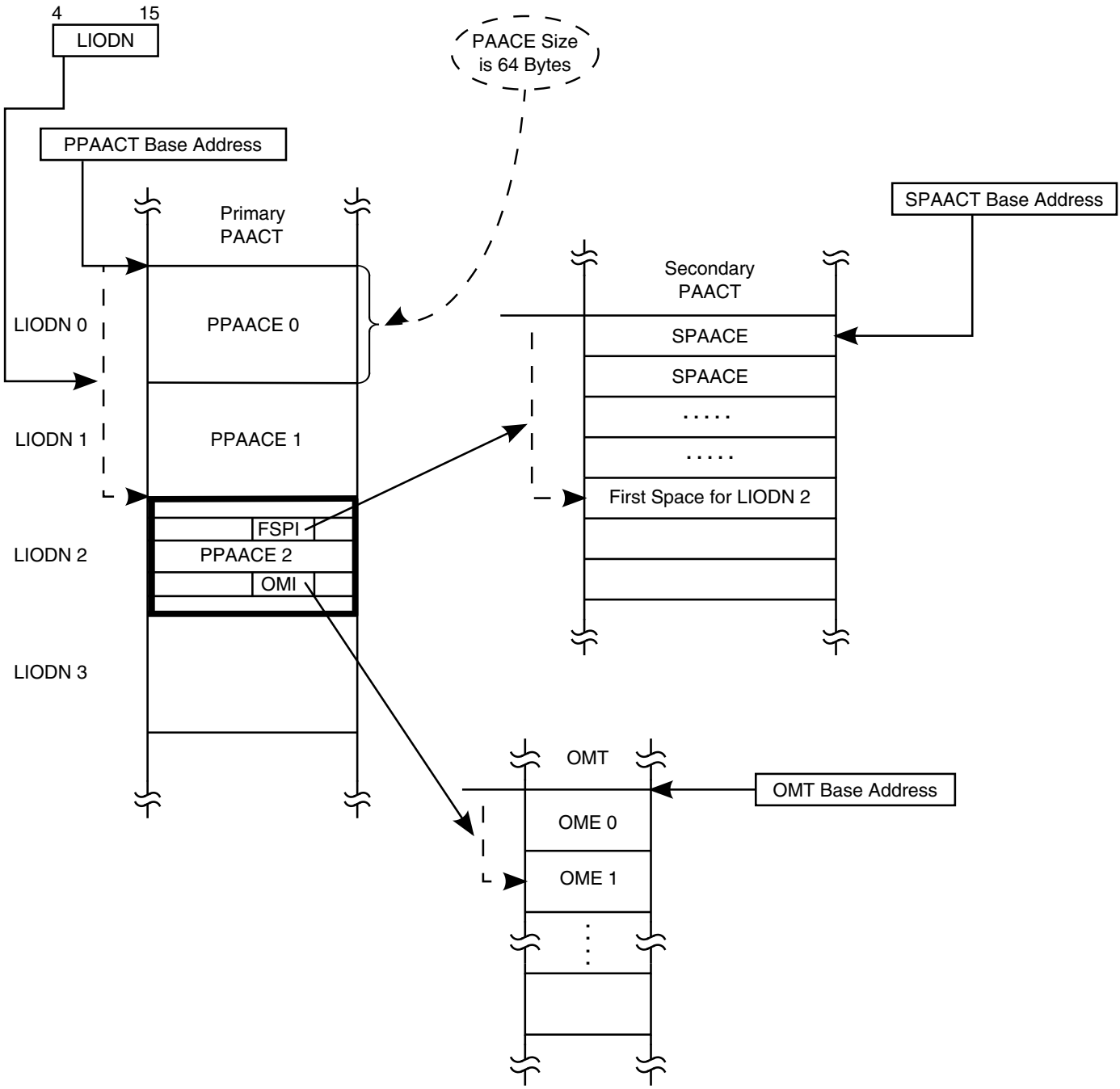


Figure 11-160. PAMU Operation flow overview

- An operation from a logical I/O device arrives at the entrance of the coherency domain. The operation carries with it the LIODN representing that logical device.
- PAMU uses the LIODN to look up the relevant PPAACE in the PPAACT cache.
- In the case of a cache-miss, PAMU will fetch the necessary data structure from the PPAACT in system memory.

- The contents of the PPAACE indicate if additional data structures have to be referenced to obtain all the necessary authorization and translation attributes for the operation by that LIODN to a particular DSA window.
- A secondary PAACE is accessed if the address accessed is beyond the primary sub-window.
- If the access falls beyond the windows assigned to the LIOD, it is considered an access violation.
- An OME is accessed optionally based on the value of the OTM. The base address of the operation mapping table (OMT) and the operation mapping index (OMI) contained in the PAACE are used to access the entry.
- The translated transaction type and operation is returned for forwarding to the destination domain.

The next section describes the above steps in greater detail.

11.5.3 Detailed Description of PAMU Actions

This section describes in detail the handling of DSA operations by PAMU.

11.5.3.1 PAMU Gate Closed and PAMU Enable Check

In secure boot mode, or if the PAMU Gate Closed (PGC) bit is set to 1, the PAMU will block all peripheral accesses through assertion of access violation regardless of the PAMU Enable (PE) bit setting and regardless of the signaled LIODN.

- If a PAMU lookup request were to be issued while the PGC bit is set to 1, an access violation has been detected and bits are set in the relevant CCSRs (see PAMU Operation Error Address High register (PAMU_POEAH), through Access Violation Address Low register (PAMU_AVAL)).
  - The access violation (AV) detected bit is set.
  - The Gate Closed Violation (GCV) bit is set to indicate the peripheral access is not allowed.
  - The LIODN and DSA Address encountering the access violation is logged.

If the PGC bit is not asserted and the PE bit is not set, the PAMU will block all peripheral accesses regardless of the signaled LIODN.

If the PGC bit is not asserted and the PE bit is set, the PAMU will provide authorization and translation services for all peripheral accesses based on the signaled LIODN. Description of this flow is detailed in subsequent sections, starting with section PPAACT Request Range Check. This setting is recommended once all authorization and translation data structures have been set up in system memory space.

If secure boot and trust architecture is not enabled, the PAMU Gate Closed (PGC) bit can be cleared.

Table 11-164. PAMU Gate Closed and PAMU Enable Check

PAMU Gate Closed	PAMU Enable	PAMU Authorizing	Comments
0	0	Yes	Access Violation
0	1	Yes	Normal Operation
1	X	Yes	Access Violation

11.5.3.2 PPAACT Request Range Check

The following describes the manner in which the PAMU lookup request is checked to see if it falls within a valid range:

- During a PAMU lookup request, the LIODN of the operation is signaled along with the rest of the information that describes the DSA operation (address, operation type, etc.).
- The PPAACT base address programmed in the PPBARs (see Primary PAACT Base Address High register (PAMU_PPBAH), and Primary PAACT Base Address Low register (PAMU_PPBAL)) along with the LIODN as offset is used to generate the PPAACE address.
- The PPAACE address is checked against the PPAACT limit address programmed in the PPLARs (see sections Primary PAACT Limit Address High register (PAMU_PPLAH), and Primary PAACT Limit Address Low register (PAMU_PPLAL)).
- A PPAACE address higher than the PPAACT limit address is due to an erroneously or illegally signaled LIODN. An access violation has been detected and bits are set in the relevant CCSRs (see Access Violation Status register 1 (PAMU_AVS1), through Access Violation Address Low register (PAMU_AVAL)):
  - The access violation (AV) detected bit is set.
  - The LIODN access violation (LAV) bits are set to indicate the LIODN index is not within the PPAACT.
  - The LIODN and DSA address encountering the access violation is logged.

11.5.3.3 PPAACT Cache Lookup

The following describes the flow of a PPAACT cache lookup and the manner in which the information is used by the cache:

- During a PAMU lookup request, the LIODN of the operation is signaled along with the rest of the information that describes the DSA operation (address, operation type, etc.).
  - The LIODN is used to index into the PPAACT cache.
- The PPAACT base address programmed in the PPBARs (see Primary PAACT Base Address High register (PAMU_PPBAH), and Primary PAACT Base Address Low register (PAMU_PPBAL)) along with the LIODN offset is used to generate the PPAACE address.
  - A match of the PPAACE address against the stored tag of a valid entry indicates a cache hit while a mismatch indicates a cache miss. If the tag does not match, a cache miss has occurred.
- If a cache-hit occurs, ECC bits are computed for the contents of the PAACE and these ECC bits are compared against the ECC bits stored along with the PAACE. Mismatched ECC bits, indicates an ECC error.

11.5.3.4 PAMU Fetch Request

Fetch requests are issued to retrieve the relevant PAACE or OME from system memory. These requests are also issued when cache misses occur.

11.5.3.5 Primary PAACE Processing

The primary PAACE is processed for authorization, access control and optionally, translation:

- The WBA and WSE fields of the PPAACE are used to determine if the address of the DSA operation is within a valid window in the I/O bus address space.
  - If the operation is not within a valid window, an access violation has been detected and several bits are set in the relevant CCSRs (see Access Violation Status register 1 (PAMU_AVS1), through Access Violation Address Low register (PAMU_AVAL)):
    - The access violation (AV) detected bit is set.

        - The window access violation (WAV) bits are set to indicate a window access violation has occurred.
        - The LIODN and DSA address encountering the access violation is logged.
- The MW field of the PPAACE is used to determine if DSA sub-windows exist.
    - The WCE field indicates the number of DSA sub-windows if DSA sub-windows exist.
- The least significant WSE+1 bits of the DSA address represent its offset (DSA window offset or DWO) within the LIOD's DSA window.
- The Sub-window index (SWI) is the most significant WCE+1 bits of the DWO.
- If the MW bit is set and if SWI = 0, then PPAACE is being accessed. In this case the SWSE field of the PPAACE is used to check if the DSA address is valid, that is, it falls within the defined sub-range.
    - If the address of the DSA operation is not within the sub-range, an access violation has been detected and bits are set in the relevant CCSRs (see Access Violation Status register 1 (PAMU_AVS1), through Access Violation Address Low register (PAMU_AVAL)):
        - The access violation (AV) detected bit is set.
        - The window access violation (WAV) bits are set to indicate a sub-window access violation has occurred.
        - The LIODN and DSA address encountering the access violation is logged.
    - If the address of a DSA operation is valid, the access permission (AP) field of the PPAACE is used to determine if the DSA operation is from a valid LIOD that has permission to issue transactions to system storage space. If the AP indicates access is denied or if the specific type of access is not permitted, bits are set in the relevant CCSRs (see Access Violation Status register 1 (PAMU_AVS1), through Access Violation Address Low register (PAMU_AVAL)):
        - The access violation (AV) detected bit is set.
        - The LIODN and DSA address encountering the access violation is logged.
        - The access permission violation (APV) bits are set to indicate the type of violation that has occurred:
            - If AP indicates that access is denied, either the DSA operation has signaled an invalid LIODN or this LIOD does not have access permissions to this system storage space, that is, the DSA window or sub-window has not been provided to the LIOD assigned the LIODN. The APV bits are set to indicate access has been denied to either the DSA window or sub-window.
            - The AP field of the entry is also used to determine if the type of operation is within the access privileges for that LIOD. Bit 0 of the IOE of the operation is compared against the permitted operation type(s) identified in the AP field. If the type of DSA operation is not permitted,

the APV bits are set to indicate an illegal access type to either the DSA window or sub-window.

- If SWI != 0, the PAACE being referenced is a SPAACE. See SPAACE Access and Processing.
- Details on how address translation is performed is detailed in Address Translation Service.
- Details on how the DSA operation type is processed is detailed in OMT Access and Service.

11.5.3.6 SPAACE Access and Processing

If SWI != 0, the DSA address indicates that the relevant PAACE is a SPAACE, and the PAMU will perform a lookup on the SPAACT as follows:

If the PPAACE indicates multiple window capability and the window count is n, then:

- The attributes for the DSA sub-window defined by addresses where SWI = 0 are contained in the PPAACE.
- The attributes for the DSA sub-window defined by addresses where SWI = 1 are contained in the SPAACE located at the FSPI of SPAACT.
- The attributes for the DSA sub-window defined by addresses where SWI = 2 are contained in the SPAACE located at the FSPI + 1.
- The attributes for the DSA sub-window defined by addresses where SWI = *n* - 1 (n > 1) are contained in the SPAACE located at the FSPI + (n-2).

See Figure 11-3 for an example illustration for n = 4.

The PAMU also performs an SPAACT range check based on the address of the SPAACE being accessed (see SPAACT Request Range Check).

11.5.3.6.1 SPAACT Request Range Check

The following describes the manner in which the SPAACE address is checked to see if it falls within a valid range:

- The SPAACE address is checked against the SPAACT limit address programmed in the SPLARs (see Secondary PAACT Limit Address High register (PAMU_SPLAH), and Secondary PAACT Limit Address Low register (PAMU_SPLAL)).
- A SPAACE address higher than the SPAACT limit address is due to an erroneous or illegal DSA address. An access violation has been detected and bits are set in the relevant CCSRs (see Access Violation Status register 1 (PAMU_AVS1), through Access Violation Address Low register (PAMU_AVAL)):
  - The access violation (AV) detected bit is set.

- The window access violation (WAV) bits are set to indicate a sub-window access violation has occurred.
- The LIODN and DSA address encountering the access violation is logged.

11.5.3.6.2 SPAACT Cache Lookup Request

SPAACT cache lookup requests are issued to retrieve the relevant SPAACE from the SPAACT cache.

- If the lookup results in a cache hit, the ECC status is checked.
  - If the ECC status indicates an error, bits are set in the relevant CCSRs (see ECC Error Control Register (PAMU_EECTL), through ECC Error Data Low (PAMU_EEDLO)).
- If the lookup response indicates a cache-miss, the SPAACE fetch request is issued.

11.5.3.6.3 Secondary PAACE Processing

- The SWSE field of the SPAACE is used to determine if the address of the DSA operation is within a valid sub-range in the I/O bus address space.
  - If the operation is not within the valid sub-range within the DSA sub-window, an access violation has been detected and bits are set in the relevant CCSRs (see Access Violation Status register 1 (PAMU_AVS1), through Access Violation Address Low register (PAMU_AVAL)):
    - The access violation (AV) detected bit is set.
    - The window access violation (WAV) bits are set to indicate a sub-window access violation has occurred.
    - The LIODN and DSA Address encountering the access violation is logged.
- If the address of a DSA operation is within the valid sub-range within the DSA sub-window, the Access Permission (AP) field of the SPAACE is used to determine if the DSA operation is from a valid LIOD that has permission to issue transactions to system storage space. If the AP indicates access is denied or if the specific type of access is not permitted, bits are set in the relevant CCSRs (see Access Violation Status register 1 (PAMU_AVS1), through Access Violation Address Low register (PAMU_AVAL)):
  - The access violation (AV) detected bit is set.
  - The LIODN and DSA Address encountering the access violation is logged.
  - The Access Permission Violation (APV) bits are set to indicate the type of violation that has occurred:

- If AP indicates that access is denied, this LIOD does not have access permissions to the DSA sub-window. The APV bits are set to indicate access has been denied to the DSA sub-window.
- The AP field of the entry is also used to determine if the type of operation is within the access privileges for that LIOD. IOE[0] of the operation is compared against the permitted operation type(s) identified in the AP field. If the type of DSA operation is not permitted, the APV bits are set to indicate an illegal access type to the DSA sub-window.

- Details on how address translation is performed is detailed in Address Translation Service.
- Details on how the DSA operation type is processed is detailed in OMT Access and Service.

11.5.3.7 Address Translation Service

The address of the DSA operation issued by the LIOD falls within the I/O address space. This section describes the mechanisms to direct DSA operations in I/O address space to the corresponding locations in system storage space.

When a valid PAACE is referenced for a DSA operation from a LIOD, the DSA address translation mode field of the PAACE is used to determine if the additional step of address translation is needed for that DSA operation. There can be two types of encodings indicated in the ATM field, as follows:

- No address translation mode
- Window address translation mode

11.5.3.7.1 No Address Translation Mode

If the address range of system storage space is equal to or less than the address range of the I/O address space, and a particular LIOD has the addressing capability to access the address range of system storage space, then the no address translation Mode may be enabled because the LIOD does not have any addressability limitations to locations in system storage space.

The following describes how no address translation mode is achieved for DSA operations:

- The I/O bus address of the DSA operation is also the system storage address.
- Because there is no address translation, the DSA window must be contained within the system storage space.
- Figure 11-161 shows an example of DSA operations with no address translation.

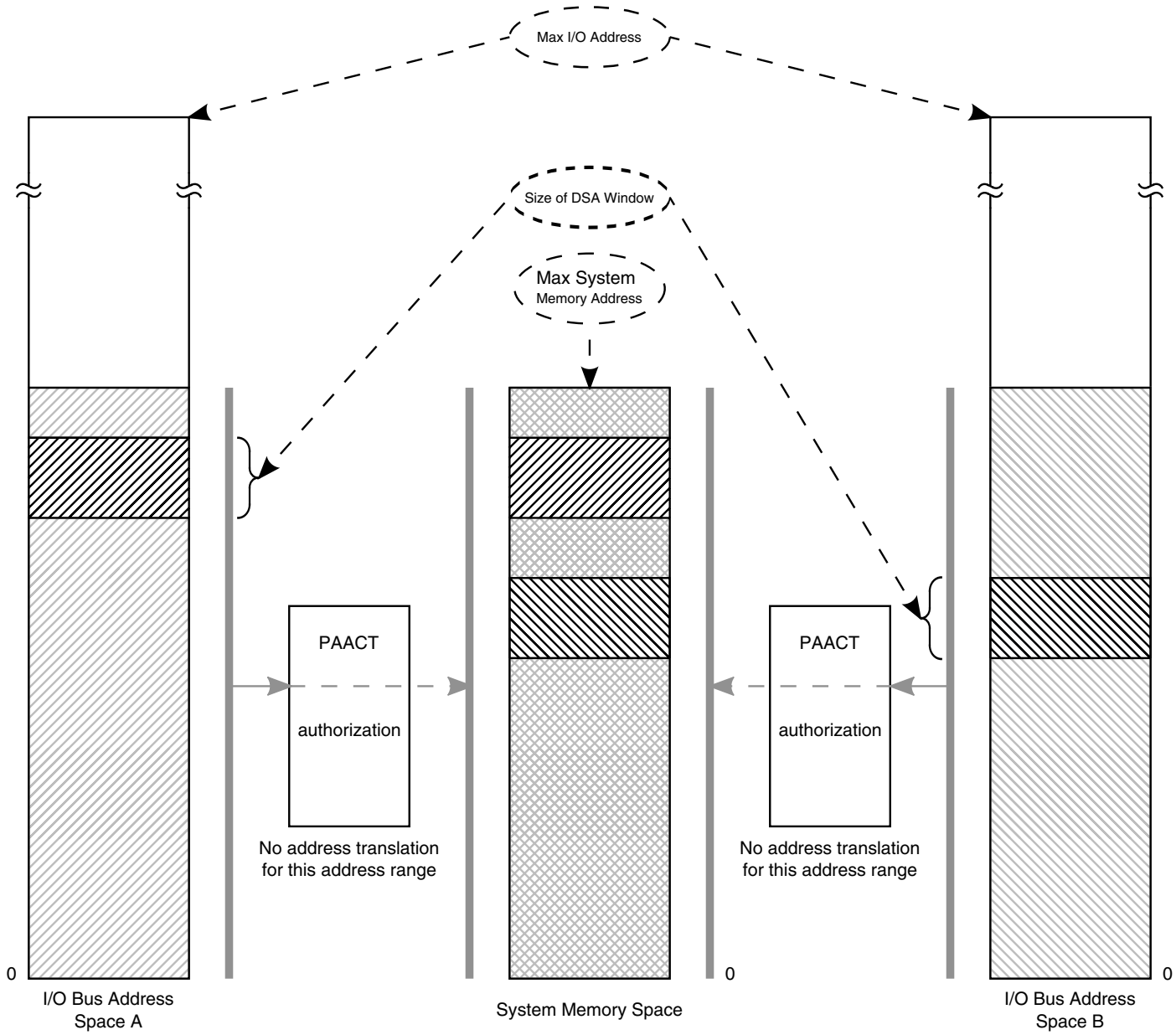


Figure 11-161. DSA Operations with No Address Translation

11.5.3.7.2 Window Address Translation Mode

Window address translation allows the capability for a LIOD to direct DSA operations to a contiguous DSA window in I/O bus address space that get translated to a contiguous window of the same size in system storage space. While no address translation mode implies that the system address is identical to the I/O bus address, with window address translation mode the DSA window can be located anywhere in the I/O bus address space including an address range where the addressing width of the I/O bus address space is larger than that defined for system storage space.

The following describes how DSA operations are achieved when window address translation is enabled:

- The TWBA field of the PAACE defines the base address of the DSA window in system storage space (see Figure 11-163). As mentioned in Table 11-6, TWBA is aligned to the size to the window described by the PAACE:
- If MW is disabled, the size of the window is indicated by the WSE field of the PAACE
- If MW is enabled, the size of the window is indicated by the SWSE field of the PAACE
- The window size is also used to determine which bits of the I/O bus address used in the DSA operation will be retained in the system storage address (also shown in Figure 11-163). because the minimum window size is 4 Kbytes, the 4-Kbyte address offset is always retained.
- Figure 11-163 shows an illustration of DSA operations with window address translation.

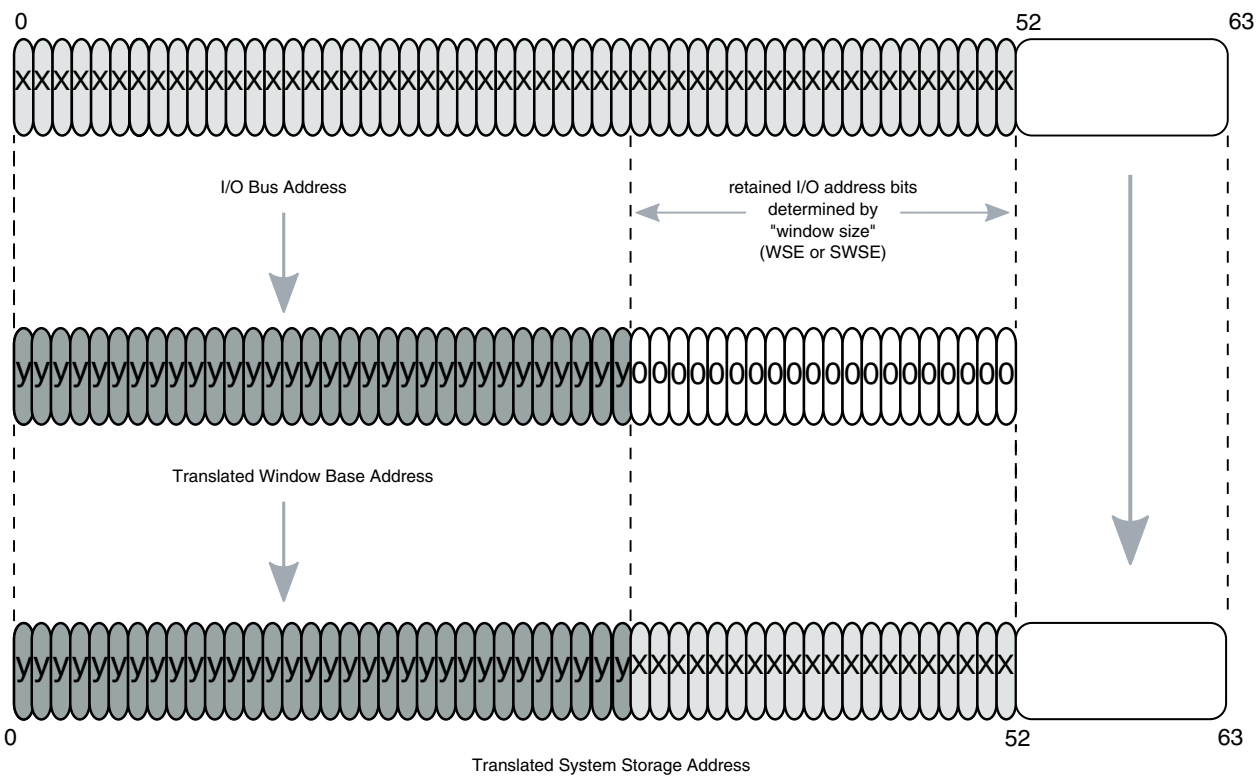


Figure 11-162. Window Address Translation

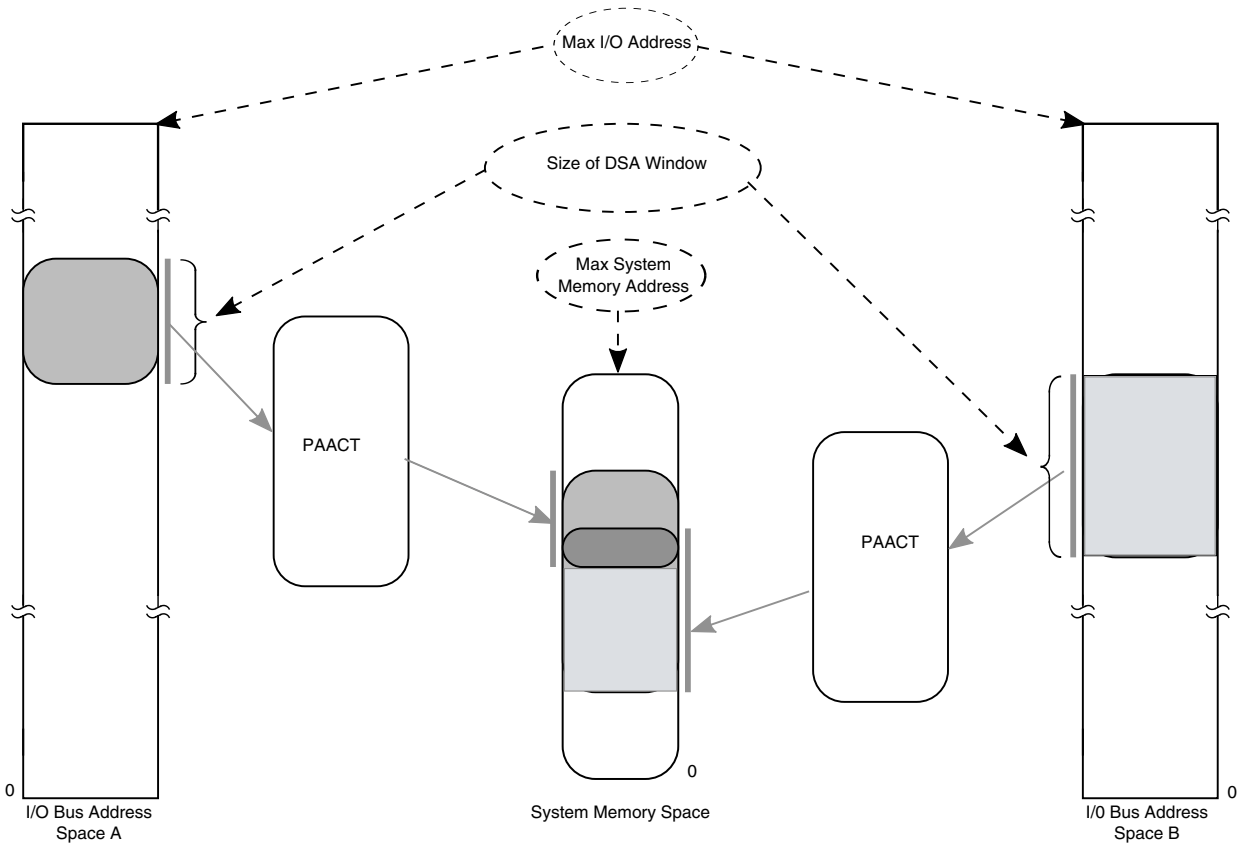


Figure 11-163. DSA Operations with Window Address Translation

11.5.3.8 OMT Access and Service

A DSA operation issued by a LIOD has an ingress operation encoding. This section describes the mechanisms to check a DSA operation's ingress operation encoding and optionally remap the encoding to a corresponding egress operation encoding.

When a valid PAACE is referenced for a DSA operation from a LIOD, the Operation Translation Mode field of the entry is used to determine the type of operation translation needed for that DSA operation. There can be three types of encodings indicated in the OTM field, as follows:

- No operation translation
- Immediate translation mode
- Indexed translation mode

11.5.3.8.1 No Operation Translation

If the source and destination protocols of the DSA operation are the same or the same ingress and egress encodings can be used to express the operation types, No operation translation may be required.

The following describes how no address translation mode is achieved for DSA operations:

- The I/O operation type of the DSA operation is also the operation type of the interconnect protocol in system storage space, or the I/O operation encoding can also be used to express the operation encoding in system storage space.

11.5.3.8.2 Operation Type Translation

The translation process starts by the PAMU-resident hardware encoding an incoming DSA operation type. The output of the encode function is a 8-bit code called the ingress operation encoding. As mentioned in PAACE Offset 0x18, bit 0 of the ingress operation encoding is the operation type qualifier:

- If IOE[0] is a 0, the DSA operation's ingress transaction is a query type of operation.
- If IOE[0] is a 1, the DSA operation's ingress transaction is a update type of operation.

The ingress operation types and their corresponding encodings to the least significant 7-bits of the IOE is I/O domain specific.

Immediate Translation Mode

The following describes DSA operation mapping when OTM indicates immediate translation mode:

- If the IOE matches the encoding contained in the IOEA field of the PAACE:
  - bit-0 of the MOEA field of the PAACE indicates whether the field contains a valid egress operation encoding.
  - If bit-0 of the MOEA field is set, MOEA[1:7] contains the corresponding egress operation encoding A (EOEA).
  - If bit-0 of the MOEA field is not set, an access violation has been detected.
- If the IOE matches the encoding contained in the IOEB field of the PAACE:
  - bit-0 of the MOEB field of the PAACE indicates whether the field contains a valid egress operation encoding.
  - If bit-0 of the MOEB field is set, MOEB[1:7] contains the corresponding egress operation encoding B (EOEB).
  - If bit-0 of the MOEB field is not set, an access violation has been detected.
- If the IOE does not match both IOEA and IOEB, an access violation has been detected.

Indexed Translation Mode

The following describes DSA operation mapping when OTM indicates indexed translation mode:

- The contents of the OMI field of the PAACE are used to index into the relevant OME off the OMT base address[1] (see example in Figure 11-164).
- The least significant 7-bits of the IOE is used to index into the OME to obtain the relevant MOE to be used[2] .
- The selected MOE contains the egress operation encoding[3]
  - bit-0 of the MOE field indicates whether field contains a valid egress operation encoding.
  - If bit-0 of the MOE field is set, MOE[1:7] contains the corresponding egress operation encoding (EOE).
  - If bit-0 of the MOE field is not set, an access violation has been detected.

1. PAMU caches the OMT to improve translation performance.
2. Thus, in the limit, up to 128 operation types may be translated.
3. A system could consolidate all operation mappings into a single global OMT with various PAACEs using appropriate indices into the table via their OMI fields.

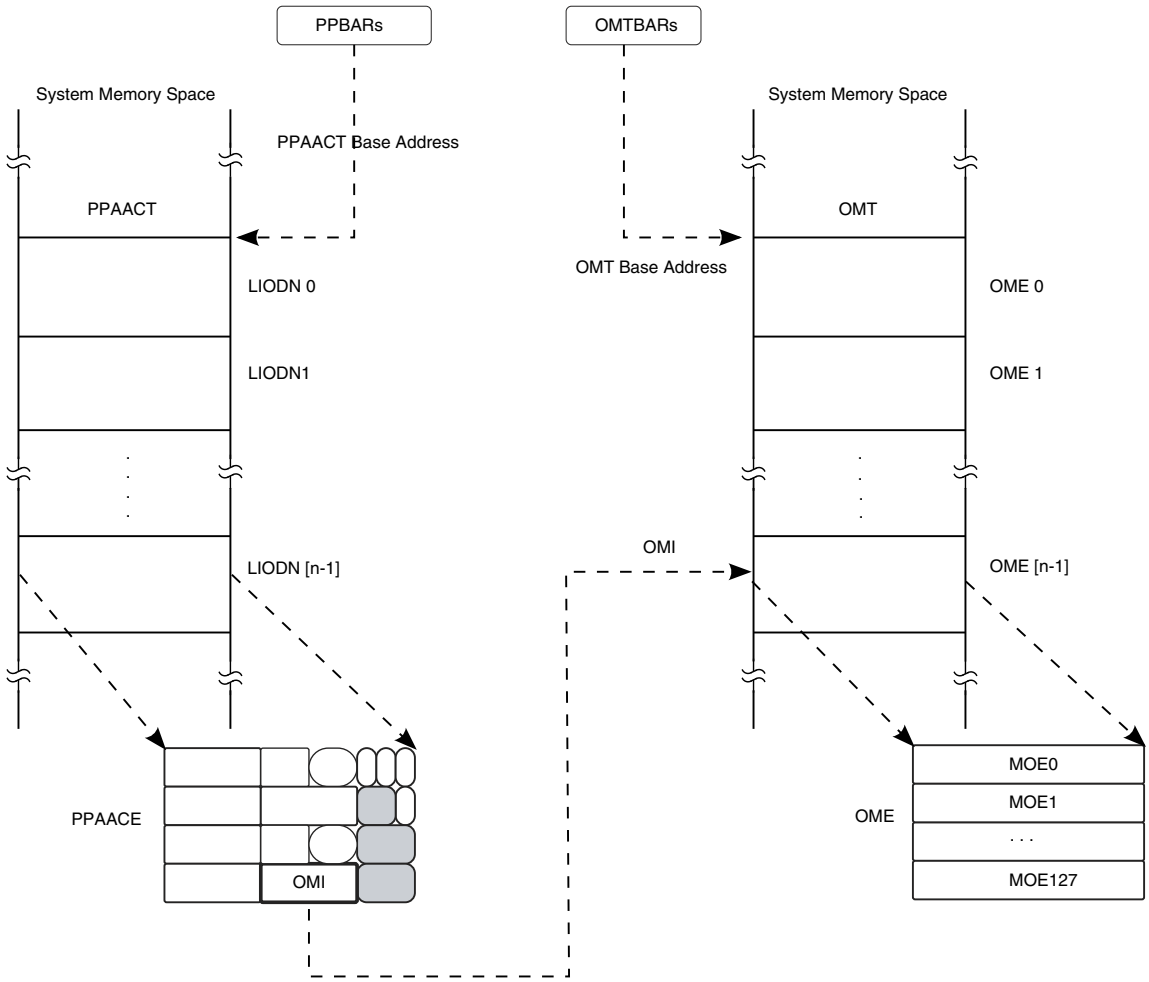


Figure 11-164. PPAACE to OME mapping in System Memory Space

11.5.3.8.3 OMT Cache Access

If the operation translation mode indicates that accessed location requires indexed operation translation performed, the PAMU will perform a lookup on the OMT cache:

- The OMT base address programmed in the OMTBARs see OMT Base Address Low register (PAMU_OBAL) and OMT Base Address High register (PAMU_OBAH) along with the OMI offset contained in the PAACE is used to determine the OME address.
- The MOMI-indicated least significant bits of the OMI are used to index into the OMT cache.
- The valid bit at the OMI index location is used to determine whether the relevant OME is in the cache.

- On a cache miss, the PAMU accesses the OME address from the system memory.
- The MIOE-indicated least significant bits of the IOE are used to index into the OME to determine the 8-bit MOE and the EOE within it.

11.5.3.9 Access Violation

As described in previous sections, during the process of authorizing and checking access permissions, PAMU may detect access violations. In addition to logging relevant error information in the access violation CCSRs (see Access Violation Status register 1 (PAMU_AVS1), through Access Violation Address Low register (PAMU_AVAL)), the following actions may also be incurred.

- An interrupt is optionally asserted based on the setup in the interrupt mapping CCSR (see PAMU Interrupt Control and Status register (PAMU_PICS)).
- For DSA operations encountering access violations, the system terminates write operations and return 0s for read operations.

11.6 PAMU Initialization/Application Information

This section describes system/software configuration and set-up of the data structures and PAMU for system implementations. This section also describes system/software interactions of the data structures and PAMU during run-time operation.

11.6.1 System Set-Up

Before the LIODs can initiate DSA operations in the system, the Hypervisor must do the following:

- Create the necessary data structures: PAACTs and OMTs in system memory.
- Set up the PAMU (see Setting Up PAMU).
- Program the corresponding LIODNs for the LIODs. Hypervisor assigns a unique LIODN to a given LIOD of a given I/O domain.

11.6.1.1 Setting Up PAMU

Each PAMU has CCSRs (see PAMU Memory Map/Register Definitions) that Hypervisor must either enable or program with the relevant locations of the data structures in system memory space:

- The base and limit addresses of the memory-resident tables accessed by PAMU are programmed. Note that a PAMU instance can access at most one instance of each of these tables at a time.
  - The primary PAACT base and limit address registers, PPBARs and PPLARs, are programmed with the relevant locations of the PPAACT.
  - The secondary PAACT base and limit address registers, SPBARs and SPLARs, are programmed with the relevant locations of the SPAACT.
  - The OMT base and limit address registers, OMTBARs and OMTLARs, are programmed with the relevant locations of the PPAACT.
- The PAMU control register is programmed to enable the necessary caches resident in the PAMU hardware.
- The PAMU fetch attributes register is programmed with the necessary information.
- As the last step, the PAMU enable bit in PAMU CCSR space must be set by Hypervisor. PAMU will now enforce authorization, translation, or access control based on the signaled LIODN.

11.6.1.2 Power-On Reset

Coming out of reset, Hypervisor must program PAMU configuration registers (PPBARs, PPLARs, SPBARs, SPLARs, OMTBARs, OMTLARs, PC) before enabling PAMU. After programming PAMU configuration registers, Hypervisor must enable PAMU by setting PE bit in PAMU control register.

11.6.2 System with Multiple PAMUs

Each PAMU may have access to authorization and access control data structures that are unique to that PAMU. Alternatively, each PAMU may have access to a common set of authorization and access control data structures. These considerations influence how each PAMU is set up.

While multiple PAMUs may share common settings in terms of accessing data structures in system memory space, the control and status registers are unique to each PAMU. For example, the error control and status registers log error information unique to each PAMU.

11.6.2.1 PAACT Locations

If the desired setup is a single set of peripheral access, authorization, and control tables (PAACTs) that are accessed by all PAMUs in the system, each PAMU would be programmed with the same primary base address and limit locations and the same secondary base address and limit locations in the corresponding CCSRs.

If the desired setup is a unique set of PAACTs for access by each PAMU in the system, each PAMU would be programmed with unique primary base address and limit locations and unique secondary base address and limit locations corresponding to the unique primary and secondary PAACT locations in system address space.

11.6.2.2 OMT Locations

Similar to the PAACT setup described in PAACT Locations, either the desired setup is a single operation mapping table (OMT) that is accessed by all PAMUs in the system, or a unique table for access by each PAMU in the system. Accordingly, all PAMUs would either be programmed with the same base address and limit locations or each PAMU would be programmed with a unique base address and limit locations corresponding to the unique OMT location in system address space.

Note that it is possible for multiple PAMUs to have unique OMT base and limit addresses programmed but have the same PPAACT base and limit address and SPAACT base and limit address programmed. This would allow for protocol specific operation translation capabilities unique to the protocol being bridged while accessing a single set of PAACT and associated data structures that contain the authorization, access control, and address translation properties unique to each LIODN.

11.6.2.3 Location of PAACT and OMT Data Structures

Regardless of whether there is a single set of PAACTs or OMT common to all PAMUs or PAACTs or OMT unique to each PAMU, these structures must be placed in one contiguous window for efficiency in the hardware coherency mechanisms. The single window is defined in a LAW that has the coherency sub-domain identifier (CSDId) field of the LAW set up in a manner that includes the relevant PAMUs in the coherency sub-domain. PAMU Cache Coherency, provides details on the setup and use of CSDId towards maintaining coherency.

11.6.3 Peer-to-Peer I/O Operations

The memory map of the system must be set up in a particular manner for authorization, access control and translation services to be made available for peer-to-peer I/O operations:

- The source peer I/O device issues an operation to I/O address space.
- CCSRs across the hierarchy relating to address decode and/or routing are set up such that the operation is sent upstream from the I/O to the host/coherency domain.
- System interface on receiving the operation performs a PAMU lookup.
- PAMU authorizes the operation and checks the access permissions.
- The translated address provided by PAMU is the address of the operation in system memory space.
- The operation with the translated system address is forwarded to the system.
- System address of the operation decodes to a LAW that identifies the peer I/O device as the destination of the operation.
- The system forwards the operation to the relevant target device.
- CCSRs across the hierarchy relating to address decode and/or routing, including the target ID designation, are set up such that the operation is sent downstream from the host/coherency domain to the destination peer I/O device.
- As a result, the destination peer I/O device receives an operation that has had authorization, access control and translation services performed.

11.6.4 PAMU Cache Coherency

The presence of caches in PAMU is transparent to application software. However, hardware coherency mechanisms are applied in order for PAMU to maintain coherent copies of the PAACT or OMT data structures in system memory. There are system setup considerations for maintaining coherent caches:

- A mastering agent, for example, a CPU, accesses either the PAACT or OMT locations in system memory.
- The system performs address decode to determine the particular LAW to be referenced for the access.
- The relevant LAW contains the coherency sub-domain identifier (CSDId) which in turn identifies all ports that are to be snooped by that operation.
- Because this access is to a PAACT or OMT location, CSDId must be set up to include ports that have PAMUs present.

11.6.5 Quiescing I/O Devices

11.6.5.1 Quiescing I/O Devices for Table/Entry Updates

While PAMU maintains coherent copies of the system memory resident data structures, a race condition still exists when device activity causes PAMU to reference in the PAMU cache a stale copy of the data structure. This is because the cache invalidating operations, related to the data structure update activity, is still in flight in the system/platform and has not reached PAMU before PAMU references the cache.

Therefore, when system software, for example, a Hypervisor, updates or moves either the PAACT or OMT locations, the software should ensure that there are no peripheral devices performing operations that require the referencing of those data structures. It may do so by quiescing the activities of the relevant I/O devices.

11.6.5.2 Quiescing I/O Devices for Enabling PAMU and its Caches

When system software, for example, a Hypervisor, enables PAMU (see PAMU Control register (PAMU_PC)) or enables one or more of its caches, the software should ensure that there are no peripheral devices performing operations that require the referencing of PAMU and its caches. Therefore, software should quiesce activity by I/O devices that would cause a PAMU lookup to occur.

11.6.6 Locality of References

The efficiency of the PPAACT, SPAACT, and OMT caches depends on how the LIODNs are enumerated, the placement of entries in system memory space, and the size of the DSA window.

11.6.6.1 Spatial Locality

The benefits associated with spatial locality of references apply to both the locality of LIODNs that accompany the different transactions flowing toward the same PAMU as well as the locality of the addresses contained within the transactions themselves:

- Enumerating LIODNs under a particular PAMU in linear order reduces the likelihood of thrashing of PPAACEs stored in the PPAACT cache.
- Placing the secondary PAACEs for one LIODN in linear order with respect to other LIODNs under the same PAMU reduces the likelihood of thrashing of SPAACEs stored in the SPAACT cache.

11.6.6.2 Temporal Locality

Achieving temporal locality of references in the caches is more a function of the size of the DSA window described by the PAACE. With the maximum transfer size in the coherency domain being 64-bytes, even the minimum 4 Kbyte granularity allowed by the architecture for a PAACE window size results in 64 references to the same PAACE for the case of a 4 Kbyte DSA access. Assigning window sizes larger than 4 Kbyte to a PAACE will result in a correspondingly higher temporal locality of reference for PAACEs stored in the PAMU cache.

11.6.7 Recovering Address Space

The architecture has several fields and settings defined as a 2^n (power-of-two) value. These values may lead to consumption of a larger address space than desired, either I/O address space or system memory space. However, it is possible to recover unallocated address space.

11.6.7.1 Primary PAACT Address Window

While LIODN is defined as a 2^n (power-of-two) value, implementations that have less than 2^n LIODNs can have system software set up the primary PAACT limit address registers to the location in system memory space following the 64-byte index location indicated by the last valid LIODN value.

11.6.7.2 Secondary PAACT Address Window

While the locations of secondary PAACEs are determined by the FSPI index in each PPAACE, system software can set up the secondary PAACT limit address registers to the location in system memory space following the last SPAACE index location of the last valid LIODN that has multiple sub-window capability.

11.6.7.3 Secondary Sub-Windows and PAACEs

The number of sub-windows for a particular LIODN is a function of the window count encoding (WCE) field in the primary PAACE, defined as a 2^n (power-of-two) value. Correspondingly, the number of secondary sub-windows and SPAACEs to be accessed for that LIODN is (2^n - 1). The secondary sub-windows described by the SPAACEs start

with the SPAACE at the FSPI index location in the SPAACT that describes the first secondary sub-window. Subsequent sub-windows are described by subsequent SPAACEs after the SPAACE at the FSPI index location.

Implementations that have less than (2^n - 1) secondary sub-windows, and therefore less than (2^n - 1) SPAACEs for a particular LIODN can have that LIODN programmed into the LIODN field for only the required number of SPAACEs. This allows for the recoverability of the I/O bus address space for unallocated sub-windows for that LIODN. This also allows for the recoverability of SPAACE index locations in the SPAACT.

11.6.8 Data Structure Size and Alignment

While the Architecture describes the size of a PAACE as being 64-bytes and OME as being 128-bytes, the PAMU implementation only references the defined 32-bytes of the PAACE data structure and 16-bytes of the OME data structure. However, in order to maintain programming model consistency with the architecture across multiple generations of PAMU implementations, software must maintain the 64-byte and 128-byte address alignment for the PAACE and OME data structures respectively.

11.7 PAMU Setup

11.7.1 PAMU Operation Encoding

The specification allows for several methods to achieve the ingress and egress operation encoding (IOE & EOE) from the source and destination interconnect protocols. For this chip, an implementation-specific decode table will be used to achieve both the ingress and egress operation types (see Figure 11-165):

- The IOE is acquired from an SoC-specific Ingress Decode Table (IDT) that identifies the operation type that generate a particular IOE.
- The EOE is an index into an SoC-specific Egress Decode Table (EDT) that elaborates the CoreNet transaction type to be generated.

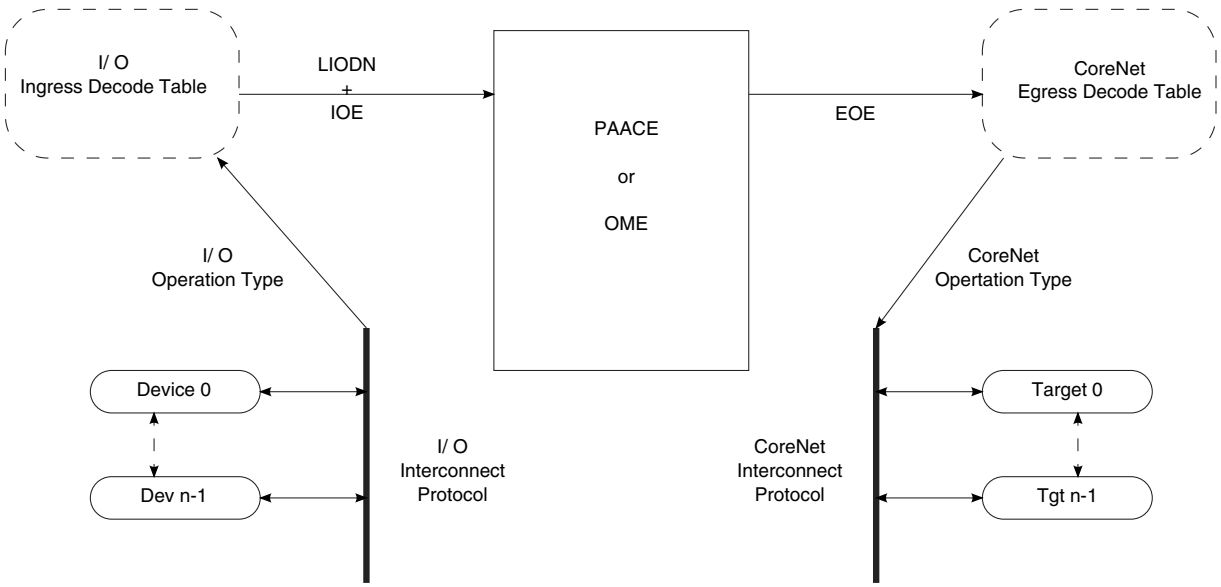


Figure 11-165. Illustration: Generation of Ingress and Egress Operation Encodings

11.7.1.1 Ingress Operation Encoding (IOE)

Table 11-165 describes the ingress operation encodings that are issued on the PAMU Lookup Interface.

Table 11-165. PAMU Ingress Operation Encodings

Ingress Operation Encoding [0:7]	Name	Description
0_000_0000	READ	Read
1_000_0001	WRITE	Write
1_000_0010	EREAD0	Enhanced Read Type 0
1_000_0011	EWRITE0	Enhanced Write Type 0
1_000_0100	DIRECT0	Directive Type 0
1_000_0101	EREAD1	Enhanced Read Type 1
1_000_0110	EWRITE1	Enhanced Write Type 1
1_000_0111	DIRECT1	Directive Type 1
1_000_1100	RAC	Read with Atomic Clear
1_000_1101	RAS	Read with Atomic Set
1_000_1110	RAD	Read with Atomic Decrement
1_000_1111	RAI	Read with Atomic Increment
All other encodings are reserved		

11.7.1.2 Egress Operation Encodings

Table 11-166 describes the egress operation encodings that are issued on the PAMU Lookup Response Interface. The EOEs are indexes into CoreNet commands.

NOTE

EOE[0] is not referenced for the Egress Decode Table. This is because EOE[0] == MOE[0] which determines if the EOE is valid in either the PAACE or OME.

Table 11-166. PAMU Egress Decode Table[1]

EOE [1:7]	Description
READ (000_0000)	Read
WRITE (000_0001)	Write
RAC (000_1100)	Read with Atomic Clear
RAS (000_1101)	Read with Atomic Set
RAD (000_1110)	Read with Atomic Decrement
RAI (000_1111)	Read with Atomic Increment
LDEC (001_0000)	Load External Cache
LDECL (001_0001)	Load External Cache with Stash Lock
LDECPE (001_0010)	Load External Cache with Preferred Exclusive
LDECPEL (001_0011)	Load External Cache with Preferred Exclusive and Lock
LDECFE (001_0100)	Load External Cache with Forced Exclusive
LDECFEL (001_0101)	Load External Cache with Forced Exclusive and Lock
RSA (001_0110)	Read with Stash Allocate
RSAU (001_0111)	Read with Stash Allocate and Unlock
READI (001_1000)	Read with Invalidate
RWNITC (001_1001)	Read with No Intention to Cache
WCI (001_1010)	Write Cache Inhibited
WWSA (001_1011)	Write with Stash Allocate
WWSAL (001_1100)	Write with Stash Allocate and Lock
WWSOT (001_1101)	Write with Stash or Target
WWSOTL (001_1110)	Write with Stash Allocate and Lock or Target
All other encodings are reserved	

1. Stashing capability is restricted to 64-byte operations in the chip. Operations < 64 bytes will not have the stash portion of the operation performed.

2. While CoreNet supports all sizes for Atomic Operations, the chip will only support 1,2,4, and 8 byte sizes. See CoreNet Platform Cache (CPC) for resultant behavior when EOE mappings do not fall within these size constraints.

11.7.1.3 IOE to EOE Translations

Due to operation semantics at the source interconnect protocol, only certain ingress operation encodings can map in a seamless manner to an egress operation encoding in order to achieve the same operation semantics at the destination interconnect protocol (CoreNet for PAMU).

11.7.1.3.1 PAMU Bypass Mode and No Operation Translation Mode

Only the basic command set is supported when PAMU is in Bypass Mode. Thus basic Read and Write operations are supported. In the case of SRIO, the Read Atomic Operations are supported as well.

The same basic command set is supported when PAMU is not bypassed but the PAACE entry for that LIODN indicates No Operation Translation has been enabled. Unsupported enhanced IOEs map to reserved EOEs that are used by hardware to detect that an unsupported IOE was encountered while No Operation Translation Mode was enabled.

Table 11-167 shows the default IOE to EOE mappings that occur when PAMU is in bypass mode or when the PAACE entry for an LIODN indicates No Operation Translation has been enabled.

Table 11-167. Default IOE to EOE mappings: PAMU Bypass Mode and No Operation Translation Mode

IOE	EOE
Basic Commands	
READ	READ
WRITE	WRITE
RAC	RAC
RAS	RAS
RAD	RAD
RAI	RAI
Enhanced Commands- Unsupported	
EREAD0	EREAD0
EWRITE0	EWRITE0
DIRECT0	DIRECT0
EREAD1	EREAD1
EWRITE1	EWRITE1
DIRECT1	DIRECT1

Table 11-168 shows the CoreNet operations that can achieved when PAMU is in bypass mode or when the PAACE entry for an LIODN indicates No Operation Translation has been enabled. Issuance of operations by an LIODN outside of that listed in Table 11-168 can be achieved only when an OMT has been created with the relevant mappings and the relevant PAACE entry for that LIODN has the Immediate or Indexed Operation Translation Mode enabled (see Immediate and Indexed Operation Translation Modes).

Table 11-168. Default CoreNet operations in PAMU Bypass Mode and No Operation Translation Mode

Description
Read
Write
Read with Atomic Clear
Read with Atomic Set
Read with Atomic Decrement
Read with Atomic Increment

11.7.1.3.2 Immediate and Indexed Operation Translation Modes

Table 11-169 shows the permitted EOEs for particular IOEs for Immediate and Indexed Operation Translation.

Table 11-169. Permitted IOE to EOE translations for Immediate and Index Operation Translation

IOE	EOE
READ, EREAD0, EREAD1	READ, RAC, RAD, RAS, RAI, RSA, RSAU, READI, RWNITC
WRITE, EWRITE0, EWRITE1	WRITE, WCI, WWSA, WWSAL, WWSOT, WWSOTL
DIRECT0, DIRECT1	LDEC, LDECL, LDECPE, LDECPEL, LDECFE, LDECFEL
RAC	RAC
RAS	RAS
RAD	RAD
RAI	RAI

11.7.2 Domain Attributes

11.7.2.1 Constrained Domain Attribute

Table 11-170 describes a domain attribute field constraint in this chip.

Table 11-170. Domain Attributes constrained

Bits	Name	PPAACE (P) and/or SPAACE (S) Field	Description
16-23	SnpID	P & S	Snoop ID While PAMU in this chip can access an 8-bit Snoop ID Domain Attribute from the PAACE, the CoreNet Coherency Fabric (CCF) in the chip supports up to 32 Snoop IDs Therefore, only SnpID[3:7] is supported.

11.7.3 Implementation Attributes

Table 11-171 describes the SoC definition of Implementation Attributes fields of PAACE Offset 0x0C through 0x0E.

Table 11-171. Implementation Attributes

Bits	Name	PPAACE (P) and/or SPAACE (S) Field	Description
0-7	-	-	Reserved
8-15	CID	P and S	Cache ID These bits define the cache identifier. This field may be used to identify a specific cache entity for the window described by this PAACE. These bits are used for explicit stashing where CID identifies the specific destination.
16-23	-	-	Reserved

11.7.4 Operation Mapping Table (OMT)

For this chip, OMI can be up to 4 bits supporting 16 OMEs. Each OME can contain up to 16 valid bytes supporting 16 MOEs. Because an OME can have at most 16 valid bytes in the 128-byte OME, PAMU fetches 16 bytes of data after a OMT cache miss. Invalidation will only invalidate one entry out of the 4 entries in the OMT cache. Each entry will hold 128 bits of data.

Chapter 12
DDR Memory Controller

12.1 DDR Introduction

The fully programmable DDR SDRAM controller supports most JEDEC standard x8, x16 DDR3/3L memories available. In addition, unbuffered and registered DIMMs are supported. However, mixing different memory types or unbuffered and registered DIMMs in the same system is not supported. Built-in error checking and correction (ECC) ensures very low bit-error rates for reliable high-frequency operation. Dynamic power management and auto-precharge modes simplify memory system design. A large set of special features, including ECC error injection, support rapid system debug.

NOTE

In this chapter, the word 'bank' refers to a physical bank specified by a chip select; 'logical bank' refers to one of the four or eight sub-banks in each SDRAM chip. A sub-bank is specified by the 3 bits on the bank address (MBA) pins during a memory access.

The figure below is a high-level block diagram of the DDR memory controller with its associated interfaces.

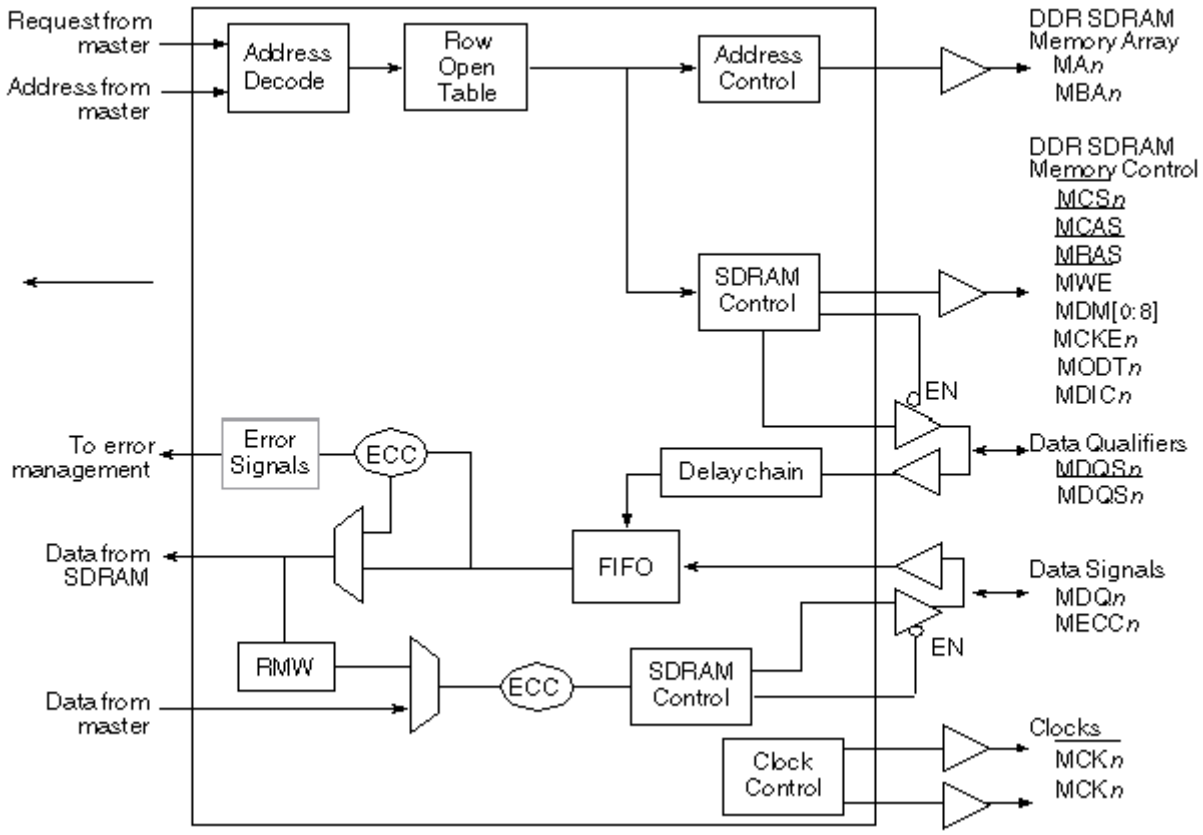


Figure 12-1. DDR Memory Controller Simplified Block Diagram

12.2 DDR Features

The DDR memory controller includes these distinctive features:

- Support for DDR3/3L SDRAM
- 64-/72-bit SDRAM data bus, 32-/40-bit SDRAM for DDR3/3L
- Programmable settings for meeting all SDRAM timing parameters
- The following SDRAM configurations are supported:
- As many as four physical banks (chip selects), each bank independently addressable
- 64-Mbit to 8 -Gbit devices depending on internal device configuration with x8/x16 data ports
- Unbuffered and registered DIMMs
- Chip select interleaving support
- Partial array self refresh support
- Support for data mask signals and read-modify-write for sub-double-word writes. Note that a read-modify-write sequence is only necessary when ECC is enabled.
- Support for double-bit error detection and single-bit error correction ECC (8-bit check word across 64-bit data)

- Support for address parity for registered DIMMs
- Open page management (dedicated entry for each logical bank)
- Automatic DRAM initialization sequence or software-controlled initialization sequence
- Automatic DRAM data initialization
- Interrupt driven rapid clear of memory
- Write leveling supported for DDR3 memories
- Support for up to eight posted refreshes
- Memory controller clock frequency of two times the SDRAM clock with support for sleep power management
- Support for error injection

12.2.1 DDR Modes of Operation

The DDR memory controller supports the following modes:

- Dynamic power management mode. The DDR memory controller can reduce power consumption by negating the SDRAM CKE signal when no transactions are pending to the SDRAM.
- Auto-precharge mode. Clearing DDR_SDRAM_INTERVAL[BSTOPRE] causes the memory controller to issue an auto-precharge command with every read or write transaction. Auto-precharge mode can be enabled for separate chip selects by setting CS *n*_CONFIG[AP_ *n*_EN].

12.3 DDR External Signal Descriptions

This section provides descriptions of the DDR memory controller's external signals. It describes each signal's behavior when the signal is asserted or negated and when the signal is an input or an output.

12.3.1 DDR Signals Overview

Memory controller signals are grouped as follows:

- Memory interface signals
- Clock signals
- Debug signals

The following table shows how DDR memory controller external signals are grouped. The device hardware specification has a pinout diagram showing pin numbers. It also lists all electrical and mechanical specifications.

Table 12-1. DDR Memory Interface Signal Summary

Name	Function/Description	Reset	Pins	I/O
MAPAR_ERR_B	Address Parity Error	One	1	I
MAPAR_OUT_B	Address Parity Out	Zero	1	O
MDQ[0:63]	Data bus	All zeros	64	I/O
MDQS[0:8]	Data strobes	All zeros	9	I/O
MDQS[0:8]_B	Complement data strobes	All ones	9	I/O
MECC[0:7]	Error checking and correcting	All zeros	8	I/O
MCAS_B	Column address strobe	One	1	O
MA[15:0]	Address bus	All zeros	16	O
MBA[2:0]	Logical bank address	All zeros	3	O
MCS[0:3]_B	Chip selects	All ones	4	O
MWE_B	Write enable	One	1	O
MRAS_B	Row address strobe	One	1	O
MDM[0:8]	Data mask	All zeros	9	O
MCK[0:3]	DRAM clock outputs	All zeros	4	O
MCK[0:3]_B	DRAM clock outputs (complement)	All zeros	4	O
MCKE[0: 3]	DRAM clock enable	All zeros	4	O
MODT[0: 3]	DRAM on-die termination	All zeros	4	O
MDIC[0:1]	Driver impedance calibration	b10	2	I/O

The table below shows the memory address signal mappings.

Table 12-2. Memory Address Signal Mappings

Signal Name (Outputs)		JEDEC DDR DIMM Signals (Inputs)
msb	MA15	A15
	MA14	A14
	MA13	A13
	MA12	A12
	MA11	A11
	MA10	A10 (AP for DDR)
	MA9	A9
	MA8	A8
	MA7	A7
	MA6	A6
	MA5	A5
	MA4	A4

Table continues on the next page...

Table 12-2. Memory Address Signal Mappings (continued)

Signal Name (Outputs)		JEDEC DDR DIMM Signals (Inputs)
	MA3	A3
	MA2	A2
	MA1	A1
lsb	MA0	A0
msb	MBA2	MBA2
	MBA1	MBA1
lsb	MBA0	MBA0

12.3.2 DDR Detailed Signal Descriptions

The following sections describe the DDR SDRAM controller input and output signals, the meaning of their different states, and relative timing information for assertion and negation.

12.3.2.1 Memory Interface Signals

The following table describes the DDR controller memory interface signals.

Table 12-3. Memory Interface Signals-Detailed Signal Descriptions

Signal	I/O	Description	
MDQ[0:63]	I/O	Data bus. Both input and output signals on the DDR memory controller.	
	O	As outputs for the bidirectional data bus, these signals operate as described below.	
		State Meaning	Asserted/Negated - Represent the value of data being driven by the DDR memory controller.
		Timing	Assertion/Negation - Driven with valid data during writes to memory. High impedance - No READ or WRITE command is in progress; data is not being driven by the memory controller or the DRAM.
	I	As inputs for the bidirectional data bus, these signals operate as described below.	
		State Meaning	Asserted/Negated - Represents the state of data being driven by the external DDR SDRAMs.
		Timing	Assertion/Negation - The DDR SDRAM drives data during a READ transaction. High impedance - No READ or WRITE command in progress; data is not being driven by the memory controller or the DRAM.
MDQS[0:8]/ MDQS[0:8]_B	I/O	Data strobes. Inputs with read data, outputs with write data. The data strobes must be differential.	
	O	As outputs, the data strobes are driven by the DDR memory controller during a write transaction.	

Table continues on the next page...

Table 12-3. Memory Interface Signals-Detailed Signal Descriptions (continued)

Signal	I/O	Description	
		State Meaning	Asserted/Negated - Driven high when positive capture data is transmitted and driven low when negative capture data is transmitted. Centered in the data "eye" for writes; coincident with the data eye for reads. Treated as a clock. Data is valid when signals toggle. See Table 12-78 for byte lane assignments.
		Timing	Assertion/Negation - If a WRITE command is registered at clock edge n, data strobes at the DRAM assert centered in the data eye on clock edge n + 1. See the JEDEC DDR SDRAM specification for more information.
	I	As inputs, the data strobes are driven by the external DDR SDRAMs during a read transaction. The data strobes are used by the memory controller to synchronize data latching.	
		State Meaning	Asserted/Negated - Driven high when positive capture data is received and driven low when negative capture data is received. Centered in the data eye for writes; coincident with the data eye for reads. Treated as a clock. Data is valid when signals toggle. See Table 12-78 for byte lane assignments.
		Timing	Assertion/Negation - If a READ command is registered at clock edge n, and the latency is programmed in TIMING_CFG_1[CASLAT] to be *m* clocks, data strobes at the DRAM assert coincident with the data on clock edge *n* + *m*. See the JEDEC DDR SDRAM specification for more information.
MECC[0:7]	I/O	Error checking and correcting codes. Input and output signals for the DDR controller's bidirectional ECC bus.	
	O	As normal mode outputs the ECC signals represent the state of ECC driven by the DDR controller on writes. See Error Checking and Correcting (ECC) for more details.	
		State Meaning	Asserted/Negated - Represents the state of ECC being driven by the DDR controller on writes.
		Timing	Assertion/Negation - Same timing as MDQ High impedance - Same timing as MDQ
	I	As inputs, the ECC signals represent the state of ECC driven by the SDRAM devices on reads.	
		State Meaning	Asserted/Negated - Represents the state of ECC being driven by the DDR SDRAMs on reads.
		Timing	Assertion/Negation - Same timing as MDQ High impedance - Same timing as MDQ
MA[15:0]	O	Address bus. Memory controller outputs for the address to the DRAM. MA[15:0] carry 16 of the address bits for the DDR memory interface corresponding to the row and column address bits. MA0 is the lsb of the address output from the memory controller.	
		State Meaning	Asserted/Negated - Represents the address driven by the DDR memory controller. Contains different portions of the address depending on the memory size and the DRAM command being issued by the memory controller. See Table 12-80 for a complete description of the mapping of these signals.
		Timing	Assertion/Negation - The address is always driven when the memory controller is enabled. It is valid when a transaction is driven to DRAM (when MCS n is active). High impedance - When the memory controller is disabled
MBA[2:0]	O	Logical bank address. Outputs that drive the logical (or internal) bank address pins of the SDRAM. Each SDRAM supports four or eight addressable logical sub-banks. Bit zero of the memory controller's output bank address must be connected to bit zero of the SDRAM's input bank address. MBA0, the least-significant bit of the three bank address signals, is asserted during the mode register set command to specify the extended mode register.	

Table continues on the next page...

Table 12-3. Memory Interface Signals-Detailed Signal Descriptions (continued)

Signal	I/O	Description	
		State Meaning	Asserted/Negated - Selects the DDR SDRAM logical (or internal) bank to be activated during the row address phase and selects the SDRAM internal bank for the read or write operation during the column address phase of the memory access. Table 12-80 describes the mapping of these signals in all cases.
		Timing	Assertion/Negation - Same timing as MAn High impedance - Same timing as MAn
MCAS_B	O	Column address strobe. Active-low SDRAM address multiplexing signal. MCAS_B is asserted for read or write transactions and for mode register set, refresh, and precharge commands.	
		State Meaning	Asserted - Indicates that a valid SDRAM column address is on the address bus for read and write transactions. See Table 12-86 for more information on the states required on MCAS_B for various other SDRAM commands. Negated - The column address is not guaranteed to be valid.
		Timing	Assertion/Negation - Assertion and negation timing is directed by the values described in DDR SDRAM timing configuration 0 (DDR_TIMING_CFG_0)DDR SDRAM timing configuration 1 (DDR_TIMING_CFG_1)DDR SDRAM timing configuration 2 (DDR_TIMING_CFG_2) and DDR SDRAM timing configuration 3 (DDR_TIMING_CFG_3). High impedance - MCAS_B is always driven unless the memory controller is disabled.
MRAS_B	O	Row address strobe. Active-low SDRAM address multiplexing signal. Asserted for activate commands. In addition; used for mode register set commands and refresh commands.	
		State Meaning	Asserted - Indicates that a valid SDRAM row address is on the address bus for read and write transactions. See Table 12-86 for more information on the states required on MRAS_B for various other SDRAM commands. Negated - The row address is not guaranteed to be valid.
		Timing	Assertion/Negation - Assertion and negation timing is directed by the values described in DDR SDRAM timing configuration 0 (DDR_TIMING_CFG_0)DDR SDRAM timing configuration 1 (DDR_TIMING_CFG_1)DDR SDRAM timing configuration 2 (DDR_TIMING_CFG_2) and DDR SDRAM timing configuration 3 (DDR_TIMING_CFG_3). High impedance - MRAS_B is always driven unless the memory controller is disabled.
MCS[0:3]_B	O	Chip selects. Four chip selects supported by the memory controller.	
		State Meaning	Asserted - Selects a physical SDRAM bank to perform a memory operation as described in Chip select n memory bounds (DDR_CS*n*_BNDS) and Chip select n configuration (DDR_CS*n*_CONFIG)." The DDR controller asserts one of the MCS_B [0:3]signals to begin a memory cycle. Negated - Indicates no SDRAM action during the current cycle.
		Timing	Assertion/Negation - Asserted to signal any new transaction to the SDRAM. The transaction must adhere to the timing constraints set in TIMING_CFG_0-TIMING_CFG_3. High impedance - Always driven unless the memory controller is disabled.
MWE_B	O	Write enable. Asserted when a write transaction is issued to the SDRAM. This is also used for mode registers set commands and precharge commands.	
		State Meaning	Asserted - Indicates a memory write operation. See Table 12-86 for more information on the states required on MWE_B for various other SDRAM commands. Negated - Indicates a memory read operation.
		Timing	Assertion/Negation - Similar timing as MRAS_B and MCAS_B. Used for write commands. High impedance - MWE_B is always driven unless the memory controller is disabled.

Table continues on the next page...

Table 12-3. Memory Interface Signals-Detailed Signal Descriptions (continued)

Signal	I/O	Description	
MDM[0:8]	O	DDR SDRAM data output mask. Masks unwanted bytes of data transferred during a write. They are needed to support sub-burst-size transactions (such as single-byte writes) on SDRAM where all I/O occurs in multi-byte bursts. MDM0 corresponds to the most significant byte (MSB) and MDM7 corresponds to the LSB, while MDM8 corresponds to the ECC byte. Table 12-78 shows byte lane encodings.	
		State Meaning	Asserted - Prevents writing to DDR SDRAM. Asserted when data is written to DRAM if the corresponding byte(s) should be masked for the write. Note that the MDMn signals are active-high for the DDR controller. MDMn is part of the DDR command encoding. Negated - Allows the corresponding byte to be read from or written to the SDRAM.
		Timing	Assertion/Negation - Same timing as MDQx as outputs. High impedance - Always driven unless the memory controller is disabled.
MODT[0: 3]	O	On-Die termination. Memory controller outputs for the ODT to the DRAM. MODT[0: 3]represents the on-die termination for the associated data, data masks, ECC, and data strobes.	
		State Meaning	Asserted/Negated - Represents the ODT driven by the DDR memory controller.
		Timing	Assertion/Negation - Driven in accordance with JEDEC DRAM specifications for on-die termination timings. It is configured through the CS n _CONFIG[ODT_RD_CFG] and CS n _CONFIG[ODT_WR_CFG] fields. High impedance - Always driven.
MDIC[0:1]	I/O	Driver impedance calibration. Note that the MDIC signals require the use of 237−Ω precision 1% resistors; MDIC0 must be pulled to GND, while MDIC1 must be pulled to GV_{DD}. See DDR Control Driver Register 2 (DDR_DDRCDR_2) for more information on these signals.	
		State Meaning	These pins are used for automatic calibration of the DDR IOs.
		Timing	These will be driven for four DRAM cycles at a time while the DDR controller is executing the automatic driver compensation.
MAPAR_ERR_B	I	Address parity error. Reflects whether an address parity error has been detected by the DRAM. This signal is active low.	
		State Meaning	Asserted - An error has been detected. Negated - An error has not been detected.
		Timing	Assertion/Negation - Driven by the registered DDR3 DIMMs 3 DRAM cycles after the parity bit has been driven by the memory controller. This error signal should be held valid for 2 DRAM cycles.
MAPAR_OUT_B	O	Address parity out. Driven by the memory controller as the parity bit calculated across the address and command bits. Even parity is used, and parity is not calculated for the MCKE[0:3], MODT[0:3], or MCS[0:3]_B signals.	
		State Meaning	Asserted - The parity bit is high. Negated - The parity bit is low.
		Timing	Assertion/Negation - Will be issued one DRAM cycle after the chip select for each command.

12.3.2.2 Clock Interface Signals

The following table contains the detailed descriptions of the clock signals of the DDR controller.

Table 12-4. Clock Signals-Detailed Signal Descriptions

Signal	I/O	Description	
MCK[0:3], MCK_B [0:3]	O	DRAM clock outputs and their complements. See Clock Distribution."	
		State Meaning	Asserted/Negated-The JEDEC DDR SDRAM specifications require true and complement clocks. A clock edge is seen by the SDRAM when the true and complement cross.
		Timing	Assertion/Negation-Timing is controlled by the DDR_CLK_CNTL register at offset 0x130.
MCKE[0: 3]	O	Clock enable. Output signals used as the clock enables to the SDRAM. MCKE[0: 3] can be negated to stop clocking the DDR SDRAM. The MCKE signals should be connected to the same rank of memory as the corresponding MCS_B and MODT signals. For example, MCKE[0] should be connected to the same rank of memory as MCS_B[0] and MODT[0].	
		State Meaning	Asserted-Clocking to the SDRAM is enabled. Negated-Clocking to the SDRAM is disabled and the SDRAM should ignore signal transitions on MCK or MCK_B. MCK/MCK_B are don't cares while MCKE[0: 3] are negated.
		Timing	Assertion/Negation-Asserted when DDR_SDRAM_CFG[MEM_EN] is set. Can be negated when entering dynamic power management or self refresh. Will be asserted again when exiting dynamic power management or self refresh. High impedance-Always driven.

12.4 DDR Memory Map/Register Definition

The table below shows the register memory map for the DDR memory controller.

DDR memory map

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8000	Chip select n memory bounds (DDR_CS0_BNDS)	32	R/W	0000_0000h	12.4.1/464
8008	Chip select n memory bounds (DDR_CS1_BNDS)	32	R/W	0000_0000h	12.4.1/464
8010	Chip select n memory bounds (DDR_CS2_BNDS)	32	R/W	0000_0000h	12.4.1/464
8018	Chip select n memory bounds (DDR_CS3_BNDS)	32	R/W	0000_0000h	12.4.1/464
8080	Chip select n configuration (DDR_CS0_CONFIG)	32	R/W	0000_0000h	12.4.2/464
8084	Chip select n configuration (DDR_CS1_CONFIG)	32	R/W	0000_0000h	12.4.2/464
8088	Chip select n configuration (DDR_CS2_CONFIG)	32	R/W	0000_0000h	12.4.2/464
808C	Chip select n configuration (DDR_CS3_CONFIG)	32	R/W	0000_0000h	12.4.2/464
80C0	Chip select n configuration 2 (DDR_CS0_CONFIG_2)	32	R/W	0000_0000h	12.4.3/467

Table continues on the next page...

DDR memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
80C4	Chip select n configuration 2 (DDR_CS1_CONFIG_2)	32	R/W	0000_0000h	12.4.3/467
80C8	Chip select n configuration 2 (DDR_CS2_CONFIG_2)	32	R/W	0000_0000h	12.4.3/467
80CC	Chip select n configuration 2 (DDR_CS3_CONFIG_2)	32	R/W	0000_0000h	12.4.3/467
8100	DDR SDRAM timing configuration 3 (DDR_TIMING_CFG_3)	32	R/W	0000_0000h	12.4.4/468
8104	DDR SDRAM timing configuration 0 (DDR_TIMING_CFG_0)	32	R/W	0011_0105h	12.4.5/471
8108	DDR SDRAM timing configuration 1 (DDR_TIMING_CFG_1)	32	R/W	0000_0000h	12.4.6/474
810C	DDR SDRAM timing configuration 2 (DDR_TIMING_CFG_2)	32	R/W	0000_0000h	12.4.7/478
8110	DDR SDRAM control configuration (DDR_DDR_SDRAM_CFG)	32	R/W	0200_0000h	12.4.8/481
8114	DDR SDRAM control configuration 2 (DDR_DDR_SDRAM_CFG_2)	32	R/W	0000_0000h	12.4.9/484
8118	DDR SDRAM mode configuration (DDR_DDR_SDRAM_MODE)	32	R/W	0000_0000h	12.4.10/487
811C	DDR SDRAM mode configuration 2 (DDR_DDR_SDRAM_MODE_2)	32	R/W	0000_0000h	12.4.11/487
8120	DDR SDRAM mode control (DDR_DDR_SDRAM_MD_CNTL)	32	R/W	0000_0000h	12.4.12/488
8124	DDR SDRAM interval configuration (DDR_DDR_SDRAM_INTERVAL)	32	R/W	0000_0000h	12.4.13/492
8128	DDR SDRAM data initialization (DDR_DDR_DATA_INIT)	32	R/W	0000_0000h	12.4.14/492
8130	DDR SDRAM clock control (DDR_DDR_SDRAM_CLK_CNTL)	32	R/W	0200_0000h	12.4.15/493
8148	DDR training initialization address (DDR_DDR_INIT_ADDR)	32	R/W	0000_0000h	12.4.16/494
814C	DDR training initialization extended address (DDR_DDR_INIT_EXT_ADDRESS)	32	R/W	0000_0000h	12.4.17/495
8160	DDR SDRAM timing configuration 4 (DDR_TIMING_CFG_4)	32	R/W	0000_0000h	12.4.18/496
8164	DDR SDRAM timing configuration 5 (DDR_TIMING_CFG_5)	32	R/W	0000_0000h	12.4.19/500
8168	DDR SDRAM timing configuration 6 (DDR_TIMING_CFG_6)	32	R/W	0000_0000h	12.4.20/502
8170	DDR ZQ calibration control (DDR_DDR_ZQ_CNTL)	32	R/W	0000_0000h	12.4.21/504
8174	DDR write leveling control (DDR_DDR_WRLVL_CNTL)	32	R/W	0000_0000h	12.4.22/507
817C	DDR Self Refresh Counter (DDR_DDR_SR_CNTR)	32	R/W	0000_0000h	12.4.23/509
8180	DDR Register Control Words 1 (DDR_DDR_SDRAM_RCW_1)	32	R/W	0000_0000h	12.4.24/510
8184	DDR Register Control Words 2 (DDR_DDR_SDRAM_RCW_2)	32	R/W	0000_0000h	12.4.25/511
8190	DDR write leveling control 2 (DDR_DDR_WRLVL_CNTL_2)	32	R/W	0000_0000h	12.4.26/512
8194	DDR write leveling control 3 (DDR_DDR_WRLVL_CNTL_3)	32	R/W	0000_0000h	12.4.27/513
8200	DDR SDRAM mode configuration 3 (DDR_DDR_SDRAM_MODE_3)	32	R/W	0000_0000h	12.4.28/515
8204	DDR SDRAM mode configuration 4 (DDR_DDR_SDRAM_MODE_4)	32	R/W	0000_0000h	12.4.29/516

Table continues on the next page...

DDR memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8208	DDR SDRAM mode configuration 5 (DDR_DDR_SDRAM_MODE_5)	32	R/W	0000_0000h	12.4.30/516
820C	DDR SDRAM mode configuration 6 (DDR_DDR_SDRAM_MODE_6)	32	R/W	0000_0000h	12.4.31/517
8210	DDR SDRAM mode configuration 7 (DDR_DDR_SDRAM_MODE_7)	32	R/W	0000_0000h	12.4.32/518
8214	DDR SDRAM mode configuration 8 (DDR_DDR_SDRAM_MODE_8)	32	R/W	0000_0000h	12.4.33/519
8B20	DDR Debug Status Register 1 (DDR_DDRDSR_1)	32	R	0000_0000h	12.4.34/520
8B24	DDR Debug Status Register 2 (DDR_DDRDSR_2)	32	R	0000_0000h	12.4.35/521
8B28	DDR Control Driver Register 1 (DDR_DDRCDR_1)	32	R/W	0000_0000h	12.4.36/522
8B2C	DDR Control Driver Register 2 (DDR_DDRCDR_2)	32	R/W	0000_0000h	12.4.37/524
8BF8	DDR IP block revision 1 (DDR_DDR_IP_REV1)	32	R	0000_0000h	12.4.38/525
8BFC	DDR IP block revision 2 (DDR_DDR_IP_REV2)	32	R	0000_0000h	12.4.39/525
8C00	DDR Enhanced Optimization Register (DDR_DDR_EOR)	32	R/W	0000_0000h	12.4.40/526
8D00	DDR Memory Test Control Register (DDR_DDR_MTCR)	32	R/W	0000_0000h	12.4.41/527
8D20	DDR Memory Test Pattern n Register (DDR_DDR_MTP)	32	R/W	0000_0000h	12.4.42/529
8D60	DDR Memory Test Start Extended Address (DDR_DDR_MT_ST_EXT_ADDR)	32	R/W	0000_0000h	12.4.43/529
8D64	DDR Memory Test Start Address (DDR_DDR_MT_ST_ADDR)	32	R/W	0000_0000h	12.4.44/530
8D68	DDR Memory Test End Extended Address (DDR_DDR_MT_END_EXT_ADDR)	32	R/W	0000_0000h	12.4.45/530
8D6C	DDR Memory Test End Address (DDR_DDR_MT_END_ADDR)	32	R/W	0000_0000h	12.4.46/531
8E00	Memory data path error injection mask high (DDR_DATA_ERR_INJECT_HI)	32	R/W	0000_0000h	12.4.47/531
8E04	Memory data path error injection mask low (DDR_DATA_ERR_INJECT_LO)	32	R/W	0000_0000h	12.4.48/531
8E08	Memory data path error injection mask ECC (DDR_ECC_ERR_INJECT)	32	R/W	0000_0000h	12.4.49/532
8E20	Memory data path read capture high (DDR_CAPTURE_DATA_HI)	32	R/W	0000_0000h	12.4.50/533
8E24	Memory data path read capture low (DDR_CAPTURE_DATA_LO)	32	R/W	0000_0000h	12.4.51/533
8E28	Memory data path read capture ECC (DDR_CAPTURE_ECC)	32	R/W	0000_0000h	12.4.52/534
8E40	Memory error detect (DDR_ERR_DETECT)	32	w1c	0000_0000h	12.4.53/535
8E44	Memory error disable (DDR_ERR_DISABLE)	32	R/W	0000_0000h	12.4.54/537
8E48	Memory error interrupt enable (DDR_ERR_INT_EN)	32	R/W	0000_0000h	12.4.55/539
8E4C	Memory error attributes capture (DDR_CAPTURE_ATTRIBUTES)	32	R/W	0000_0000h	12.4.56/541

Table continues on the next page...

DDR memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8E50	Memory error address capture (DDR_CAPTURE_ADDRESS)	32	R/W	0000_0000h	12.4.57/542
8E54	Memory error extended address capture (DDR_CAPTURE_EXT_ADDRESS)	32	R/W	0000_0000h	12.4.58/543
8E58	Single-Bit ECC memory error management (DDR_ERR_SBE)	32	R/W	0000_0000h	12.4.59/543

12.4.1 Chip select n memory bounds (DDR_CS*n*_BNDS)

The chip select bounds registers (CS*n*_BNDS) define the starting and ending address of the memory space that corresponds to the individual chip selects. Note that the size specified in CS*n*_BNDS should equal the size of physical DRAM. Also, note that EA*n* must be greater than or equal to SA*n*.

If chip select interleaving is enabled, all fields in the lower interleaved chip select will be used, and the other chip selects' bounds registers will be unused. For example, if chip selects 0 and 1 are interleaved, all fields in CS0_BNDS will be used, and all fields in CS1_BNDS will be unused.

Address: 8000h base + 0h offset + (8d × i), where i=0d to 3d

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	SAn																EAn															
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

DDR_CS*n*_BNDS field descriptions

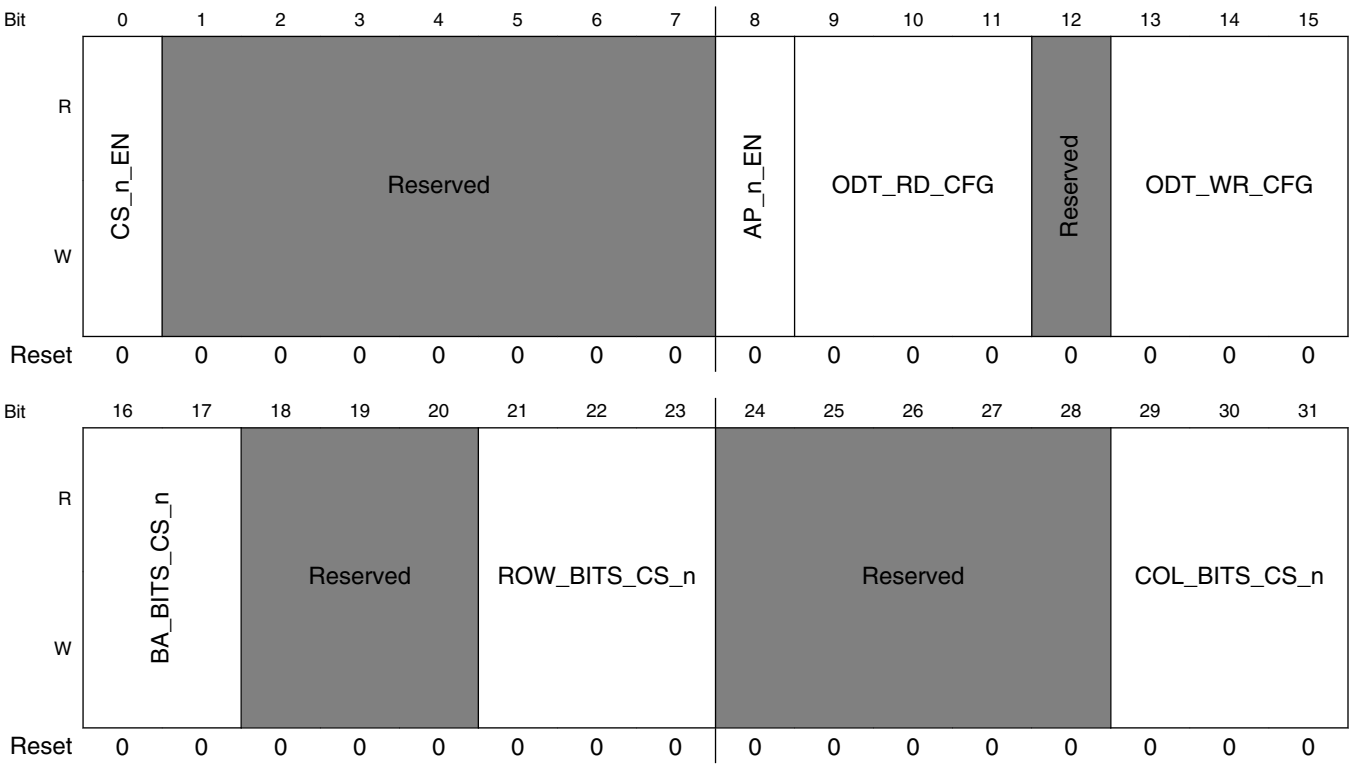
Field	Description
0–15 SAn	Starting address for chip select (bank) *n. This value is compared against the 16 msbs of the 40-bit address.*
16–31 EAn	Ending address for chip select (bank) n. This value is compared against the *16* msbs of the 40-bit address.

12.4.2 Chip select n configuration (DDR_CS*n*_CONFIG)

The chip select configuration registers (CS*n*_CONFIG) enable the DDR chip selects and set the number of row and column bits used for each chip select. These registers should be loaded with the correct number of row and column bits for each SDRAM. Because CS*n*_CONFIG[ROW_BITS_CS_*n*, COL_BITS_CS_*n*] establish address multiplexing, the user should take great care to set these values correctly.

If chip select interleaving is enabled, then all fields in the lower interleaved chip select will be used, and the other registers' fields will be unused, with the exception of the ODT_RD_CFG and ODT_WR_CFG fields. For example, if chip selects 0 and 1 are interleaved, all fields in CS0_CONFIG will be used, but only the ODT_RD_CFG and ODT_WR_CFG fields in CS1_CONFIG will be used.

Address: 8000h base + 80h offset + (4d × i), where i=0d to 3d



DDR_CS*n*_CONFIG field descriptions

Field	Description
0 CS_n_EN	Chip select n enable 0 Chip select *n*is not active 1 Chip select *n*is active and assumes the state set in CS*n*_BNDS.
1–7 -	This field is reserved. Reserved
8 AP_n_EN	Chip select n auto-precharge enable 0 Chip select *n*will only be auto-precharged if global auto-precharge mode is enabled (DDR_SDRAM_INTERVAL[BSTOPRE] = 0). 1 Chip select *n*will always issue an auto-precharge for read and write transactions.
9–11 ODT_RD_CFG	ODT for reads configuration. Note that CAS latency plus additive latency must be at least 3 cycles for ODT_RD_CFG to be enabled. Values not shown are reserved. 000 Never assert ODT for reads 001 Assert ODT only during reads to CS*n*

Table continues on the next page...

DDR_CS*n*_CONFIG field descriptions (continued)

Field	Description
	010 Assert ODT only during reads to other chip selects 011 Assert ODT only during reads to other DIMM modules. It is assumed that CS0 and CS1 are on the same DIMM module, whereas CS2 and CS3 are on a separate DIMM module. 100 Assert ODT for all reads 101 Assert ODT only during transactions to same DIMM 110 Assert ODT only during transactions to own CS and other DIMM. 111 Assert ODT only during transactions to other CS in same DIMM.
12 -	This field is reserved. Reserved
13–15 ODT_WR_CFG	ODT for writes configuration. Note that write latency plus additive latency must be at least 3 cycles for ODT _WR_CFG to be enabled. Values not shown are reserved. 000 Never assert ODT for writes 001 Assert ODT only during writes to CS*n* 010 Assert ODT only during writes to other chip selects 011 Assert ODT only during writes to other DIMM modules. It is assumed that CS0 and CS1 are on the same DIMM module, whereas CS2 and CS3 are on a separate DIMM module. 100 Assert ODT for all writes 101 Assert ODT only during transactions to same DIMM 110 Assert ODT only during transactions to own CS and other DIMM. 111 Assert ODT only during transactions to other CS in same DIMM.
16–17 BA_BITS_CS_n	Number of bank bits for SDRAM on chip select*n*. These bits correspond to the sub-bank bits driven on MBA*n* in Table 12-80 and Table 12-81. 00 2 logical bank bits 01 3 logical bank bits 10-11 Reserved
18–20 -	This field is reserved. Reserved
21–23 ROW_BITS_CS_n	Number of row bits for SDRAM on chip select*n*. See Table 12-80 and Table 12-81 for details. Values not shown are reserved. 000 12 row bits 001 13 row bits 010 14 row bits 011 15 row bits 100 16 row bits
24–28 -	This field is reserved. Reserved
29–31 COL_BITS_CS_n	Number of column bits for SDRAM on chip select n. For DDR, the decoding is as follows: 000 8 column bits 001 9 column bits 010 10 column bits 011 11 column bits 100-111 Reserved

12.4.3 Chip select n configuration 2 (DDR_CS*n*_CONFIG_2)

The chip select configuration (CS*n*_CONFIG_2) registers enable the partial array self refresh address decode in each chip select.

If chip select interleaving is enabled, then all fields in the lower interleaved chip select will be used, and the other registers' fields will be unused.

Address: 8000h base + C0h offset + (4d × i), where i=0d to 3d

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

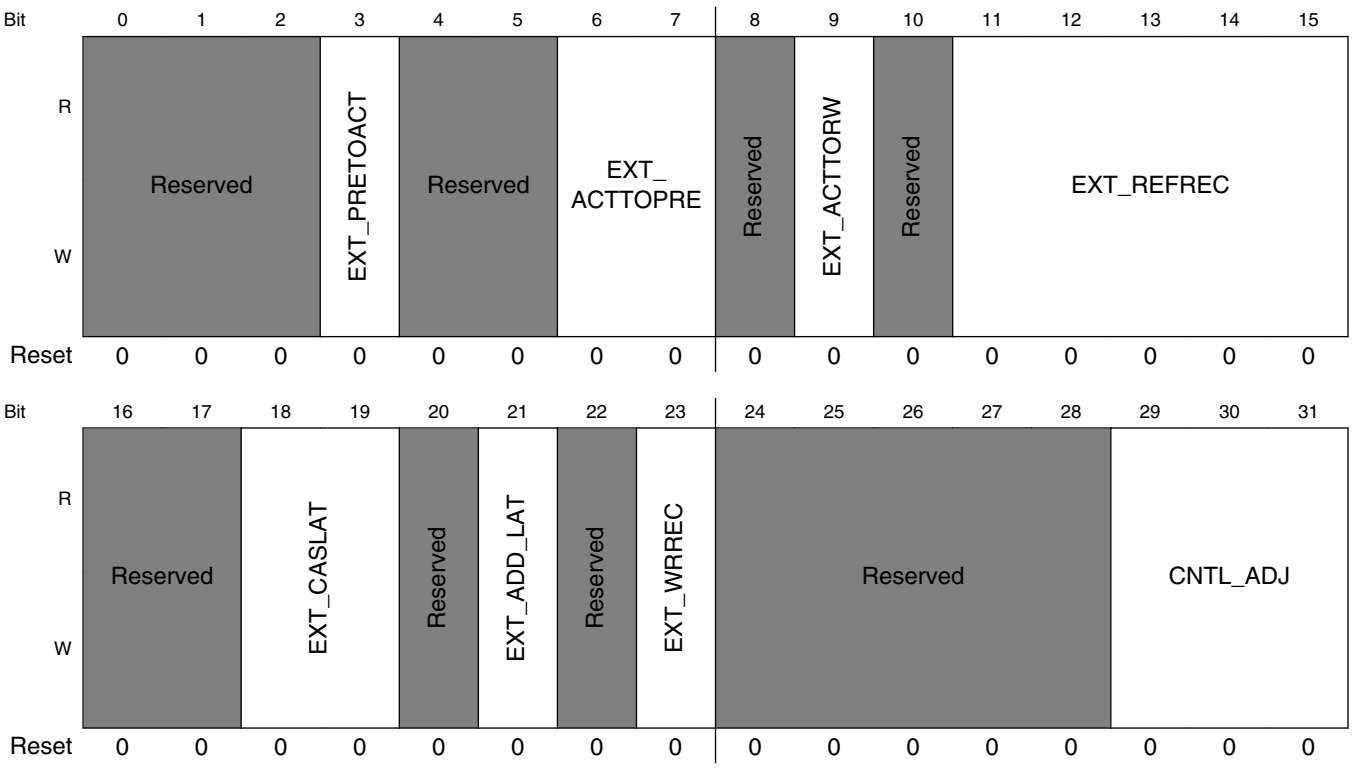
DDR_CS*n*_CONFIG_2 field descriptions

Field	Description
0 PASR_DEC	Partial array decoding. This bit can be used to enable the same address decoding used for partial array self refresh without modifying the values written to the DRAM mode registers. If this decoding is desired without enabling partial array self refresh, then this bit should be used. Note that if PASR_CFG is a non-zero value, then this bit will be ignored. 0 Normal address decoding is used, unless PASR_CFG is non-zero. 1 Address decoding typically used for partial array self refresh will be used (Refer to DDR SDRAM Address Multiplexing for details on decoding).
1–4 -	This field is reserved. Reserved
5–7 PASR_CFG	Partial array self refresh config. Controls the bits that will be placed on MA[2:0] during the write to the EMRS(2) register when the automatic hardware DRAM initialization is used (DDR_SDRAM_CFG[BI] is cleared when DDR_SDRAM_CFG[MEM_EN] is set). If this field is a non-zero value, then it will override the least significant 3 bits in DDR_SDRAM_MODE_2[ESDMODE2] during the automatic initialization for chip select n. In addition, if a non-zero value is programmed in this field, then the address decode for chip select n will be optimized for partial array self refresh, as shown in DDR SDRAM Address Multiplexing. 000 Partial array self refresh is disabled 001-111 Partial array self refresh is enabled per JEDEC specifications. Overriding the least significant 3 bits of EMRS or EMRS(2) is only supported for DDR3/3L memory types.
8–31 -	This field is reserved. Reserved

12.4.4 DDR SDRAM timing configuration 3 (DDR_TIMING_CFG_3)

DDR SDRAM timing configuration register 3 sets the extended refresh recovery time, which is combined with TIMING_CFG_1[REFREC] to determine the full refresh recovery time.

Address: 8000h base + 100h offset = 8100h



DDR_TIMING_CFG_3 field descriptions

Field	Description
0–2 -	This field is reserved. Reserved
3 EXT_PRETOACT	Extended precharge-to-activate interval (t_{RP}). Determines the number of clock cycles from a precharge command until an activate or refresh command is allowed. This field is concatenated with TIMING_CFG_1[PRETOACT] to obtain a 5-bit value for the total precharge to activate time. 0 0 clocks 1 16 clocks
4–5 -	This field is reserved. Reserved
6–7 EXT_ACTTOPRE	Extended Activate to precharge interval (t_{RAS}). Determines the number of clock cycles from an activate command until a precharge command is allowed. This field is concatenated with

Table continues on the next page...

DDR_TIMING_CFG_3 field descriptions (continued)

Field	Description
	TIMING_CFG_1[ACTTOPRE] to obtain a 6-bit value for the total activate to precharge. Note that a 6-bit value of 00_0000 is the same as a 6-bit value of 01_0000. Both values represent 16 cycles. 00 0 clocks 01 16 clocks 10 32 clocks 11 48 clocks
8 -	This field is reserved. Reserved
9 EXT_ACTTORW	Extended activate to read/write interval for SDRAM (t_{RCD}). Controls the number of clock cycles from an activate command until a read or write command is allowed. This field is concatenated with TIMING_CFG_1[ACTTORW] to obtain a 5-bit value for the total activate to read/write time.
10 -	This field is reserved. Reserved
11–15 EXT_REFREC	Extended refresh recovery time (t_{RFC}). Controls the number of clock cycles from a refresh command until an activate command is allowed. This field is concatenated with TIMING_CFG_1[REFREC] to obtain an 9-bit value for the total refresh recovery. Note that hardware adds an additional 8 clock cycles to the final, 9-bit value of the refresh recovery. t_{RFC} = {EXT_REFREC || REFREC} + 8, such that t_{RFC} is calculated as follows: 00000 0 clocks 00001 16 clocks 00010 32 clocks 00011 48 clocks 00100 64 clocks 00101 80 clocks 00110 96 clocks 00111 112 clocks 01000 128 clocks 01001 144 clocks 01010 160 clocks 01011 176 clocks 01100 192 clocks 01101 208 clocks 01110 224 clocks 01111 240 clocks 10000 256 clocks 10001 272 clocks 10010 288 clocks 10011 304 clocks 10100 320 clocks 10101 336 clocks 10110 352 clocks 10111 368 clocks 11000 384 clocks 11001 400 clocks 11010 416 clocks 11011 432 clocks

Table continues on the next page...

DDR_TIMING_CFG_3 field descriptions (continued)

Field	Description
	11100 448 clocks 11101 464 clocks 11110 480 clocks 11111 496 clocks
16–17 -	This field is reserved. Reserved
18–19 EXT_CASLAT	Extended MCAS_B latency from READ command. Number of clock cycles between registration of a READ command by the SDRAM and the availability of the first output data. If a READ command is registered at clock edge n and the latency is m clocks, data is available nominally coincident with clock edge *n* + *m*. This field is concatenated with TIMING_CFG_1[CASLAT] to obtain a 5-bit value for the total CAS latency. Note that the value of this field is added to the programmed value in TIMING_CFG_1[CASLAT]. The largest total CAS latency supported is 20 clocks. 00 0 clocks 01 8 clocks 10 16 clocks 11 Reserved
20 -	This field is reserved. Reserved
21 EXT_ADD_LAT	Extended Additive Latency. The additive latency must be set to a value less than TIMING_CFG_1[ACTTORW]. Note that the value of this field is added to the programmed value in TIMING_CFG_2[ADD_LAT]. The largest total additive latency supported is 19 clocks. 0 0 clocks 1 16 clocks
22 -	This field is reserved. Reserved
23 EXT_WRREC	Extended last data to precharge minimum interval (t_{WR}). Determines the number of clock cycles from the last data associated with a write command until a precharge command is allowed. If DDR_SDRAM_CFG_2[OBC_CFG] is set, then this field needs to be programmed to (t_{WR} + 2 cycles). This field is concatenated with TIMING_CFG_1[WRREC] to obtain a 5-bit value for the total write recovery time. 0 0 clocks 1 16 clocks
24–28 -	This field is reserved. Reserved
29–31 CNTL_ADJ	Control Adjust. Controls the amount of delay to add to the lightly loaded control signals w/ respect to all other DRAM address and command signals. The signals affected by this field are MODT[0:3], MCS[0:3]_B, and MCKE[0:3] 000 MODT[0:3], MCS[0:3]_B, and MCKE[0:3] will be launched aligned with the other DRAM address and control signals. 001 MODT[0:3], MCS[0:3]_B, and MCKE[0:3] will be launched 1/2 platform cycle later than the other DRAM address and control signals. 010 MODT[0:3], MCS[0:3]_B, and MCKE[0:3] will be launched 1 platform cycle later than the other DRAM address and control signals. 011 MODT[0:3], MCS[0:3]_B, and MCKE[0:3] will be launched 3/2 platform cycles later than the other DRAM address and control signals.

Table continues on the next page...

DDR_TIMING_CFG_3 field descriptions (continued)

Field	Description
	100 MODT[0:3], MCS[0:3]_B, and MCKE[0:3] will be launched 2 platform cycles later than the other DRAM address and control signals. 101 MODT[0:3], MCS[0:3]_B, and MCKE[0:3] will be launched 5/2 platform cycles later than the other DRAM address and control signals. 110-111 Reserved

12.4.5 DDR SDRAM timing configuration 0 (DDR_TIMING_CFG_0)

DDR SDRAM timing configuration register 0 sets the number of clock cycles between various SDRAM control commands.

Address: 8000h base + 104h offset = 8104h

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	RWT		WRT		RRT		WWT		ACT_PD_EXIT				PRE_PD_EXIT				EXT_PRE_PD_EXIT		Reserved									MRS_CYC				
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	1

DDR_TIMING_CFG_0 field descriptions

Field	Description
0–1 RWT	Read-to-write turnaround (t_{RTW}). Specifies how many extra cycles will be added between a read to write turnaround. If 0 clocks is chosen, then the DDR controller will use a fixed number based on the CAS latency and write latency. Choosing a value other than 0 adds extra cycles past this default calculation. As a default the DDR controller will determine the read-to-write turnaround as CL - WL + BL/2 + 2. In this equation, CL is the CAS latency rounded up to the next integer, WL is the programmed write latency, and BL is the burst length. This field is concatenated with TIMING_CFG_4[EXT_RWT] to obtain a 3-bit value for the total read-to-write turnaround 00 0 clocks 01 1 clock 10 2 clocks 11 3 clocks
2–3 WRT	Write-to-read turnaround. Specifies how many extra cycles will be added between a write to read turnaround. If 0 clocks is chosen, then the DDR controller will use a fixed number based on the, read latency, and write latency. Choosing a value other than 0 adds extra cycles past this default calculation. As a default, the DDR controller will determine the write-to-read turnaround as WL - CL + BL/2 + 1. In this equation, CL is the CAS latency rounded down to the next integer, WL is the programmed write latency, and BL is the burst length. This field is concatenated with TIMING_CFG_4[EXT_WRT] to obtain a 3-bit value for the total write-to-read turnaround 00 0 clocks 01 1 clock

Table continues on the next page...

DDR_TIMING_CFG_0 field descriptions (continued)

Field	Description
	10 2 clocks 11 3 clocks
4–5 RRT	Read-to-read turnaround. Specifies how many extra cycles will be added between reads to different chip selects. As a default, 3 cycles will be required between read commands to different chip selects. Extra cycles may be added with this field. Note: If 8-beat bursts are enabled, then 5 cycles will be the default. This field is concatenated with TIMING_CFG_4[EXT_RRT] to obtain a 3-bit value for the total read-to-read turnaround. 00 0 clocks 01 1 clock 10 2 clocks 11 3 clocks
6–7 WWT	Write-to-write turnaround. Specifies how many extra cycles will be added between writes to different chip selects. As a default, 2 cycles will be required between write commands to different chip selects. Extra cycles may be added with this field. Note: If 8-beat bursts are enabled, then 4 cycles will be the default. This field is concatenated with TIMING_CFG_4[EXT_WWT] to obtain a 3-bit value for the total write-to-write turnaround 00 0 clocks 01 1 clock 10 2 clocks 11 3 clocks
8–11 ACT_PD_EXIT	Active powerdown exit timing (t_{XP}). Specifies how many clock cycles to wait after exiting active powerdown before issuing any command. 0000 Reserved 0001 1 clock 0010 2 clocks 0011 3 clocks 0100 4 clocks 0101 5 clocks 0110 6 clocks 0111 7 clocks 1100 8 clocks 1101 9 clocks 1110 10 clocks 1111 11 clocks 1100 12 clocks 1101 13 clocks 1110 14 clocks 1111 15 clocks
12–15 PRE_PD_EXIT	Precharge powerdown exit timing (t_{XP}). Specifies how many clock cycles to wait after exiting precharge powerdown before issuing any command. This field is concatenated with TIMING_CFG_0[EXT_PRE_PD_EXIT] to obtain a 6-bit value for the total precharge powerdown exit timing. 0000 Reserved 0001 1 clock 0010 2 clocks

Table continues on the next page...

DDR_TIMING_CFG_0 field descriptions (continued)

Field	Description
	0011 3 clocks 0100 4 clocks 0101 5 clocks 0110 6 clocks 0111 7 clocks 1000 8 clocks 1001 9 clocks 1010 10 clocks 1011 11 clocks 1100 12 clocks 1101 13 clocks 1110 14 clocks 1111 15 clocks
16–17 EXT_PRE_PD_EXIT	Extended precharge powerdown exit timing (t_{XP}). Specifies how many clock cycles to wait after exiting precharge powerdown before issuing any command. Note the decoding for this field is not a straight decode. This field is concatenated with TIMING_CFG_0[PRE_PD_EXIT] to obtain a 6-bit value for the total precharge powerdown exit timing. 00 0 clocks 01 16 clocks 10 32 clocks 11 48 clocks
18–26 -	This field is reserved. Reserved
27–31 MRS_CYC	Mode register set cycle time (t_{MRD}, t_{MOD}). Specifies the number of cycles that must pass after a Mode Register Set command until any other command. This should be set to the greater of $t_{MRS\ and\ tMOD)}$. 00000 Reserved 00001 1 clock 00010 2 clocks 00011 3 clocks 00100 4 clocks 00101 5 clocks 00110 6 clocks 00111 7 clocks 01000 8 clocks 01001 9 clocks 01010 10 clocks 01011 11 clocks 01100 12 clocks 01101 13 clocks 01110 14 clocks 01111 15 clocks 10000 16 clocks 10001 17 clocks 10010 18 clocks 10011 19 clocks 10100 20 clocks

Table continues on the next page...

DDR_TIMING_CFG_0 field descriptions (continued)

Field	Description
	10101 21 clocks 10110 22 clocks 10111 23 clocks 11000 24 clocks 11001 25 clocks 11010 26 clocks 11011 27 clocks 11100 28 clocks 11101 29 clocks 11110 30 clocks 11111 31 clocks

12.4.6 DDR SDRAM timing configuration 1 (DDR_TIMING_CFG_1)

DDR SDRAM timing configuration register 1 sets the number of clock cycles between various SDRAM control commands.

Address: 8000h base + 108h offset = 8108h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	PRETOACT				ACTTOPRE				ACTTORW				CASLAT			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	REFREC				WRREC				ACTTOACT				WRTORD			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDR_TIMING_CFG_1 field descriptions

Field	Description
0–3 PRETOACT	Precharge-to-activate interval (t_{RP}). Determines the number of clock cycles from a precharge command until an activate or refresh command is allowed. This field is concatenated with TIMING_CFG_3[EXT_PRETOACT] to obtain a 5-bit value for the total precharge to activate time. 0000 Reserved 0001 1 clock 0010 2 clocks 0011 3 clocks 0100 4 clocks 0101 5 clocks 0110 6 clocks 0111 7 clocks 1000 8 clocks 1001 9 clocks 1010 10 clocks 1011 11 clocks 1100 12 clocks 1101 13 clocks 1110 14 clocks 1111 15 clocks
4–7 ACTTOPRE	Activate to precharge interval (t_{RAS}). Determines the number of clock cycles from an activate command until a precharge command is allowed. This field is concatenated with TIMING_CFG_3[EXT_ACTTOPRE] to obtain a 6-bit value for the total activate to precharge time. Note that the decode of 0000-0011 is equal to 16-19 clocks when TIMING_CFG_3[EXT_ACTTOPRE] = 00, but it is equal to 0-3 clocks when TIMING_CFG_3[EXT_ACTTOPRE] = 01, 10, or 11. 0000 16 clocks 0001 17 clocks 0010 18 clocks 0011 19 clocks 0100 4 clocks 0101 5 clocks 0110 6 clocks 0111 7 clocks ... 1111 15 clocks
8–11 ACTTORW	Activate to read/write interval for SDRAM (t_{RCD}). Controls the number of clock cycles from an activate command until a read or write command is allowed. This field is concatenated with TIMING_CFG_3[EXT_ACTTORW] to obtain a 5-bit value for the total activate to read/write time. 0000 Reserved 0001 1 clock 0010 2 clocks 0011 3 clocks 0100 4 clocks 0101 5 clocks 0110 6 clocks 0111 7 clocks

Table continues on the next page...

DDR_TIMING_CFG_1 field descriptions (continued)

Field	Description
	1000 8 clocks 1001 9 clocks 1010 10 clocks 1011 11 clocks 1100 12 clocks 1101 13 clocks 1110 14 clocks 1111 15 clocks
12–14 CASLAT	MCAS_B latency from READ command. Number of clock cycles between registration of a READ command by the SDRAM and the availability of the first output data. If a READ command is registered at clock edge n and the latency is m clocks, data is available nominally coincident with clock edge *n* + *m*. This field is concatenated with TIMING_CFG_3[EXT_CASLAT] to obtain a 5-bit value for the total CAS latency. This value must be programmed at initialization as described in DDR SDRAM control configuration 2 (DDR_DDR_SDRAM_CFG_2).") Note that the largest total CAS latency supported is 20 clocks. 000 1 clock 001 2 clocks 010 3 clocks 011 4 clocks 100 5 clocks 101 6 clocks 110 7 clocks 111 8 clocks
15 -	This field is reserved. Reserved
16–19 REFREC	Refresh recovery time (t_{RFC}). Controls the number of clock cycles from a refresh command until an activate command is allowed. This field is concatenated with TIMING_CFG_3[EXTREFREC] to obtain a 9-bit value for the total refresh recovery. Note that hardware adds an additional 8 clock cycles to the final, 9-bit value of the refresh recovery, such that t_{RFC} is calculated as follows: t_{RFC} = {EXT_REFREC || REFREC} + 8. 0000 8 clocks 0001 9 clocks 0010 10 clocks 0011 11 clocks ... 1111 23 clocks
20–23 WRREC	Last data to precharge minimum interval (t_{WR}). Determines the number of clock cycles from the last data associated with a write command until a precharge command is allowed. If DDR_SDRAM_CFG_2[OBC_CFG] is set, then this field needs to be programmed to (t_{WR} + 2 cycles). This field is concatenated with TIMING_CFG_3[EXT_WRREC] to obtain a 5-bit value for the total write recovery time. 0000 Reserved 0001 1 clock 0010 2 clocks 0011 3 clocks 0100 4 clocks

Table continues on the next page...

DDR_TIMING_CFG_1 field descriptions (continued)

Field	Description
	0101 5 clocks 0110 6 clocks 0111 7 clocks 1000 8 clocks 1001 9 clocks 1010 10 clocks 1011 11 clocks 1100 12 clocks 1101 13 clocks 1110 14 clocks 1111 15 clocks
24–27 ACTTOACT	Activate-to-activate interval (t_{RRD}). Number of clock cycles from an activate command until another activate command is allowed for a different logical bank in the same physical bank (chip select). 0000 Reserved 0001 1 clock 0010 2 clocks 0011 3 clocks 0100 4 clocks 0101 5 clocks 0110 6 clocks 0111 7 clocks 1000 8 clocks 1001 9 clocks 1010 10 clocks 1011 11 clocks 1100 12 clocks 1101 13 clocks 1110 14 clocks 1111 15 clocks
28–31 WRTORD	Last write data pair to read command issue interval (t_{WTR}). Number of clock cycles between the last write data pair and the subsequent read command to the same physical bank. If DDR_SDRAM_CFG_2[OBC_CFG] is set, then this field needs to be programmed to (t_{WTR} + 2 cycles). 0000 Reserved 0001 1 clock 0010 2 clocks 0011 3 clocks 0100 4 clocks 0101 5 clocks 0110 6 clocks 0111 7 clocks 1000 8 clocks 1001 9 clocks 1010 10 clocks 1011 11 clocks 1100 12 clocks 1101 13 clocks

Table continues on the next page...

DDR_TIMING_CFG_1 field descriptions (continued)

Field	Description
	1110 14 clocks 1111 15 clocks

12.4.7 DDR SDRAM timing configuration 2 (DDR_TIMING_CFG_2)

DDR SDRAM timing configuration 2 sets the clock delay to data for writes.

Address: 8000h base + 10Ch offset = 810Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ADD_LAT				Reserved				WR_LAT					Reserved		RD_TO_PRE
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	RD_TO_PRE			WR_DATA_DELAY				CKE_PLS			FOUR_ACT					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDR_TIMING_CFG_2 field descriptions

Field	Description
0–3 ADD_LAT	Additive latency. The additive latency must be set to a value less than TIMING_CFG_1[ACTTORW]. This field is added to TIMING_CFG_3]EXT_ADD_LAT]. The maximum total additive latency supported is 19 clocks. 0000 0 clocks 0001 1 clock 0010 2 clocks 0011 3 clocks 0100 4 clocks 0101 5 clocks 0110 6 clocks 0111 7 clocks 1000 8 clocks 1001 9 clocks 1010 10 clocks 1011 11 clocks 1100 12 clocks 1101 13 clocks 1110 14 clocks 1111 Reserved
4–8 -	This field is reserved. Reserved

Table continues on the next page...

DDR_TIMING_CFG_2 field descriptions (continued)

Field	Description
9–12 WR_LAT	Write latency. Note that the total write latency for DDR3 is equal to WR_LAT + ADD_LAT. Note that the total write latency must be at least 6 cycles if using unbuffered DIMMs in 1T timing mode. 0000 Reserved 0001 Reserved 0010 Reserved 0011 Reserved 0100 Reserved 0101 5 clocks 0110 6 clocks 0111 7 clocks 1000 8 clocks 1001 9 clocks 1010 10 clocks 1011 11 clocks 1100 12 clocks 1101 13 clocks 1110 14 clocks 1111 15 clocks
13–14 -	This field is reserved. Reserved
15–18 RD_TO_PRE	Read to precharge (t_{RTP}). If DDR_SDRAM_CFG_2[OBC_CFG] is set, then this field needs to be programmed to (t_{RTP} + 2 cycles). 0000 Reserved 0001 1 clock 0010 2 clocks 0011 3 clocks 0100 4 clocks 0101 5 clocks 0110 6 clocks 0111 7 clocks 1000 8 clocks 1001 9 clocks 1010 10 clocks 1011 11 clocks 1100 12 clocks 1101 13 clocks 1110 14 clocks 1111 15 clocks
19–22 WR_DATA_ DELAY	Write command to write data strobe timing adjustment. Controls the amount of delay applied to the data and data strobes for writes. See DDR SDRAM Write Timing Adjustments for details. The write preamble will typically be driven high for 1/2 DRAM cycle, and then it will be driven low for 1/2 DRAM cycle. However, for WR_DATA_DELAY settings of 0 clocks and 1/4 clocks, the write preamble will be driven low for the entire DRAM cycle. If the preamble needs to switch high first (to meet DDR3 specifications), then these values should not be used. 0000 0 clock delay 0001 2 clock delay

Table continues on the next page...

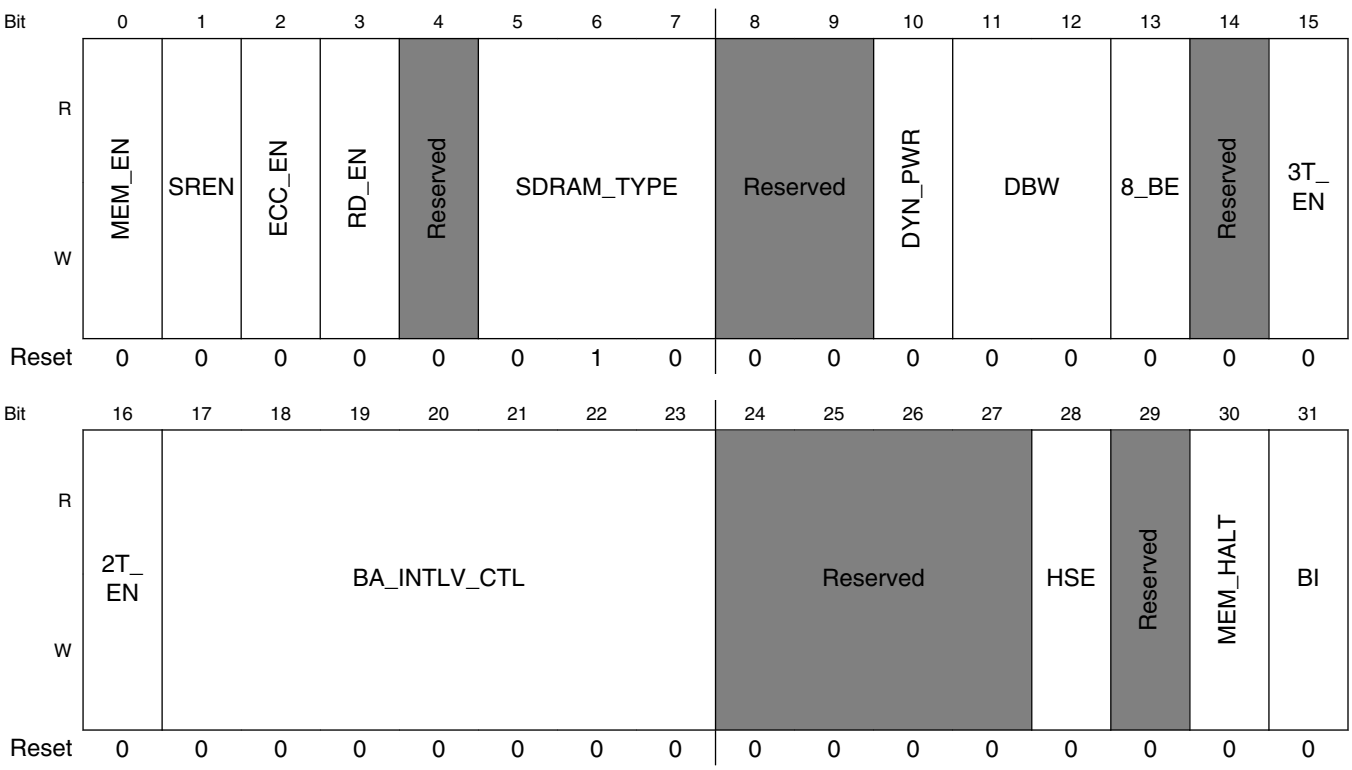
DDR_TIMING_CFG_2 field descriptions (continued)

Field	Description
	0010 1/4 clock delay 0011 9/4 clock delay 0100 1/2 clock delay 0101 5/2 clock delay 0110 3/4 clock delay 0111 Reserved 1000 1 clock delay 1001 Reserved 1010 5/4 clock delay 1011 Reserved 1100 3/2 clock delay 1101 Reserved 1110 7/4 clock delay 1111 Reserved
23–25 CKE_PLS	Minimum CKE pulse width (t_{CKE}). 000 8 clocks 001 1 clock 010 2 clocks 011 3 clocks 100 4 clocks 101 5 clocks 110 6 clocks 111 7 clocks
26–31 FOUR_ACT	Window for four activates (t_{FAW}). This is applied to DDR3 with eight logical banks only. 000000 Reserved 000001 1 cycle 000010 2 cycles 000011 3 cycles 000100 4 cycles ... 011110 30 cycles 011111 31 cycles 100000 32 cycles 111110 62 cycles 111111 63 cycles

12.4.8 DDR SDRAM control configuration (DDR_DDR_SDRAM_CFG)

The DDR SDRAM control configuration register enables the interface logic and specifies certain operating features such as self refreshing, error checking and correcting, registered DIMMs, and dynamic power management.

Address: 8000h base + 110h offset = 8110h



DDR_DDR_SDRAM_CFG field descriptions

Field	Description
0 MEM_EN	DDR SDRAM interface logic enable. **NOTE:** The DDRC must be programmed before the corresponding Local Access Windows are programmed and enabled for the DDR target(s). 0 SDRAM interface logic is disabled. 1 SDRAM interface logic is enabled. Must not be set until all other memory configuration parameters have been appropriately configured by initialization code.
1 SREN	Self refresh enable (during sleep). 0 SDRAM self refresh is disabled during sleep. Whenever self-refresh is disabled, the system is responsible for preserving the integrity of SDRAM during sleep. 1 SDRAM self refresh is enabled during sleep.

Table continues on the next page...

DDR_DDR_SDRAM_CFG field descriptions (continued)

Field	Description
2 ECC_EN	ECC enable. Note that uncorrectable read errors may cause an interrupt. 0 No ECC errors are reported. No ECC interrupts are generated. 1 ECC is enabled.
3 RD_EN	Registered DIMM enable. Specifies the type of DIMM used in the system. Note that RD_EN and 2T_EN must not both be set at the same time. 0 Indicates unbuffered DIMMs. 1 Indicates registered DIMMs.
4 -	This field is reserved. Reserved
5–7 SDRAM_TYPE	Type of SDRAM device to be used. This field will be used when issuing the automatic hardware initialization sequence to DRAM via Mode Register Set and Extended Mode Register Set commands. Default value is 111 designating DDR3 SDRAM. 000-110 Reserved 111 DDR3 SDRAM
8–9 -	This field is reserved. Reserved
10 DYN_PWR	Dynamic power management mode 0 Dynamic power management mode is disabled. 1 Dynamic power management mode is enabled. If there is no ongoing memory activity, the SDRAM CKE signal is negated.
11–12 DBW	DRAM data bus width. 00 64-bit bus is used. 01 32-bit bus is used. 10 Reserved 11 Reserved
13 8_BE	8-beat burst enable. 0 4-beat bursts are used on the DRAM interface. This is only supported if DDR_SDRAM_CFG_2[OBC_CFG] is also set. 1 8-beat bursts are used on the DRAM interface.
14 -	This field is reserved. Reserved
15 3T_EN	Enable 3T timing. This field cannot be set if DDR_SDRAM_CFG[2T_EN] is also set. This field cannot be used with a 32-bit bus if 4-beat bursts are used. 0 1T timing is enabled if 2T_EN is cleared. The DRAM command/address are held for only 1 cycle on the DRAM bus. 1 3T timing is enabled. The DRAM command/address are held for 3 full cycles on the DRAM bus for every DRAM transaction. However, the chip select is only held for the third cycle. **NOTE:** 3T timing may not be used with 4-beat bursts, unless DDR_SDRAM_CFG_2[OBC_CFG] is set.
16 2T_EN	Enable 2T timing. This field should not be set if DDR_SDrAM_CFG[3T_EN] is set. Note that RD_EN and 2T_EN must not both be set at the same time.

Table continues on the next page...

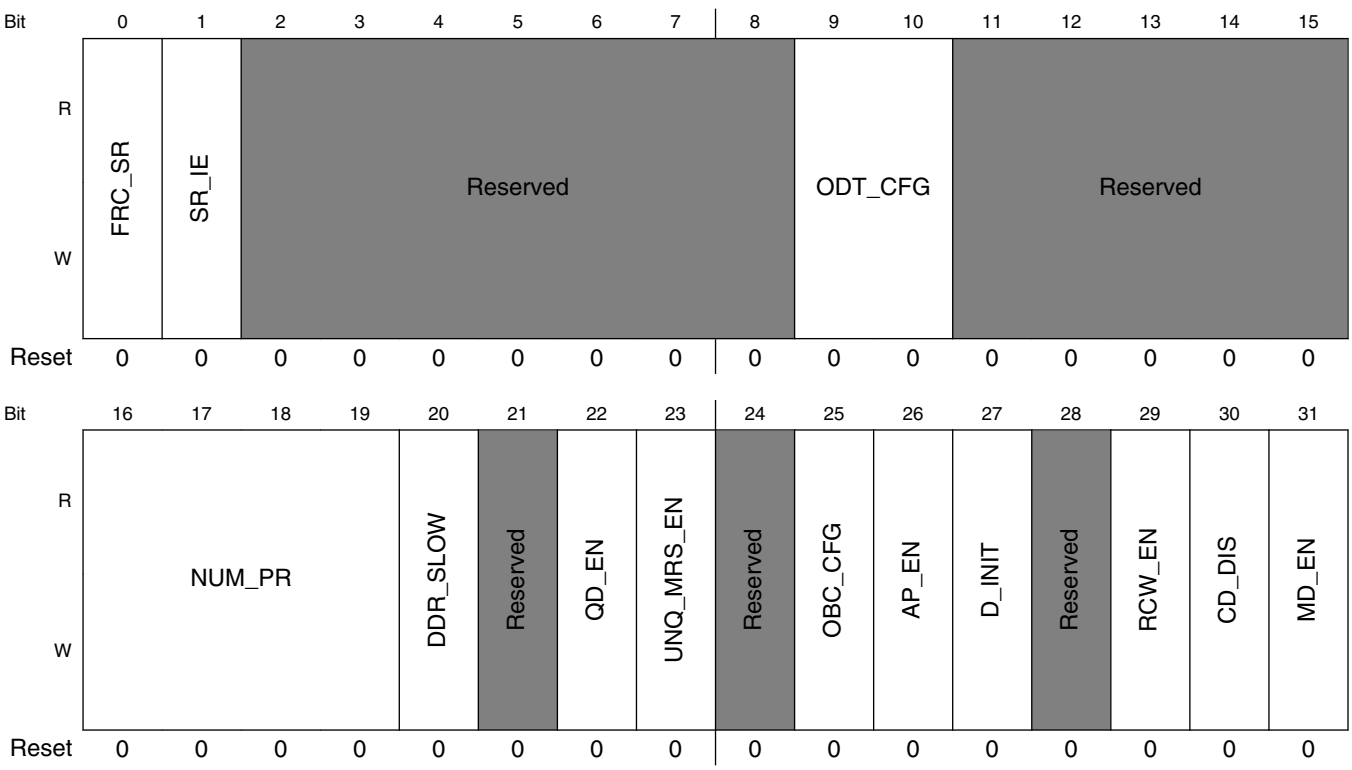
DDR_DDR_SDRAM_CFG field descriptions (continued)

Field	Description
	0 1T timing is enabled if 3T_EN is cleared. The DRAM command/address are held for only 1 cycle on the DRAM bus. 1 2T timing is enabled. The DRAM command/address are held for 2 full cycles on the DRAM bus for every DRAM transaction. However, the chip select is only held for the second cycle.
17–23 BA_INTLV_CTL	Bank (chip select) interleaving control. Set this field only if you wish to use bank interleaving. ('*x*' denotes a don't care bit value. All unlisted field values are reserved.) 0000000 No external memory banks are interleaved 1000000 External memory banks 0 and 1 are interleaved 0100000 External memory banks 2 and 3 are interleaved 1100000 External memory banks 0 and 1 are interleaved together and banks 2 and 3 are interleaved together xx00100 External memory banks 0 through 3 are all interleaved together
24–27 -	This field is reserved. Reserved
28 HSE	Global half-strength override S Sets I/O driver impedance to half strength. This impedance will be used by the MDIC, address/command, data, and clock impedance values, but only if automatic hardware calibration is disabled and the corresponding group's software override is disabled in the DDR control driver register(s) described in DDR Control Driver Register 1 (DDR_DDRCDR_1) and DDR Control Driver Register 2 (DDR_DDRCDR_2)." This bit should be cleared if using automatic hardware calibration. 0 I/O driver impedance will be configured to full strength. 1 I/O driver impedance will be configured to half strength.
29 -	This field is reserved. Reserved
30 MEM_HALT	DDR memory controller halt. When this bit is set, the memory controller will not accept any new data read/write transactions to DDR SDRAM until the bit is cleared again. This can be used when bypassing initialization and forcing MODE REGISTER SET commands through software. 0 DDR controller will accept new transactions. 1 DDR controller will finish any remaining transactions, and then it will remain halted until this bit is cleared by software.
31 BI	Bypass initialization See DDR training initialization address (DDR_DDR_INIT_ADDR) for details on avoiding ECC errors in this mode. 0 DDR controller will cycle through initialization routine based on SDRAM_TYPE 1 Initialization routine will be bypassed. Software is responsible for initializing memory through DDR_SDRAM_MD_CTRL register. If software is initializing memory, then the MEM_HALT bit can be set to prevent the DDR controller from issuing transactions during the initialization sequence. Note that the DDR controller will not issue a DLL reset to the DRAMs when bypassing the initialization routine, regardless of the value of DDR_SDRAM_CFG[DLL_RST_DIS]. If a DLL reset is required, then the controller should be forced to enter and exit self refresh after the controller is enabled.

12.4.9 DDR SDRAM control configuration 2 (DDR_DDR_SDRAM_CFG_2)

The DDR SDRAM control configuration register 2 provides more control configuration for the DDR controller.

Address: 8000h base + 114h offset = 8114h



DDR_DDR_SDRAM_CFG_2 field descriptions

Field	Description
0 FRC_SR	Force self refresh 0 DDR controller will operate in normal mode. 1 DDR controller will enter self-refresh mode.
1 SR_IE	Self-refresh interrupt enable. The DDR controller can be placed into self refresh mode by forcing the PIC to assert *sie0*. This is considered a panic interrupt by the DDR controller, and it will enter self refresh as soon as possible. DDR_SDRAM_CFG[SREN] must also be set if the panic interrupt will be used. 0 DDR controller will not enter self-refresh mode if panic interrupt is asserted. 1 DDR controller will enter self-refresh mode if panic interrupt is asserted.
2–8 -	This field is reserved. Reserved

Table continues on the next page...

DDR_DDR_SDRAM_CFG_2 field descriptions (continued)

Field	Description
9–10 ODT_CFG	ODT configuration. This field defines how ODT will be driven to the on-chip IOs. See DDR Control Driver Register 1 (DDR_DDRCDR_1) and DDR Control Driver Register 2 (DDR_DDRCDR_2) which define the termination value that will be used. 00 Never assert ODT to internal IOs 01 Reserved 10 Assert ODT to internal IOs only during reads to DRAM 11 Always keep ODT asserted to internal IOs
11–15 -	This field is reserved. Reserved
16–19 NUM_PR	Number of posted refreshes. This will determine how many posted refreshes, if any, can be issued at one time. Note that if posted refreshes are used, then this field, along with DDR_SDRAM_INTERVAL[REFINT], must be programmed such that the maximum t_{ras} specification cannot be violated. 0000 Reserved 0001 1 refresh will be issued at a time 0010 2 refreshes will be issued at a time 0011 3 refreshes will be issued at a time 0100 4 refreshes will be issued at a time 0101 5 refreshes will be issued at a time 0110 6 refreshes will be issued at a time 0111 7 refreshes will be issued at a time 1000 8 refreshes will be issued at a time 1001-1111 Reserved
20 DDR_SLOW	DDR Slow Frequency. Indicates to the controller if it will be run at a lower frequency. 0 The DDR controller will be run at DDR-1250 or higher. 1 The DDR controller will be run less than DDR-1250.
21 -	Reserved This field is reserved. This bit is reserved
22 QD_EN	Quad-rank enable. Determines if a quad-ranked DIMM is used. This bit should also be set if quad-stacked discrete memory chips are used. 0 Quad-ranked DIMMs are not used. 1 Quad-ranked DIMMs are used. The controller will internally swap CKE[1] and CKE[2], along with MODT[1] and MODT[2].
23 UNQ_MRS_EN	Unique MRS Enable. Determines if the DDR_SDRAM_MODE_{3..8} registers will be used when initializing the memories for chip selects 1, 2, and 3. These can be used to provide unique values to the Mode Registers of the DRAM to allow different termination values for each rank.
24 -	This field is reserved. Reserved
25 OBC_CFG	On-The-Fly Burst Chop Configuration. Determines if on-the-fly Burst Chop will be used. This bit should only be set if DDR3 memories are used. If on-the-fly Burst Chop mode is not used with DDR3 memories, then fixed Burst Chop mode may be used if the proper turnaround times are programmed into TIMING_CFG_0 and TIMING_CFG_4. DDR_SDRAM_CFG[8_BE] should be cleared for both on-the-fly Burst Chop mode or fixed Burst Chop mode when using a 64-bit data bus with DDR3 memories.

Table continues on the next page...

DDR_DDR_SDRAM_CFG_2 field descriptions (continued)

Field	Description
	0 On-the-fly Burst Chop mode is disabled. Fixed burst lengths as defined in DDR_SDRAM_CFG[8_BE] are used. If fixed Burst Chop will be used (with DDR3 memories), then DDR_SDRAM_CFG[8_BE] should be cleared. 1 On-the-fly Burst Chop mode will be used. DDR_SDRAM_CFG[8_BE] should be cleared for on-the-fly Burst Chop mode. DDR_SDRAM_CFG[DBW] should also be cleared for on-the-fly Burst Chop mode
26 AP_EN	Address Parity Enable. Determines if address parity will be generated and checked for the address and control signals when using registered DIMMs. If address parity is used, the MAPAR_OUT and MAPAR_ERR_B pins will be used to drive the parity bit and to receive errors from the open-drain parity error signal. Even parity will be used, and parity will be generated for the MA[15:0], MBA[2:0], MRAS_B, MCAS_B, MWE_B signals. Parity will not be generated for the MCKE[0:3], MODT[0:3], or MCS[0:3]_B signals. Note that address parity should not be used for non-zero values of TIMING_CFG_3[CNTL_ADJ]. 0 Address parity will not be used 1 Address parity will be used
27 D_INIT	DRAM data initialization This bit is set by software, and it is cleared by hardware. If software sets this bit before the memory controller is enabled, the controller will automatically initialize DRAM after it is enabled. This bit will be automatically cleared by hardware once the initialization is completed. This data initialization bit should only be set when the controller is idle. 0 There is not data initialization in progress, and no data initialization is scheduled 1 The memory controller will initialize memory once it is enabled. This bit will remain asserted until the initialization is complete. The value in DDR_DATA_INIT register will be used to initialize memory.
28 -	This field is reserved. Reserved
29 RCW_EN	Register Control Word Enable. If DDR3 registered DIMMs are used, it may be necessary to write the register control words before issuing commands to DRAM. If this bit is set, the controller will write the register control words after DDR_SDRAM_CFG[MEM_EN] is set, unless DDR_SDRAM_CFG[BI] is set. The register control words will be written with the values in DDR_SDRAM_RCW_1 and DDR_SDRAM_RCW_2. 0 Register control words will not be automatically written during DRAM initialization 1 Register control words will be automatically written during DRAM initialization. This bit should only be set if DDR3 registered DIMMs are used, and the default settings need to be modified.
30 CD_DIS	Corrupted Data Disable. If this bit is set, then the corrupted data feature will be disabled. When the corrupted data feature is enabled, the DDR controller will inverted the generated ECC code for any beat of data which is known to have corrupted data. When a read to the corrupted data is later generated, the ERR_DETECT[CDE] error will be set if error reporting is enabled. 0 Corrupted data is enabled 1 Corrupted data is disabled
31 MD_EN	Mirrored DIMM Enable. Some DDR3 DIMMs will be mirrored, where certain MA and MBA pins are mirrored on one side of the DIMM. When this bit is set, the controller will know to swap these signals before transmitting to the DRAM. The controller will assume that CS1 and CS3 are the 'mirrored' ranks of memory. The following signals are mirrored (MBA[0] vs. MBA[1]; MA[3] vs. MA[4]; MA[5] vs. MA[6]; MA[7] vs. MA[8]). 0 Mirrored DIMMs are not used 1 Mirrored DIMMs are used

12.4.10 DDR SDRAM mode configuration (DDR_DDR_SDRAM_MODE)

The DDR SDRAM mode configuration register sets the values loaded into the DDR's mode registers.

Address: 8000h base + 118h offset = 8118h

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	ESDMODE																SDMODE															
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

DDR_DDR_SDRAM_MODE field descriptions

Field	Description
0–15 ESDMODE	Extended SDRAM mode. Specifies the initial value loaded into the DDR SDRAM extended mode register. The range and meaning of legal values is specified by the DDR SDRAM manufacturer. When this value is driven onto the address bus (during the DDR SDRAM initialization sequence), MA[0] presents the lsb of ESDMODE, which, in the big-endian convention shown in the figure, corresponds to ESDMODE[15]. The msb of the SDRAM extended mode register value must be stored at ESDMODE[0]. The value programmed into this field is also used for writing MR1 during write leveling for DDR3, although the bits specifically related to the write leveling scheme are handled automatically by the DDR controller. Even if DDR_SDRAM_CFG[BI] is set, this field is still used during write leveling. The write leveling enable bit should be cleared by software in this field.
16–31 SDMODE	SDRAM mode. Specifies the initial value loaded into the DDR SDRAM mode register. The range of legal values is specified by the DDR SDRAM manufacturer. When this value is driven onto the address bus (during DDR SDRAM initialization), MA[0] presents the lsb of SDMODE, which, in the big-endian convention shown in the figure, corresponds to SDMODE[15]. The msb of the SDRAM mode register value must be stored at SDMODE[0]. Because the memory controller forces SDMODE[7] to certain values depending on the state of the initialization sequence, (for resetting the SDRAM's DLL) the corresponding bits of this field are ignored by the memory controller. Note that SDMODE[7] is mapped to MA[8].

12.4.11 DDR SDRAM mode configuration 2 (DDR_DDR_SDRAM_MODE_2)

The DDR SDRAM mode 2 configuration register sets the values loaded into the DDR's extended mode 2 and 3 registers.

Address: 8000h base + 11Ch offset = 811Ch

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	ESDMODE2																ESDMODE3															
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

DDR_DDR_SDRAM_MODE_2 field descriptions

Field	Description
0–15 ESDMODE2	Extended SDRAM mode 2. Specifies the initial value loaded into the DDR SDRAM extended 2 mode register. The range and meaning of legal values is specified by the DDR SDRAM manufacturer. When this value is driven onto the address bus (during the DDR SDRAM initialization sequence), MA[0] presents the lsb bit of ESDMODE2, which, in the big-endian convention shown in the register figure, corresponds to ESDMODE2[15]. The msb of the SDRAM extended mode 2 register value must be stored at ESDMODE2[0].
16–31 ESDMODE3	Extended SDRAM mode 3. Specifies the initial value loaded into the DDR SDRAM extended 3 mode register. The range of legal values of legal values is specified by the DDR SDRAM manufacturer. When this value is driven onto the address bus (during DDR SDRAM initialization), MA[0] presents the lsb of ESDMODE3, which, in the big-endian convention shown in the figure, corresponds to ESDMODE3[15]. The msb of the SDRAM extended mode 3 register value must be stored at ESDMODE3[0].

12.4.12 DDR SDRAM mode control (DDR_DDR_SDRAM_MD_CNTL)

The DDR SDRAM mode control register allows the user to carry out the following tasks:

- Issue a mode register set command to a particular chip select
- Issue an immediate refresh to a particular chip select
- Issue an immediate precharge or precharge all command to a particular chip select
- Force the CKE signals to a specific value

Before issuing a command via the DDR_SDRAM_MD_CNTL register, the DDR interface should be idle. This can be done by setting DDR_SDRAM_CFG[MEM_HALT] and disabling refreshes by clearing DDR_INTERVAL[REFINT]. If there are memory contents that need to be preserved during this time, then software should also force any required refresh commands while DDR_INTERVAL[REFINT] is cleared.

The accompanying table shows the user how to set the fields of this register to accomplish the above tasks.

NOTE

Note that MD_EN, SET_REF, and SET_PRE are mutually exclusive; only one of these fields can be set at a time.

The following table shows how DDR_SDRAM_MD_CNTL fields should be set for each of the tasks described above.

Table 12-29. Settings of DDR_SDRAM_MD_CNTL Fields

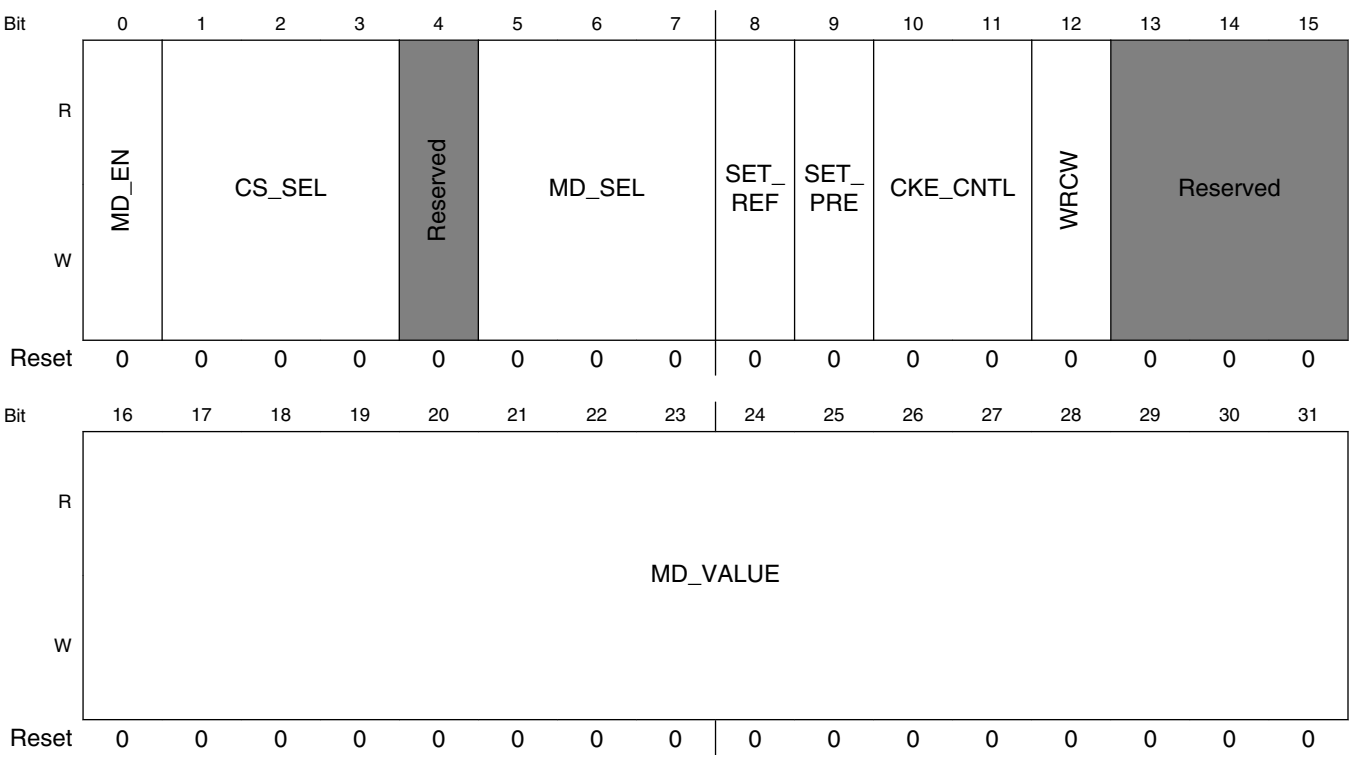
Field	Mode Register Set	Refresh	Precharge	Clock Enable Signals Control
MD_EN	1	0	0	-
SET_REF	0	1	0	-

Table continues on the next page...

Table 12-29. Settings of DDR_SDRAM_MD_CNTL Fields (continued)

Field	Mode Register Set	Refresh	Precharge	Clock Enable Signals Control
SET_PRE	0	0	1	-
CS_SEL	Chooses chip select (CS)			-
MD_SEL	Select mode register. See DDR SDRAM mode control (DDR_DDR_SDRAM_MD_CNTL).	-	Selects logical bank	-
MD_VALUE	Value written to mode register	-	Only bit 5 is significant. See DDR SDRAM mode control (DDR_DDR_SDRAM_MD_CNTL).	-
CKE_CNTL	0	0	0	See DDR SDRAM mode control (DDR_DDR_SDRAM_MD_CNTL).

Address: 8000h base + 120h offset = 8120h



DDR_DDR_SDRAM_MD_CNTL field descriptions

Field	Description
0 MD_EN	Mode enable. Setting this bit specifies that valid data in MD_VALUE is ready to be written to DRAM as one of the following commands:

Table continues on the next page...

DDR_DDR_SDRAM_MD_CNTL field descriptions (continued)

Field	Description
	• MODE REGISTER SET • EXTENDED MODE REGISTER SET • EXTENDED MODE REGISTER SET 2 • EXTENDED MODE REGISTER SET 3 The specific command to be executed is selected by setting MD_SEL. In addition, the chip select must be chosen by setting CS_SEL.MD_EN is set by software and cleared by hardware once the command has been issued. 0 Indicates that no mode register set command needs to be issued. 1 Indicates that valid data contained in the register is ready to be issued as a mode register set command.
1–3 CS_SEL	Select chip select. Specifies the chip select that will be driven active due to any command forced by software in DDR_SDRAM_MD_CNTL. 000 Chip select 0 is active 001 Chip select 1 is active 010 Chip select 2 is active 011 Chip select 3 is active 100 Chip select 0 and chip select 1 are active 101 Chip select 2and chip select 3 are active 110-111 Reserved
4 - Reserved	This field is reserved.
5–7 MD_SEL	Mode register select. MD_SEL specifies one of the following: • During a mode select command, selects the SDRAM mode register to be changed • During a precharge command, selects the SDRAM logical bank to be precharged. A precharge all command ignores this field. • During a refresh command, this field is ignored. Note that MD_SEL contains the value that will be presented onto the memory bank address pins (MBA*n*) of the DDR controller. 000 MR 001 EMR 010 EMR2 011 EMR3
8 SET_REF	Set refresh. Forces an immediate refresh to be issued to the chip select specified by DDR_SDRAM_MD_CNTL[CS_SEL]. This bit will be set by software and cleared by hardware once the command has been issued. 0 Indicates that no refresh command needs to be issued. 1 Indicates that a refresh command is ready to be issued.
9 SET_PRE	Set precharge. Forces a precharge or precharge all to be issued to the chip select specified by DDR_SDRAM_MD_CNTL[CS_SEL]. This bit will be set by software and cleared by hardware once the command has been issued. 0 Indicates that no precharge all command needs to be issued. 1 Indicates that a precharge all command is ready to be issued.
10–11 CKE_CNTL	Clock enable control. Allows software to globally clear or set the all CKE signals issued to DRAM. Once software has forced the value driven on CKE, that value will continue to be forced until software clears the

Table continues on the next page...

DDR_DDR_SDRAM_MD_CNTL field descriptions (continued)

Field	Description
	CKE_CNTL bits. At that time, the DDR controller will continue to drive the CKE signals to the same value forced by software until another event causes the CKE signals to change (that is, self refresh entry/exit, power down entry/exit). 00 CKE signals are not forced by software. 01 CKE signals are forced to a low value by software. 10 CKE signals are forced to a high value by software. 11 Reserved
12 WRCW	Write register control word. If software sets this bit, then a register control word will be written by asserting the selected chip selects while providing the programmed data on the MA and MBA signals. The RAS, CAS, and WE will remain deasserted during this write The MD_EN field should also be set to force a register control word write. This should only be set if DDR3 registered DIMM s are used, and the register needs to be configured. If DDR_SDRAM_MD_CNTL is used to write RCW2 specifically, then software must guarantee that the timing parameter, t-STAB, is met before future accesses to the controller are allowed. In addition, DDR_SDRAM_MD_CNTL register cannot be used to write the RCWs if write leveling will be used, since write leveling is run automatically before DDR_SDRAM_MD_CNTL can be used to force RCW writes. 0 Indicates that a register control word write will not be issued if MD_EN is set. 1 Indicates that a register control word write will be issued if MD_EN is set.
13–15 -	This field is reserved. Reserved
16–31 MD_VALUE	Mode register value. This field, which specifies the value that will be presented on the memory address pins of the DDR controller during a mode register set command, is significant only when this register is used to issue a mode register set command or a precharge or precharge all command. For a mode register set command, this field contains the data to be written to the selected mode register. For a precharge command, only bit five is significant: 0 Issue a precharge command; MD_SEL selects the logical bank to be precharged 1 Issue a precharge all command; all logical banks are precharged

12.4.13 DDR SDRAM interval configuration (DDR_DDR_SDRAM_INTERVAL)

The DDR SDRAM interval configuration register sets the number of DRAM clock cycles between bank refreshes issued to the DDR SDRAMs. In addition, the number of DRAM cycles that a page is maintained after it is accessed is provided here.

Address: 8000h base + 124h offset = 8124h

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

DDR_DDR_SDRAM_INTERVAL field descriptions

Field	Description
0–15 REFINT	Refresh interval. Represents the number of memory bus clock cycles between refresh cycles. Depending on DDR_SDRAM_CFG_2[NUM_PR], some number of rows are refreshed in each DDR SDRAM physical bank during each refresh cycle. The value for REFINT depends on the specific SDRAMs used and the interface clock frequency. Refreshes will not be issued when the REFINT is set to all 0s. This field is concatenated with TIMING_CFG_4[EXT_REFINT] to obtain a 17-bit value for the total refresh interval.
16–17 -	This field is reserved. Reserved
18–31 BSTOPRE	Precharge interval. Sets the duration (in memory bus clocks) that a page is retained after a DDR SDRAM access. If BSTOPRE is zero, the DDR memory controller uses auto-precharge read and write commands rather than operating in page mode. This is called global auto-precharge mode.

12.4.14 DDR SDRAM data initialization (DDR_DDR_DATA_INIT)

The DDR SDRAM data initialization register provides the value that will be used to initialize memory if DDR_SDRAM_CFG2[D_INIT] is set.

Address: 8000h base + 128h offset = 8128h

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	INIT_VALUE																															
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

DDR_DDR_DATA_INIT field descriptions

Field	Description
0–31 INIT_VALUE	Initialization value. Represents the value that DRAM will be initialized with if DDR_SDRAM_CFG2[D_INIT] is set.

12.4.15 DDR SDRAM clock control (DDR_DDR_SDRAM_CLK_CNTL)

The DDR SDRAM clock control configuration register provides a 1/16-cycle clock adjustment.

Address: 8000h base + 130h offset = 8130h

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					CLK_ADJUST					Reserved																					
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDR_DDR_SDRAM_CLK_CNTL field descriptions

Field	Description
0–4 -	This field is reserved. Reserved
5–9 CLK_ADJUST	Clock adjust 00000 Clock is launched aligned with address/command 00001 Clock is launched 1/16 applied cycle after address/command 00010 Clock is launched 1/8 applied cycle after address/command 00011 Clock is launched 3/16 applied cycle after address/command 00100 Clock is launched 1/4 applied cycle after address/command 00101 Clock is launched 5/16 applied cycle after address/command 00110 Clock is launched 3/8 applied cycle after address/command 00111 Clock is launched 7/16 applied cycle after address/command 01000 Clock is launched 1/2 applied cycle after address/command 01001 Clock is launched 9/16 applied cycle after address/command 01010 Clock is launched 5/8 applied cycle after address/command 01011 Clock is launched 11/16 applied cycle after address/command 01100 Clock is launched 3/4 applied cycle after address/command 01101 Clock is launched 13/16 applied cycle after address/command 01110 Clock is launched 7/8 applied cycle after address/command 01111 Clock is launched 15/16 applied cycle after address/command 10000 Clock is launched 1 applied cycle after address/command 10010-11110 Reserved
10–31 -	This field is reserved. Reserved

12.4.16 DDR training initialization address (DDR_DDR_INIT_ADDR)

The DDR SDRAM initialization address register provides the address that will be used for the data strobe to data skew adjustment and automatic CAS_B to preamble calibration after POR.

When the default value is used (that is, address 0x0), all chip selects are considered for the training. If DDR_INIT_ADDR is set to any value other than the default value of address zero, then only the first chip select will be trained. When multiple chip selects are used and DQS/DQ skew is not common between chip selects/ranks, then the default address value of 0x0 is be recommended to obtain the best timing margins.

After the skew adjustment, this address will contain bad ECC data. This is not important at POR, as all of memory should be subsequently initialized if ECC is enabled (either by software or through the use of DDR_SDRAM_CFG_2[D_INIT]).

If an HRESET_B has been issued after the DRAM is in self-refresh mode, however, memory is not initialized, so this address should be written to using an 8- or 32-byte transaction to avoid possible ECC errors if this address could later be accessed.

Address: 8000h base + 148h offset = 8148h

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	INIT_ADDR																															
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

DDR_DDR_INIT_ADDR field descriptions

Field	Description
0–31 INIT_ADDR	Initialization address. Represents the address that will be used for the data strobe to data skew adjustment and automatic CAS to preamble calibration at POR. This address will be written to during the initialization sequence.

12.4.17 DDR training initialization extended address (DDR_DDR_INIT_EXT_ADDRESS)

The DDR SDRAM initialization extended address register provides the extended address that will be used for the data strobe to data skew adjustment and automatic CAS_B to preamble calibration after POR.

Address: 8000h base + 14Ch offset = 814Ch

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

DDR_DDR_INIT_EXT_ADDRESS field descriptions

Field	Description
0 UIA	Use initialization address. 0 Use the default address for training sequence as calculated by the controller. This will be the first valid address in each enabled chip select. 1 Use the initialization address programmed in DDR_INIT_ADDR and DDR_INIT_EXT_ADDR.
1–23 -	This field is reserved. Reserved
24–31 INIT_EXT_ADDR	Initialization extended address. Represents the extended address that will be used for the data strobe to data skew adjustment and automatic CAS_B to preamble calibration at POR. This extended address will be written to during the initialization sequence.

12.4.18 DDR SDRAM timing configuration 4 (DDR_TIMING_CFG_4)

The DDR SDRAM timing configuration 4 register provides additional timing fields required to support DDR3 memories.

Address: 8000h base + 160h offset = 8160h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	RWT				WRT				RRT				WWT			
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	EXT_RWT		Reserved	EXT_ WRT	Reserved	EXT_RRT	Reserved	EXT_ WWT	Reserved			EXT_REFINT	Reserved		DLL_LOCK	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDR_TIMING_CFG_4 field descriptions

Field	Description
0–3 RWT	Read-to-write turnaround for same chip select. Specifies how many cycles will be added between a read to write turnaround for transactions to the same chip select. If a value of 0000 is chosen, then the DDR controller will use the value used for transactions to different chip selects, as defined in TIMING_CFG_0[RWT]. This field can be used to improve performance when operating in burst-chop mode by forcing transactions to the same chip select to use extra cycles, while transaction to different chip selects can utilize the tri-state time on the DRAM interface. Regardless of the value that is set in this field, the value defined by TIMING_CFG_0[RWT] will also be met before issuing a write command. 0000 Default 0001 1 clock 0010 2 clocks 0011 3 clocks 0100 4 clocks 0101 5 clocks 0110 6 clocks 0111 7 clocks

Table continues on the next page...

DDR_TIMING_CFG_4 field descriptions (continued)

Field	Description
	1000 8 clocks 1001 9 clocks 1010 10 clocks 1011 11 clocks 1100 12 clocks 1101 13 clocks 1110 14 clocks 1111 15 clocks
4–7 WRT	Write-to-read turnaround for same chip select. Specifies how many cycles will be added between a write to read turnaround for transactions to the same chip select. If a value of 0000 is chosen, then the DDR controller will use the value used for transactions to different chip selects, as defined in TIMING_CFG_0[WRT]. This field can be used to improve performance when operating in burst-chop mode by forcing transactions to the same chip select to use extra cycles, while transaction to different chip selects can utilize the tri-state time on the DRAM interface. Regardless of the value that is set in this field, the value defined by TIMING_CFG_0[WRT] will also be met before issuing a read command. 0000 Default 0001 1 clock 0010 2 clocks 0011 3 clocks 0100 4 clocks 0101 5 clocks 0110 6 clocks 0111 7 clocks 1000 8 clocks 1001 9 clocks 1010 10 clocks 1011 11 clocks 1100 12 clocks 1101 13 clocks 1110 14 clocks 1111 15 clocks
8–11 RRT	Read-to-read turnaround for same chip select. Specifies how many cycles will be added between reads to the same chip select. If a value of 0000 is chosen, then 2 cycles will be required between read commands to the same chip select if 4-beat bursts are used (4 cycles will be required if 8-beat bursts are used). Note that DDR3 does not support 4-beat bursts. However, this field may be used to add extra cycles when burst-chop mode is used, and the DDR controller must wait 4 cycles for read-to-read transactions to the same chip select. 0000 BL/2 clocks 0001 BL/2 + 1 clock 0010 BL/2 + 2 clocks 0011 BL/2 + 3 clocks 0100 BL/2 + 4 clocks 0101 BL/2 + 5 clocks 0110 BL/2 + 6 clocks 0111 BL/2 + 7 clocks 1000 BL/2 + 8 clocks 1001 BL/2 + 9 clocks

Table continues on the next page...

DDR_TIMING_CFG_4 field descriptions (continued)

Field	Description
	1010 BL/2 + 10 clocks 1011 BL/2 + 11 clocks 1100 BL/2 + 12 clocks 1101 BL/2 + 13 clocks 1110 BL/2 + 14 clocks 1111 BL/2 + 15 clocks
12–15 WWT	Write-to-write turnaround for same chip select. Specifies how many cycles will be added between writes to the same chip select. If a value of 0000 is chosen, then 2 cycles will be required between write commands to the same chip select if 4-beat bursts are used (4 cycles will be required if 8-beat bursts are used). Note that DDR3 does not support 4-beat bursts. However, this field may be used to add extra cycles when burst-chop mode is used, and the DDR controller must wait 4 cycles for write-to-write transactions to the same chip select. 0000 BL/2 clocks 0001 BL/2 + 1 clock 0010 BL/2 + 2 clocks 0011 BL/2 + 3 clocks 0100 BL/2 + 4 clocks 0101 BL/2 + 5 clocks 0110 BL/2 + 6 clocks 0111 BL/2 + 7 clocks 1000 BL/2 + 8 clocks 1001 BL/2 + 9 clocks 1010 BL/2 + 10 clocks 1011 BL/2 + 11 clocks 1100 BL/2 + 12 clocks 1101 BL/2 + 13 clocks 1110 BL/2 + 14 clocks 1111 BL/2 + 15 clocks
16–17 EXT_RWT	Extended read-to-write turnaround (t_{RTW}). Specifies how many extra cycles will be added between a read to write turnaround. If 0 clocks is chosen, then the DDR controller will use a fixed number based on the CAS latency and write latency. Choosing a value other than 0 adds extra cycles past this default calculation. As a default the DDR controller will determine the read-to-write turnaround as CL - WL + BL/2 + 2. In this equation, CL is the CAS latency rounded up to the next integer, WL is the programmed write latency, and BL is the burst length. This field is concatenated with TIMING_CFG_0[RWT] to obtain a 4-bit value for the total read-to-write turnaround 00 0 clocks 01 4 clocks 10 8 clocks 11 12 clocks
18 -	This field is reserved. Reserved
19 EXT_WRT	Extended write-to-read turnaround. Specifies how many extra cycles will be added between a write to read turnaround. If 0 clocks is chosen, then the DDR controller will use a fixed number based on the, read latency, and write latency. Choosing a value other than 0 adds extra cycles past this default calculation. As a default, the DDR controller will determine the write-to-read turnaround as WL - CL + BL/2 + 1. In this equation, CL is the CAS latency rounded down to the next integer, WL is the programmed write latency,

Table continues on the next page...

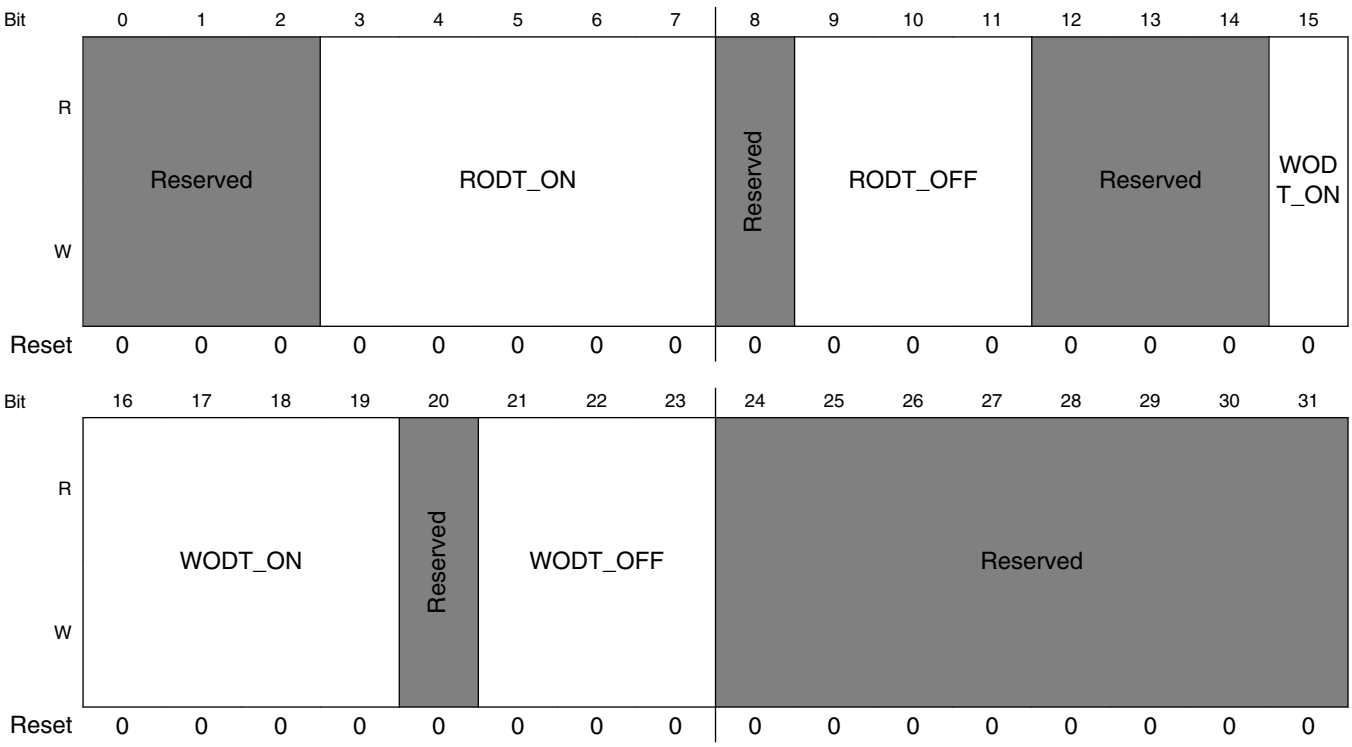
DDR_TIMING_CFG_4 field descriptions (continued)

Field	Description
	and BL is the burst length. This field is concatenated with TIMING_CFG_0[WRT] to obtain a 3-bit value for the total write-to-read turnaround 0 0 clocks 1 4 clocks
20 -	This field is reserved. Reserved
21 EXT_RRT	Extended read-to-read turnaround. Specifies how many extra cycles will be added between reads to different chip selects. As a default, 3 cycles will be required between read commands to different chip selects. Extra cycles may be added with this field. Note: If 8-beat bursts are enabled, then 5 cycles will be the default. This field is concatenated with TIMING_CFG_0[RRT] to obtain a 3-bit value for the total read-to-read turnaround 0 0 clocks 1 4 clocks
22 -	This field is reserved. Reserved
23 EXT_WWT	Extended write-to-write turnaround. Specifies how many extra cycles will be added between writes to different chip selects. As a default, 2 cycles will be required between write commands to different chip selects. Extra cycles may be added with this field. Note: If 8-beat bursts are enabled, then 4 cycles will be the default. This field is concatenated with TIMING_CFG_0[WWT] to obtain a 3-bit value for the total write-to-write turnaround 0 0 clocks 1 4 clocks
24–26 -	This field is reserved. Reserved
27 EXT_REFINT	Refresh interval. Represents the number of memory bus clock cycles between refresh cycles. Depending on DDR_SDRAM_CFG_2[NUM_PR], some number of rows are refreshed in each DDR SDRAM physical bank during each refresh cycle. The value for REFINT depends on the specific SDRAMs used and the interface clock frequency. Refreshes will not be issued when the REFINT is set to all 0s. This field is concatenated with DDR_SDRAM_INTERVAL[REFINT] to obtain a 17-bit value for the total refresh interval. 0 0 clocks 1 65,536 clocks
28–29 -	This field is reserved. Reserved
30–31 DLL_LOCK	DDR SDRAM DLL Lock Time. This provides the number of cycles that it will take for the DRAMs DLL to lock at POR and after exiting self refresh. The controller will wait the specified number of cycles before issuing any commands after exiting POR or self refresh. 00 200 clocks 01 512 clocks 10 Reserved 11 Reserved

12.4.19 DDR SDRAM timing configuration 5 (DDR_TIMING_CFG_5)

The DDR SDRAM timing configuration 5 register provides additional timing fields required to support DDR3 memories.

Address: 8000h base + 164h offset = 8164h



DDR_TIMING_CFG_5 field descriptions

Field	Description
0–2 -	This field is reserved. Reserved
3–7 RODT_ON	Read to ODT on. Specifies the number of cycles that will pass from when a read command is placed on the DRAM bus until the assertion of the relevant ODT signal(s). The default case (00000) provides a decode of [CASLAT - WR_LAT] to provide the expected default for DDR3 memories. If 2T timing is used, an extra cycle will automatically be added to the value selected in this field. 00000 CASLAT - WR_LAT 00001 0 clocks 00010 1 clocks 00011 2 clocks 01100 11 clocks 01101 - 11111 Reserved
8 -	This field is reserved. Reserved

Table continues on the next page...

DDR_TIMING_CFG_5 field descriptions (continued)

Field	Description
9–11 RODT_OFF	Read to ODT off. Specifies the number of cycles that the relevant ODT signal(s) will remain asserted for each read transaction. The default case (000) will leave the ODT signal(s) asserted for 4 DRAM cycles. 000 4 clocks 001 1 clock 010 2 clocks 010 3 clocks 100 4 clocks 101 5 clocks 110 6 clocks 111 7 clocks
12–14 -	This field is reserved. Reserved
15–19 WODT_ON	Write to ODT On Specifies the number of cycles that will pass from when a write command is placed on the DRAM bus until the assertion of the relevant ODT signal(s). The default case (00000) provides a decode of 0 cycles to provide the expected default for DDR3 memories. If 2T timing is used, an extra cycle will automatically be added to the value selected in this field. 00000 0 clocks 00001 0 clocks 00010 1 clocks 00011 2 clocks 00100 3 clocks 00101 4 clocks 00110 5 clocks 00111-11111 Reserved
20 -	This field is reserved. Reserved
21–23 WODT_OFF	Write to ODT off for DRAM. This field specifies the number of cycles that the relevant ODT signal(s) remain asserted for each write transaction. Select 4 clock cycles. The memory controller automatically adds 2 clock cycles when an 8-beat burst is used. Therefore, regardless of burst type selection, fixed 8-beat, fixed 4-beat burst chop, or on-the-fly burst chop mode value of 4 clock cycles should be selected for this field. 3'b100 for any burst type selected 000 4 clocks 001 1 clock 010 2 clocks 010 3 clocks 100 4 clocks 101 5 clocks 110 6 clocks 111 7 clocks
24–31 -	This field is reserved. Reserved

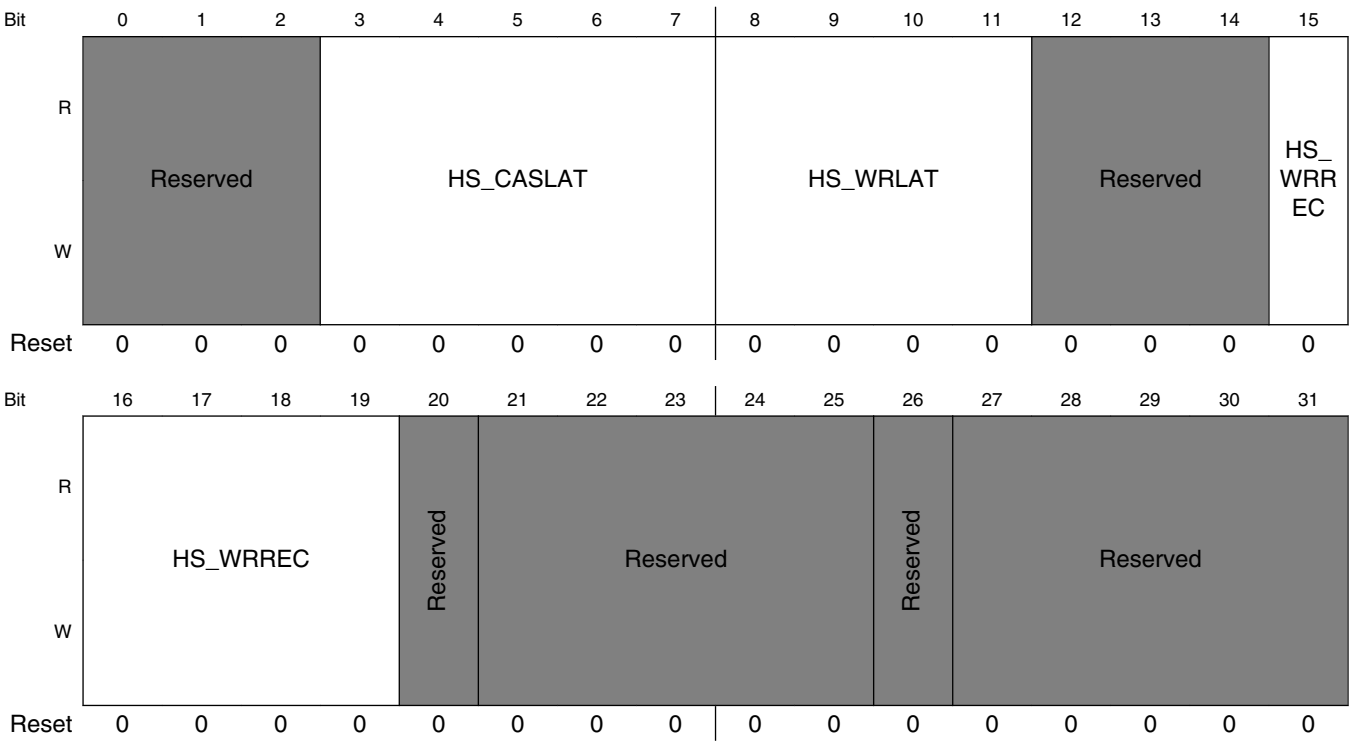
12.4.20 DDR SDRAM timing configuration 6 (DDR_TIMING_CFG_6)

The DDR SDRAM timing configuration 6 register provides additional timing fields.

If setting TIMING_CFG_6[HS_CASLAT] or TIMING_CFG_6[HS_WRLAT] to a non-zero value, then the additive latency must be programmed to 0 clocks.

If setting TIMING_CFG_6[HS_WRLAT] to a non-zero value, then TIMING_CFG_5[WODT_ON] should be programmed to 0 clocks.

Address: 8000h base + 168h offset = 8168h



DDR_TIMING_CFG_6 field descriptions

Field	Description
0–2 -	This field is reserved.
3–7 HS_CASLAT	MCAS_B latency from READ command while DDR controller is operating at half frequency. Number of clock cycles between registration of a READ command by the SDRAM and the availability of the first output data. If a READ command is registered at clock edge *n* and the latency is *m* clocks, data is available nominally coincident with clock edge *n* + *m*. If a non-zero field of this register is used, then the additive latency (TIMING_CFG_2[ADD_LAT] must be programmed to 0 clocks). Patterns not shown are reserved. 00000 Use full speed value 00101 5 clocks 00110 6 clocks

Table continues on the next page...

DDR_TIMING_CFG_6 field descriptions (continued)

Field	Description
	00111 7 clocks 01000 8 clocks 01001 9 clocks 01010 10 clocks 01011 11 clocks 01100 12 clocks 01101 13 clocks 01110 14 clocks 01111 15 clocks 10000 16 clocks 10001 17 clocks 10010 18 clocks 10011 19 clocks 10100 20 clocks
8–11 HS_WRLAT	Write latency while DDR controller is operating at half frequency. Note that the total write latency for DDR3 is equal to WR_LAT + ADD_LAT. Note that the total write latency must be at least 6 cycles if using unbuffered DIMMs in 1T timing modeIf using a non-zero value for this field, then TIMING_CFG_5[WODT_ON] must be set to 0 clocks to ensure ODT is driven correctly for writes. Patterns not shown are reserved. 0000 Use full speed value 0001-0100 Reserved 0101 5 clocks 0110 6 clocks ... 1111 15 clocks
12–14 -	This field is reserved.
15–19 HS_WRREC	Last data to precharge minimum interval (t_{WR}) while DDR controller is operating at half frequency. Determines the number of clock cycles from the last data associated with a write command until a precharge command is allowed. If DDR_SDRAM_CFG_2[OBC_CFG] is set, then this field needs to be programmed to (t_{WR} + 2 cycles). 00000 Use full speed value. 00001 1 clock 00010 2 clocks 00011 3 clocks 00100 4 clocks 00101 5 clocks 00110 6 clocks 00111 7 clocks 01000 8 clocks 01001 9 clocks 01010 10 clocks 01011 11 clocks 01100 12 clocks 01101 13 clocks 01110 14 clocks

Table continues on the next page...

DDR_TIMING_CFG_6 field descriptions (continued)

Field	Description
	01111 15 clocks 10000 16 clocks 10001 17 clocks 10010 18 clocks 10011 19 clocks 10100 20 clocks 10101 21 clocks 10110 22 clocks 10111 23 clocks 11000 24 clocks 11001 25 clocks 11010 26 clocks 11011 27 clocks 11100 28 clocks 11101 29 clocks 11110 30 clocks 11111 31 clocks
20 -	This field is reserved.
21–25 -	This field is reserved.
26 -	This field is reserved.
27–31 -	This field is reserved.

12.4.21 DDR ZQ calibration control (DDR_DDR_ZQ_CNTL)

The DDR ZQ Calibration Control register provides the enable and controls required for ZQ calibration when using DDR3 SDRAM devices.

There is a limitation for various DRAM timing parameters when ZQ calibration is used. The factors involved in this limitation are DDR_ZQ_CNTL[ZQOPER], DDR_ZQ_CNTL[ZQCS], TIMING_CFG_1[PRETOACT], TIMING_CFG_1[REFREC], DDR_SDRAM_INTERVAL[REFINT], and the number of chip selects enabled. If the following condition is true:

[((DDR_ZQ_CNTL[ZQOPER] + DDR_ZQ_CNTL[ZQCS])* (# enabled chip selects)) +

TIMING_CFG_1[PRETOACT] +

TIMING_CFG_1[REFREC] + $2t_{CK}$] > (DDR_SDRAM_INTERVAL[REFINT]),

then it is possible that one refresh will be skipped when the controller is exiting self refresh. If this is an issue, then posted refreshes could be used to extend the refresh interval. Another alternative is to use the DDR_SDRAM_MD_CNTL register to force an extra refresh to each chip select after exiting self refresh mode. However, DDR3 timing parameters for most devices/frequencies will not allow for a refresh to be missed.

Address: 8000h base + 170h offset = 8170h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	ZQ_ EN	Reserved			ZQINIT				Reserved				ZQOPER			
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				ZQCS				Reserved							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDR_DDR_ZQ_CNTL field descriptions

Field	Description
0 ZQ_EN	ZQ Calibration Enable. This bit determines if ZQ calibrating will be used. This bit should only be set if DDR3 memory is used (DDR_SDRAM_CFG[SDRAM_TYPE] = 3'b111). 0 ZQ Calibration will not be used. 1 ZQ Calibration will be used. A ZQCL command will be issued by the DDR controller after POR and anytime the DDR controller is exiting self refresh. A ZQCS command will be issued every 32 refresh sequences to account for VT variations.
1–3 -	This field is reserved. Reserved
4–7 ZQINIT	POR ZQ Calibration Time (t_{ZQinit}). Determines the number of cycles that must be allowed for DRAM ZQ calibration at POR. Each chip select will be calibrated separately, and this time must elapse after the ZQCL command is issued for each chip select before a separate command may be issued. 0000-0110 Reserved 0111 128 clocks 1000 256 clocks 1001 512 clocks 1010 1024 clocks 1011-1111 Reserved
8–11 -	This field is reserved. Reserved
12–15 ZQOPER	Normal Operation Full Calibration Time (t_{ZQoper}). Determines the number of cycles that must be allowed for DRAM ZQ calibration when exiting self refresh. Each chip select will be calibrated separately, and this time must elapse after the ZQCL command is issued for each chip select before a separate command may be issued. 0000-0110 Reserved 0111 128 clocks 1000 256 clocks 1001 512 clocks

Table continues on the next page...

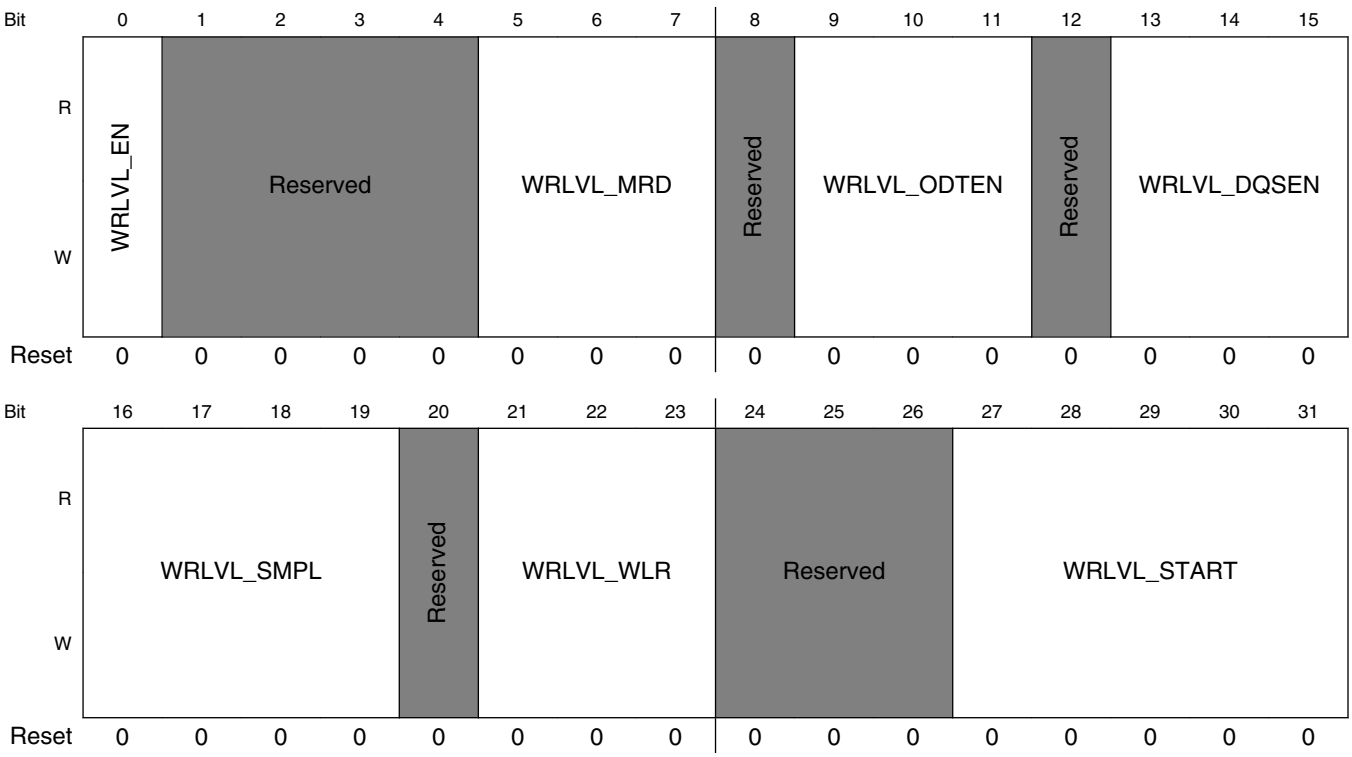
DDR_DDR_ZQ_CNTL field descriptions (continued)

Field	Description
	1010 1024 clocks 1011-1111 Reserved
16–19 -	This field is reserved. Reserved
20–23 ZQCS	Normal Operation Short Calibration Time (t_{ZQCS}). Determines the number of cycles that must be allowed for DRAM ZQ calibration during dynamic calibration which is issued every 32 refresh cycles. Each chip select will be calibrated separately, and this time must elapse after the ZQCS command is issued for each chip select before a separate command may be issued. 0000 1 clocks 0001 2 clocks 0010 4 clocks 0011 8 clocks 0100 16 clocks 0101 32 clocks 0110 64 clocks 0111 128 clocks 1000 256 clocks 1001 512 clocks 1010-1111 Reserved
24–31 -	This field is reserved. Reserved

12.4.22 DDR write leveling control (DDR_DDR_WRLVL_CNTL)

The DDR Write Leveling Control register provides controls for write leveling, as it is supported for DDR3 memory devices.

Address: 8000h base + 174h offset = 8174h



DDR_DDR_WRLVL_CNTL field descriptions

Field	Description
0 WRLVL_EN	Write Leveling Enable. This bit determines if write leveling will be used. If this bit is set, then the DDR controller will perform write leveling immediately after initializing the DRAM. 0 Write leveling will not be used 1 Write leveling will be used
1–4 -	This field is reserved. Reserved
5–7 WRLVL_MRD	First DQS pulse rising edge after margining mode is programmed (t_{WL_MRD}). Determines how many cycles to wait after margining mode has been programmed before the first DQS pulse may be issued. This field is only relevant when DDR_WRLVL_CNTL[WRLVL_EN] is set. 000 1 clocks 001 2 clocks 010 4 clocks 011 8 clocks

Table continues on the next page...

DDR_DDR_WRLVL_CNTL field descriptions (continued)

Field	Description
	100 16 clocks 101 32 clocks 110 64 clocks 111 128 clocks
8 -	This field is reserved. Reserved
9–11 WRLVL_ODTEN	ODT delay after margining mode is programmed (t_{WL_ODTEN}). Determines how many cycles to wait after margining mode has been programmed.until ODT may be asserted.This field is only relevant when DDR_WRLVL_CNTL[WRLVL_EN] is set. 000 1 clocks 001 2 clocks 010 4 clocks 011 8 clocks 100 16 clocks 101 32 clocks 110 64 clocks 111 128 clocks
12 -	This field is reserved. Reserved
13–15 WRLVL_DQSEN	DQS/DQS_B delay after margining mode is programmed (t_{WL_DQSEN}). Determines how many cycles to wait after margining mode has been programmed.until DQS may be actively driven. This field is only relevant when DDR_WRLVL_CNTL[WRLVL_EN] is set. 000 1 clocks 001 2 clocks 010 4 clocks 011 8 clocks 100 16 clocks 101 32 clocks 110 64 clocks 111 128 clocks
16–19 WRLVL_SMPL	Write leveling sample time. Determines the number of cycles that must pass before the data signals are sampled after a DQS pulse during margining mode. This field should be programmed at least 6 cycles higher than t_{WLO} to allow enough time for propagation delay and sampling of the prime data bits. This field is only relevant when DDR_WRLVL_CNTL[WRLVL_EN] is set. 0000 32 clocks 0001 1 clocks 0010 2 clocks 0011 3 clocks 0100 4 clocks 0101 5 clocks 0110 6 clocks 0111 7 clocks 1000 8 clocks 1001 9 clocks 1010 10 clocks

Table continues on the next page...

DDR_DDR_WRLVL_CNTL field descriptions (continued)

Field	Description
	1011 11 clocks 1100 12 clocks 1101 13 clocks 1110 14 clocks 1111 15 clocks
20 -	This field is reserved. Reserved
21–23 WRLVL_WLR	Write leveling repetition time. Determines the number of cycles that must pass between DQS pulses during write leveling. This field is only relevant when DDR_WRLVL_CNTL[WRLVL_EN] is set. 000 1 clocks 001 2 clocks 010 4 clocks 011 8 clocks 100 16 clocks 101 32 clocks 110 64 clocks 111 128 clocks
24–26 -	This field is reserved. Reserved
27–31 WRLVL_START	Write leveling start time for DQS[0]. Determines the value to use for the DQS_ADJUST for the first sample when write leveling is enabled. 00000-10100 Bit settings indicate the amount of delay in 1/8 clock increments. For example, 00000 means 0 clock delay; 00010 means 1/4 clock delay; 00011 means 3/8 clock delay. 10101-11111 Reserved

12.4.23 DDR Self Refresh Counter (DDR_DDR_SR_CNTR)

The DDR Self Refresh Counter register can be programmed to force the DDR controller to enter self refresh after a predefined period of idle time.

Address: 8000h base + 17Ch offset = 817Ch

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												SR_IT				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

DDR_DDR_SR_CNTR field descriptions

Field	Description
0–11 -	This field is reserved. Reserved

Table continues on the next page...

DDR_DDR_SR_CNTR field descriptions (continued)

Field	Description
12–15 SR_IT	Self Refresh Idle Threshold. Defines the number of DRAM cycles that must pass while the DDR controller is idle before it will enter self refresh. Anytime a transaction is issued to the DDR controller, it will reset its internal counter. When a new transaction is received by the DDR controller, it will exit self refresh and reset its internal counter. If this field is zero, then the described power savings feature will be disabled. In addition, if a non-zero value is programmed into this field, then the DDR controller will exit self refresh anytime a transaction is issued to the DDR controller, regardless of the reason self refresh was initially entered. 0000 Automatic self refresh entry disabled 0001 2^10 DRAM clocks 0010 2^12 DRAM clocks 0011 2^14 DRAM clocks 0100 2^16 DRAM clocks 0101 2^18 DRAM clocks 0110 2^20 DRAM clocks 0111 2^22 DRAM clocks 1000 2^24 DRAM clocks 1001 2^26 DRAM clocks 1010 2^28 DRAM clocks 1011 2^30 DRAM clocks 1100-1111 Reserved
16–31 -	This field is reserved. Reserved

12.4.24 DDR Register Control Words 1 (DDR_DDR_SDRAM_RCW_1)

The DDR Register Control Word 1 register should be programmed with the intended values of the register control words if DDR_SDRAM_CFG[RCW_EN] is set. Each 4-bit field represents the value that will be placed on MBA[1], MBA[0], MA[4], and MA[3] during register control word writes.

Address: 8000h base + 180h offset = 8180h

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	RCW0				RCW1				RCW2				RCW3				RCW4				RCW5				RCW6				RCW7			
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

DDR_DDR_SDRAM_RCW_1 field descriptions

Field	Description
0–3 RCW0	Register Control Word 0. Represents the value that will be placed on MBA[1], MBA[0], MA[4], and MA[3] during writes to register control word 0.

Table continues on the next page...

DDR_DDR_SDRAM_RCW_1 field descriptions (continued)

Field	Description
4–7 RCW1	Register Control Word 1. Represents the value that will be placed on MBA[1], MBA[0], MA[4], and MA[3] during writes to register control word 1.
8–11 RCW2	Register Control Word 2. Represents the value that will be placed on MBA[1], MBA[0], MA[4], and MA[3] during writes to register control word 2.
12–15 RCW3	Register Control Word 3. Represents the value that will be placed on MBA[1], MBA[0], MA[4], and MA[3] during writes to register control word 3.
16–19 RCW4	Register Control Word 4. Represents the value that will be placed on MBA[1], MBA[0], MA[4], and MA[3] during writes to register control word 4.
20–23 RCW5	Register Control Word 5. Represents the value that will be placed on MBA[1], MBA[0], MA[4], and MA[3] during writes to register control word 5.
24–27 RCW6	Register Control Word 6. Represents the value that will be placed on MBA[1], MBA[0], MA[4], and MA[3] during writes to register control word 6.
28–31 RCW7	Register Control Word 7. Represents the value that will be placed on MBA[1], MBA[0], MA[4], and MA[3] during writes to register control word 7.

12.4.25 DDR Register Control Words 2 (DDR_DDR_SDRAM_RCW_2)

The DDR Register Control Word 2 register should be programmed with the intended values of the register control words if DDR_SDRAM_CFG[RCW_EN] is set. Each 4-bit field represents the value that will be placed on MBA[1], MBA[0], MA[4], and MA[3] during register control word writes.

Address: 8000h base + 184h offset = 8184h

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	RCW8				RCW9				RCW10				RCW11				RCW12				RCW13				RCW14				RCW15			
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

DDR_DDR_SDRAM_RCW_2 field descriptions

Field	Description
0–3 RCW8	Register Control Word 8. Represents the value that will be placed on MBA[1], MBA[0], MA[4], and MA[3] during writes to register control word 8.

Table continues on the next page...

DDR_DDR_SDRAM_RCW_2 field descriptions (continued)

Field	Description
4–7 RCW9	Register Control Word 9. Represents the value that will be placed on MBA[1], MBA[0], MA[4], and MA[3] during writes to register control word 9.
8–11 RCW10	Register Control Word 10. Represents the value that will be placed on MBA[1], MBA[0], MA[4], and MA[3] during writes to register control word 10.
12–15 RCW11	Register Control Word 11. Represents the value that will be placed on MBA[1], MBA[0], MA[4], and MA[3] during writes to register control word 11.
16–19 RCW12	Register Control Word 12. Represents the value that will be placed on MBA[1], MBA[0], MA[4], and MA[3] during writes to register control word 12.
20–23 RCW13	Register Control Word 13. Represents the value that will be placed on MBA[1], MBA[0], MA[4], and MA[3] during writes to register control word 13.
24–27 RCW14	Register Control Word 14. Represents the value that will be placed on MBA[1], MBA[0], MA[4], and MA[3] during writes to register control word 14.
28–31 RCW15	Register Control Word 15. Represents the value that will be placed on MBA[1], MBA[0], MA[4], and MA[3] during writes to register control word 15.

12.4.26 DDR write leveling control 2 (DDR_DDR_WRLVL_CNTL_2)

The DDR Write Leveling Control 2 register provides controls for write leveling, as it is supported for DDR3 memory devices. This register specifically defines the starting points for the individual data strobes.

NOTE

For each field WRLVL_START_n, a setting of 0000 is not recommended; it disables the per-byte write level start time selection. It is recommended to select proper individual delay for each byte lane based on board skews.

Address: 8000h base + 190h offset = 8190h

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			WRLVL_START_1					Reserved			WRLVL_START_2					Reserved			WRLVL_START_3					Reserved			WRLVL_START_4				
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

DDR_DDR_WRLVL_CNTL_2 field descriptions

Field	Description
0–2 -	This field is reserved. Reserved
3–7 WRLVL_START_1	Write leveling start time for DQS[1]. Determines the value to use for the DQS_ADJUST for the first sample when write leveling is enabled. 00000 Use value from DDR_WRLVL_CNTL[WRLVL_START]

Table continues on the next page...

DDR_DDR_WRLVL_CNTL_2 field descriptions (continued)

Field	Description
	00001-10100 Bit settings indicate the amount of delay in 1/8 clock increments. For example, 00000 means 0 clock delay; 00010 means 1/4 clock delay; 00011 means 3/8 clock delay. 10101-11111 Reserved
8–10 - Reserved	This field is reserved.
11–15 WRLVL_START_2	Write leveling start time for DQS[2]. Determines the value to use for the DQS_ADJUST for the first sample when write leveling is enabled. 00000 Use value from DDR_WRLVL_CNTL[WRLVL_START] 00001-10100 Bit settings indicate the amount of delay in 1/8 clock increments. For example, 00000 means 0 clock delay; 00010 means 1/4 clock delay; 00011 means 3/8 clock delay. 10101-11111 Reserved
16–18 - Reserved	This field is reserved.
19–23 WRLVL_START_3	Write leveling start time for DQS[3]. Determines the value to use for the DQS_ADJUST for the first sample when write leveling is enabled. 00000 Use value from DDR_WRLVL_CNTL[WRLVL_START] 00001-10100 Bit settings indicate the amount of delay in 1/8 clock increments. For example, 00000 means 0 clock delay; 00010 means 1/4 clock delay; 00011 means 3/8 clock delay. 10101-11111 Reserved
24–26 - Reserved	This field is reserved.
27–31 WRLVL_START_4	Write leveling start time for DQS[4]. Determines the value to use for the DQS_ADJUST for the first sample when write leveling is enabled. 00000 Use value from DDR_WRLVL_CNTL[WRLVL_START] 00001-10100 Bit settings indicate the amount of delay in 1/8 clock increments. For example, 00000 means 0 clock delay; 00010 means 1/4 clock delay; 00011 means 3/8 clock delay. 10101-11111 Reserved

12.4.27 DDR write leveling control 3 (DDR_DDR_WRLVL_CNTL_3)

The DDR Write Leveling Control 3 register provides controls for write leveling, as it is supported for DDR3 memory devices. This register specifically defines the starting points for the individual data strobes.

NOTE

For each field WRLVL_START_n, a setting of 0000 is not recommended; it disables the per-byte write level start time selection. It is recommended to select proper individual delay for each byte lane based on board skews.

Address: 8000h base + 194h offset = 8194h

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			WRLVL_START_5					Reserved			WRLVL_START_6					Reserved			WRLVL_START_7					Reserved			WRLVL_START_8				
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

DDR_DDR_WRLVL_CNTL_3 field descriptions

Field	Description
0–2 -	This field is reserved. Reserved
3–7 WRLVL_START_5	Write leveling start time for DQS[5]. Determines the value to use for the DQS_ADJUST for the first sample when write leveling is enabled. 00000 Use value from DDR_WRLVL_CNTL[WRLVL_START] 00001-10100 Bit settings indicate the amount of delay in 1/8 clock increments. For example, 00000 means 0 clock delay; 00010 means 1/4 clock delay; 00011 means 3/8 clock delay. 10101-11111 Reserved
8–10 -	This field is reserved. Reserved
11–15 WRLVL_START_6	Write leveling start time for DQS[6]. Determines the value to use for the DQS_ADJUST for the first sample when write leveling is enabled. 00000 Use value from DDR_WRLVL_CNTL[WRLVL_START] 00001-10100 Bit settings indicate the amount of delay in 1/8 clock increments. For example, 00000 means 0 clock delay; 00010 means 1/4 clock delay; 00011 means 3/8 clock delay. 10101-11111 Reserved
16–18 -	This field is reserved. Reserved
19–23 WRLVL_START_7	Write leveling start time for DQS[7]. Determines the value to use for the DQS_ADJUST for the first sample when write leveling is enabled. 00000 Use value from DDR_WRLVL_CNTL[WRLVL_START] 00001-10100 Bit settings indicate the amount of delay in 1/8 clock increments. For example, 00000 means 0 clock delay; 00010 means 1/4 clock delay; 00011 means 3/8 clock delay. 10101-11111 Reserved
24–26 -	This field is reserved. Reserved
27–31 WRLVL_START_8	Write leveling start time for DQS[8]. Determines the value to use for the DQS_ADJUST for the first sample when write leveling is enabled. 00000 Use value from DDR_WRLVL_CNTL[WRLVL_START] 00001-10100 Bit settings indicate the amount of delay in 1/8 clock increments. For example, 00000 means 0 clock delay; 00010 means 1/4 clock delay; 00011 means 3/8 clock delay. 10101-11111 Reserved

12.4.28 DDR SDRAM mode configuration 3 (DDR_DDR_SDRAM_MODE_3)

The DDR SDRAM mode configuration register 3 sets the values loaded into the DDR's mode registers. This register is used specifically for chip select 1 if DDR_SDRAM_CFG_2[UNQ_MRS_EN] is set.

Address: 8000h base + 200h offset = 8200h

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	ESDMODE																SDMODE															
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

DDR_DDR_SDRAM_MODE_3 field descriptions

Field	Description
0–15 ESDMODE	Extended SDRAM mode. Specifies the initial value loaded into the DDR SDRAM extended mode register for chip select 1 if DDR_SDRAM_CFG_2[UNQ_MRS_EN] is set. The range and meaning of legal values is specified by the DDR SDRAM manufacturer. When this value is driven onto the address bus (during the DDR SDRAM initialization sequence), MA[0] presents the lsb of ESDMODE, which, in the big-endian convention shown in the register figure, corresponds to ESDMODE[15]. The msb of the SDRAM extended mode register value must be stored at ESDMODE[0]. The value programmed into this field is also used for writing MR1 during write leveling for DDR3, although the bits specifically related to the write leveling scheme are handled automatically by the DDR controller. Even if DDR_SDRAM_CFG[BI] is set, this field is still used during write leveling. The write leveling enable bit should be cleared by software in this field.
16–31 SDMODE	SDRAM mode. Specifies the initial value loaded into the DDR SDRAM mode register for chip select 1 if DDR_SDRAM_CFG_2[UNQ_MRS_EN] is set. The range of legal values is specified by the DDR SDRAM manufacturer. When this value is driven onto the address bus (during DDR SDRAM initialization), MA[0] presents the lsb of SDMODE, which, in the big-endian convention shown in the register figure, corresponds to SDMODE[15]. The msb of the SDRAM mode register value must be stored at SDMODE[0]. Because the memory controller forces SDMODE[7] to certain values depending on the state of the initialization sequence, (for resetting the SDRAM's DLL) the corresponding bits of this field are ignored by the memory controller. Note that SDMODE[7] is mapped to MA[8].

12.4.29 DDR SDRAM mode configuration 4 (DDR_DDR_SDRAM_MODE_4)

The DDR SDRAM mode configuration register 4 sets the values loaded into the DDR's extended mode 2 and 3 registers. This register is used specifically for chip select 1 if DDR_SDRAM_CFG_2[UNQ_MRS_EN] is set.

Address: 8000h base + 204h offset = 8204h

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	ESDMODE2																ESDMODE3															
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

DDR_DDR_SDRAM_MODE_4 field descriptions

Field	Description
0–15 ESDMODE2	Extended SDRAM mode 2. Specifies the initial value loaded into the DDR SDRAM extended 2 mode register for chip select 1 if DDR_SDRAM_CFG_2[UNQ_MRS_EN] is set. The range and meaning of legal values is specified by the DDR SDRAM manufacturer. When this value is driven onto the address bus (during the DDR SDRAM initialization sequence), MA[0] presents the lsb bit of ESDMODE2, which, in the big-endian convention shown in the register figure, corresponds to ESDMODE2[15]. The msb of the SDRAM extended mode 2 register value must be stored at ESDMODE2[0].
16–31 ESDMODE3	Extended SDRAM mode 3. Specifies the initial value loaded into the DDR SDRAM extended 3 mode register for chip select 1 if DDR_SDRAM_CFG_2[UNQ_MRS_EN] is set. The range of legal values of legal values is specified by the DDR SDRAM manufacturer. When this value is driven onto the address bus (during DDR SDRAM initialization), MA[0] presents the lsb of ESDMODE3, which, in the big-endian convention shown in the register figure, corresponds to ESDMODE3[15]. The msb of the SDRAM extended mode 3 register value must be stored at ESDMODE3[0].

12.4.30 DDR SDRAM mode configuration 5 (DDR_DDR_SDRAM_MODE_5)

The DDR SDRAM mode configuration register 5 sets the values loaded into the DDR's mode registers. This register is used specifically for chip select 2 if DDR_SDRAM_CFG_2[UNQ_MRS_EN] is set.

Address: 8000h base + 208h offset = 8208h

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	ESDMODE																SDMODE															
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

DDR_DDR_SDRAM_MODE_5 field descriptions

Field	Description
0–15 ESDMODE	Extended SDRAM mode. Specifies the initial value loaded into the DDR SDRAM extended mode register for chip select 2 if DDR_SDRAM_CFG_2[UNQ_MRS_EN] is set. The range and meaning of legal values is specified by the DDR SDRAM manufacturer. When this value is driven onto the address bus (during the DDR SDRAM initialization sequence), MA[0] presents the lsb of ESDMODE, which, in the big-endian convention shown in the register figure, corresponds to ESDMODE[15]. The msb of the SDRAM extended mode register value must be stored at ESDMODE[0]. The value programmed into this field is also used for writing MR1 during write leveling for DDR3, although the bits specifically related to the write leveling scheme are handled automatically by the DDR controller. Even if DDR_SDRAM_CFG[BI] is set, this field is still used during write leveling. The write leveling enable bit should be cleared by software in this field.
16–31 SDMODE	SDRAM mode. Specifies the initial value loaded into the DDR SDRAM mode register for chip select 2 if DDR_SDRAM_CFG_2[UNQ_MRS_EN] is set. The range of legal values is specified by the DDR SDRAM manufacturer. When this value is driven onto the address bus (during DDR SDRAM initialization), MA[0] presents the lsb of SDMODE, which, in the big-endian convention shown in the register figure, corresponds to SDMODE[15]. The msb of the SDRAM mode register value must be stored at SDMODE[0]. Because the memory controller forces SDMODE[7] to certain values depending on the state of the initialization sequence, (for resetting the SDRAM's DLL) the corresponding bits of this field are ignored by the memory controller. Note that SDMODE[7] is mapped to MA[8].

12.4.31 DDR SDRAM mode configuration 6 (DDR_DDR_SDRAM_MODE_6)

The DDR SDRAM mode configuration register 6 sets the values loaded into the DDR's extended mode 2 and 3 registers. This register is used specifically for chip select 2 if DDR_SDRAM_CFG_2[UNQ_MRS_EN] is set.

Address: 8000h base + 20Ch offset = 820Ch

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	ESDMODE2																ESDMODE3															
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

DDR_DDR_SDRAM_MODE_6 field descriptions

Field	Description
0–15 ESDMODE2	Extended SDRAM mode 2. Specifies the initial value loaded into the DDR SDRAM extended 2 mode register for chip select 2 if DDR_SDRAM_CFG_2[UNQ_MRS_EN] is set. The range and meaning of legal values is specified by the DDR SDRAM manufacturer. When this value is driven onto the address bus (during the DDR SDRAM initialization sequence), MA[0] presents the lsb bit of ESDMODE2, which, in the big-endian convention shown in the register figure, corresponds to ESDMODE2[15]. The msb of the SDRAM extended mode 2 register value must be stored at ESDMODE2[0].

Table continues on the next page...

DDR_DDR_SDRAM_MODE_6 field descriptions (continued)

Field	Description
16–31 ESDMODE3	Extended SDRAM mode 3. Specifies the initial value loaded into the DDR SDRAM extended 3 mode register for chip select 2 if DDR_SDRAM_CFG_2[UNQ_MRS_EN] is set. The range of legal values of legal values is specified by the DDR SDRAM manufacturer. When this value is driven onto the address bus (during DDR SDRAM initialization), MA[0] presents the lsb of ESDMODE3, which, in the big-endian convention shown in the register figure, corresponds to ESDMODE3[15]. The msb of the SDRAM extended mode 3 register value must be stored at ESDMODE3[0].

12.4.32 DDR SDRAM mode configuration 7 (DDR_DDR_SDRAM_MODE_7)

The DDR SDRAM mode configuration register 7 sets the values loaded into the DDR's mode registers. This register is used specifically for chip select 3 if DDR_SDRAM_CFG_2[UNQ_MRS_EN] is set.

Address: 8000h base + 210h offset = 8210h

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	ESDMODE																SDMODE															
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

DDR_DDR_SDRAM_MODE_7 field descriptions

Field	Description
0–15 ESDMODE	Extended SDRAM mode. Specifies the initial value loaded into the DDR SDRAM extended mode register for chip select 3 if DDR_SDRAM_CFG_2[UNQ_MRS_EN] is set. The range and meaning of legal values is specified by the DDR SDRAM manufacturer. When this value is driven onto the address bus (during the DDR SDRAM initialization sequence), MA[0] presents the lsb of ESDMODE, which, in the big-endian convention shown in the register figure, corresponds to ESDMODE[15]. The msb of the SDRAM extended mode register value must be stored at ESDMODE[0]. The value programmed into this field is also used for writing MR1 during write leveling for DDR3, although the bits specifically related to the write leveling scheme are handled automatically by the DDR controller. Even if DDR_SDRAM_CFG[BI] is set, this field is still used during write leveling. The write leveling enable bit should be cleared by software in this field.
16–31 SDMODE	SDRAM mode. Specifies the initial value loaded into the DDR SDRAM mode register for chip select 3 if DDR_SDRAM_CFG_2[UNQ_MRS_EN] is set. The range of legal values is specified by the DDR SDRAM manufacturer. When this value is driven onto the address bus (during DDR SDRAM initialization), MA[0] presents the lsb of SDMODE, which, in the big-endian convention shown in the register figure, corresponds to SDMODE[15]. The msb of the SDRAM mode register value must be stored at SDMODE[0]. Because the memory controller forces SDMODE[7] to certain values depending on the state of the initialization sequence, (for resetting the SDRAM's DLL) the corresponding bits of this field are ignored by the memory controller. Note that SDMODE[7] is mapped to MA[8].

12.4.33 DDR SDRAM mode configuration 8 (DDR_DDR_SDRAM_MODE_8)

The DDR SDRAM mode configuration register 8 sets the values loaded into the DDR's extended mode 2 and 3 registers. This register is used specifically for chip select 3 if DDR_SDRAM_CFG_2[UNQ_MRS_EN] is set.

Address: 8000h base + 214h offset = 8214h

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	ESDMODE2																ESDMODE3															
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

DDR_DDR_SDRAM_MODE_8 field descriptions

Field	Description
0–15 ESDMODE2	Extended SDRAM mode 2. Specifies the initial value loaded into the DDR SDRAM extended 2 mode register for chip select 3 if DDR_SDRAM_CFG_2[UNQ_MRS_EN] is set. The range and meaning of legal values is specified by the DDR SDRAM manufacturer. When this value is driven onto the address bus (during the DDR SDRAM initialization sequence), MA[0] presents the lsb bit of ESDMODE2, which, in the big-endian convention shown in the register figure, corresponds to ESDMODE2[15]. The msb of the SDRAM extended mode 2 register value must be stored at ESDMODE2[0].
16–31 ESDMODE3	Extended SDRAM mode 3. Specifies the initial value loaded into the DDR SDRAM extended 3 mode register for chip select 3 if DDR_SDRAM_CFG_2[UNQ_MRS_EN] is set. The range of legal values of legal values is specified by the DDR SDRAM manufacturer. When this value is driven onto the address bus (during DDR SDRAM initialization), MA[0] presents the lsb of ESDMODE3, which, in the big-endian convention shown in the register figure, corresponds to ESDMODE3[15]. The msb of the SDRAM extended mode 3 register value must be stored at ESDMODE3[0].

12.4.34 DDR Debug Status Register 1 (DDR_DDRDSR_1)

The DDRDSR_1 register contains the current settings of the driver impedance for the command/control and data.

Address: 8000h base + B20h offset = 8B20h

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	CZ	Reserved							DZ	Reserved						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

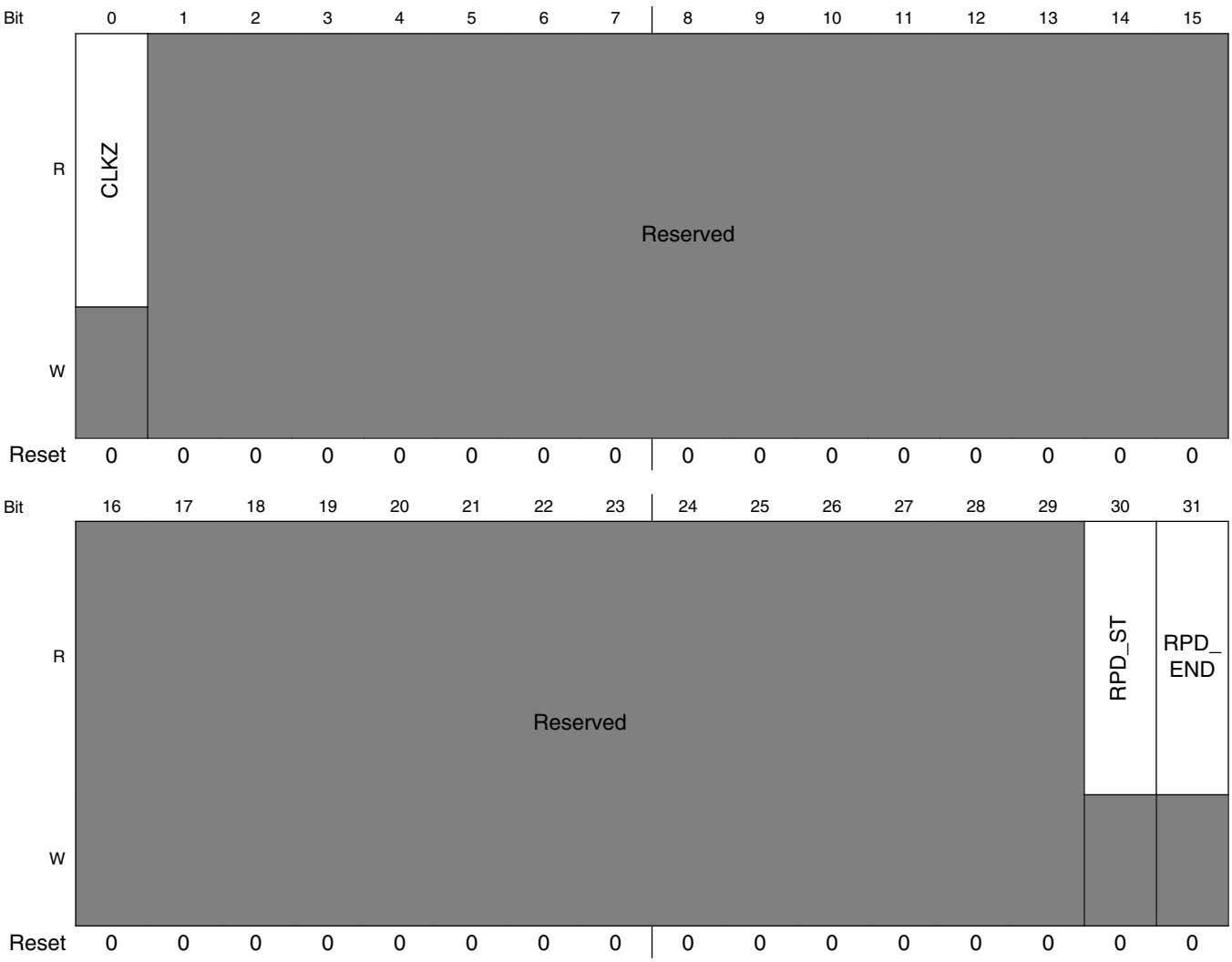
DDR_DDRDSR_1 field descriptions

Field	Description
0–15 -	This field is reserved. Reserved
16 CZ	Current setting of driver command impedance
17–23 -	This field is reserved. Reserved
24 DZ	Current setting of driver data impedance
25–31 -	This field is reserved. Reserved

12.4.35 DDR Debug Status Register 2 (DDR_DDRDSR_2)

The DDRDSR_2 register contains the current settings of the driver impedance for the DDR drivers for clocks.

Address: 8000h base + B24h offset = 8B24h



DDR_DDRDSR_2 field descriptions

Field	Description
0 CLKZ	Current setting of driver clock impedance
1–29 -	This field is reserved. Reserved

Table continues on the next page...

DDR_DDRDSR_2 field descriptions (continued)

Field	Description
30 RPD_ST	Rapid clear of memory start. See DDR Rapid Clear of Memory for more information. 0 The rapid clear of memory function has not been started. 1 The rapid clear of memory function has started. During the rapid clear of memory, writes to the DDR memory mapped registers will not affect the register contents. This bit is cleared by software writing a 1.
31 RPD_END	Rapid clear of memory end. See DDR Rapid Clear of Memory," for more information. 0 The rapid clear of memory function has not ended. 1 The rapid clear of memory function has completed. This bit is cleared by software writing a 1.

12.4.36 DDR Control Driver Register 1 (DDR_DDRCDR_1)

DDRCDR_1 sets the hardware DDR driver calibration enable, the ODT termination value for IOs, and the software override enables for address/command and data bus signals. The combined DDRCDR_1[ODT] and DDRCDR_2[ODT] set the on-die-termination values in the memory controller for the data bus signals. Hardware DDR driver calibration must be enabled by setting DDRCDR_1[DHC_EN]. MDIC[0:1] pins are required to be connected to proper calibration resistors. The proper value and connection of the MDIC resistor are specified in the corresponding device hardware specification document.

The global half-strength override field DDR_SDRAM_CFG[HSE] or software overrides in DDRCDR_1 or DDRCDR_2 registers determine whether the DDR drivers calibrate to full-strength or half-strength. The software overrides in DDRCDR_1 or DDRCDR_2 registers take precedence over the DDR_SDRAM_CFG[HSE] setting.

NOTE

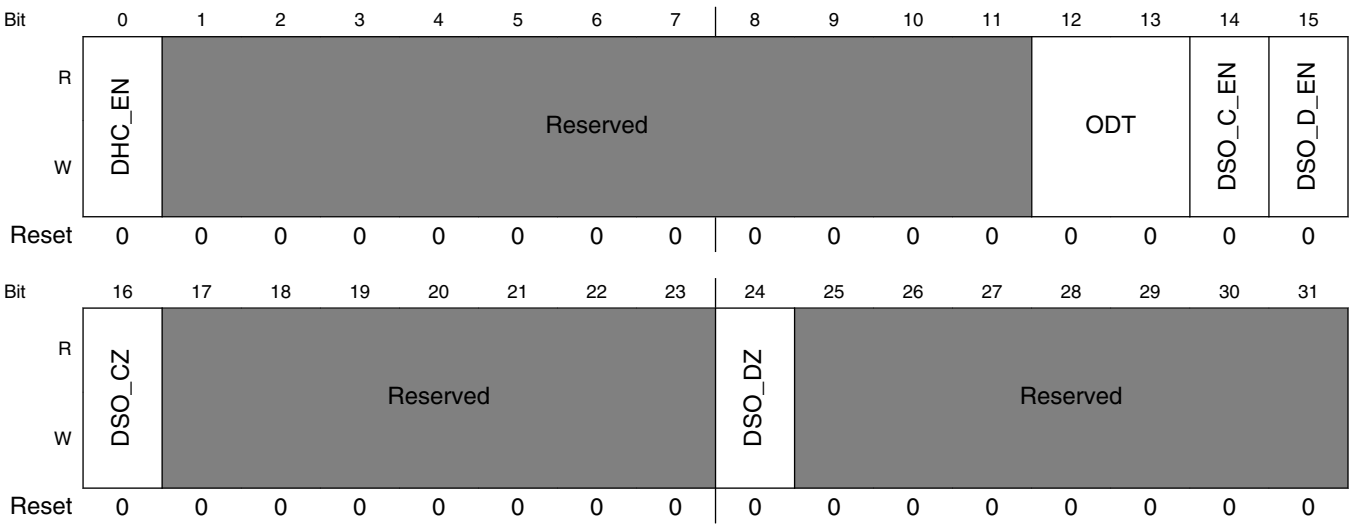
All driver calibration related registers should be set 500 µs before the DDR controller is enabled (that is, before DDR_SDRAM_CFG[MEM_EN] is set).

The table below lists the valid impedance override values. Note that the drivers may be calibrated to either full-strength or half-strength.

Table 12-54. Valid Impedance Override Values

Driver impedance	Impedance Override Value
Highest	0
Lowest	1

Address: 8000h base + B28h offset = 8B28h



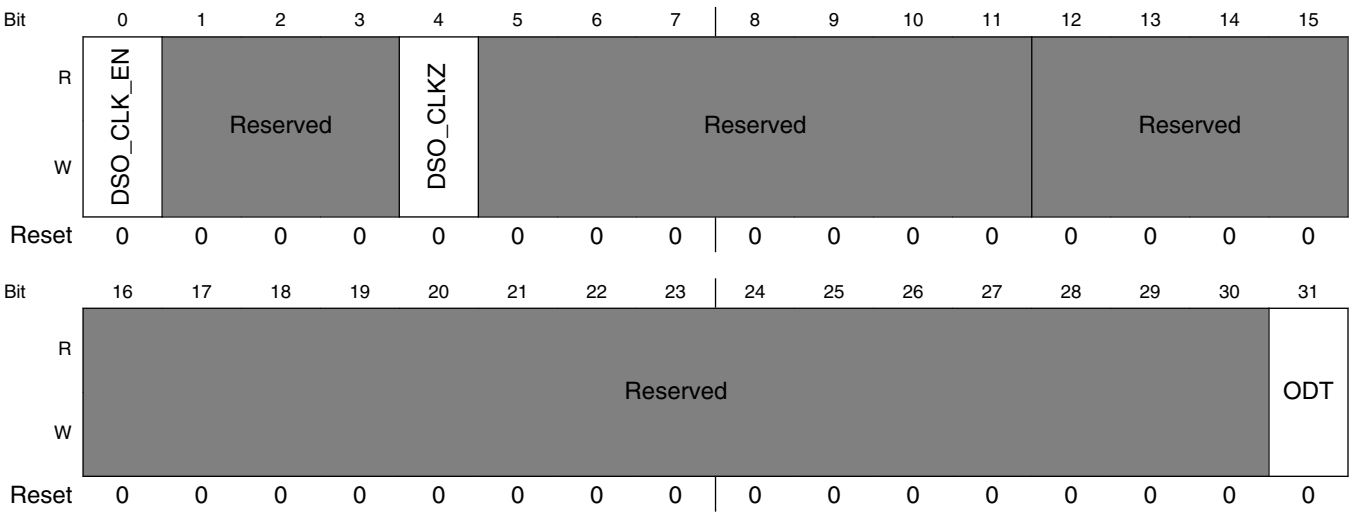
DDR_DDRCDR_1 field descriptions

Field	Description
0 DHC_EN	DDR driver hardware compensation enable
1–11 -	This field is reserved. Reserved. These bits are writeable, but they are unused.
12–13 ODT	ODT termination value for IOs. This is combined with DDRCDR_2[ODT] to determine the termination value. Below is the termination based on concatenating these two fields. Note that the order of concatenation is (from left to right) DDRCDR_1[ODT], DDRCDR_2[ODT] **NOTE:** All other settings reserved 000 Termsel off 001 120 Ω 011 75 Ω 101 60 Ω 111 46 Ω
14 DSO_C_EN	Driver software override enable for address/command
15 DSO_D_EN	Driver software override enable for data
16 DSO_CZ	DDR driver software command impedance override
17–23 -	This field is reserved. Reserved. These bits are writeable, but they are unused.
24 DSO_DZ	Driver software data impedance override
25–31 -	This field is reserved. Reserved. These bits are writeable, but they are unused.

12.4.37 DDR Control Driver Register 2 (DDR_DDRCDR_2)

The DDRCDR_2 sets the driver software override enable for clocks, and the DDR clocks driver P/N impedance.

Address: 8000h base + B2Ch offset = 8B2Ch



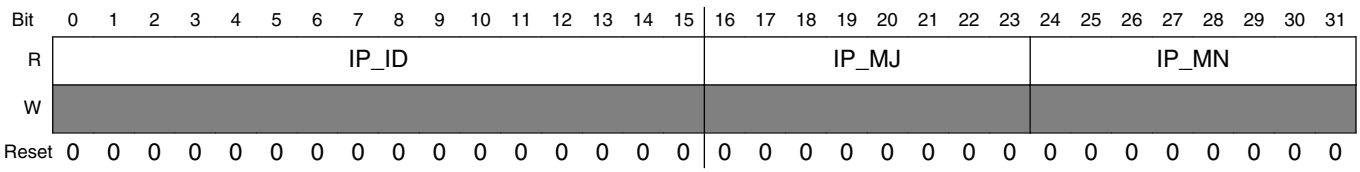
DDR_DDRCDR_2 field descriptions

Field	Description
0 DSO_CLK_EN	Driver software override enable for clocks
1–3 -	This field is reserved. Reserved
4 DSO_CLKZ	Driver software clocks impedance override
5–11 -	This field is reserved. Reserved. These bits are writeable, but they are unused.
12–30 -	This field is reserved. Reserved
31 ODT	ODT termination value for IOs. This is combined with DDRCDR_2[ODT] to determine the termination value. Below is the termination based on concatenating these two fields. Note that the order of concatenation is (from left to right) DDRCDR_1[ODT], DDRCDR_2[ODT] 000 Termsel off 001 120 Ω 011 75 Ω 101 60 Ω 111 47 Ω

12.4.38 DDR IP block revision 1 (DDR_DDR_IP_REV1)

The DDR IP block revision 1 register provides read-only fields with the IP block ID, along with major and minor revision information.

Address: 8000h base + BF8h offset = 8BF8h



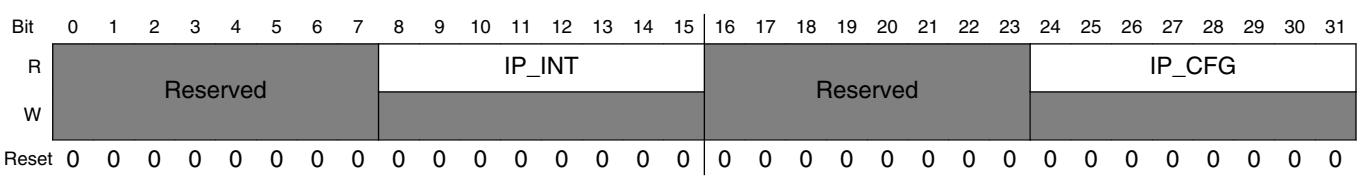
DDR_DDR_IP_REV1 field descriptions

Field	Description
0–15 IP_ID	IP block ID. For the DDR controller, this value is 0x0002.
16–23 IP_MJ	Major revision. This is currently set to 0x04.
24–31 IP_MN	Minor revision. This is currently set to 0x07.

12.4.39 DDR IP block revision 2 (DDR_DDR_IP_REV2)

The DDR IP block revision 2 register provides read-only fields with the IP block integration and configuration options.

Address: 8000h base + BFCh offset = 8BFCh



DDR_DDR_IP_REV2 field descriptions

Field	Description
0–7 -	This field is reserved. Reserved
8–15 IP_INT	IP block integration options

Table continues on the next page...

DDR_DDR_IP_REV2 field descriptions (continued)

Field	Description
16–23 -	This field is reserved. Reserved
24–31 IP_CFG	IP block configuration options

12.4.40 DDR Enhanced Optimization Register (DDR_DDR_EOR)

The DDR Enhanced Optimization Register provides the enable and controls for some performance enhancements.

Address: 8000h base + C00h offset = 8C00h

DDR_DDR_EOR field descriptions

Field	Description
0 RD_BDW_OPT_DIS	Read bandwidth optimization disable. This bit controls an optimization feature to improve the read bandwidth. 0 The read bandwidth is optimized during transaction scheduling. 1 The read bandwidth is not optimized.
1–4 -	This field is reserved. Reserved, should be cleared.
5–7 RD_REOD_DIS	Read reorder disable. 1nn Level 1 reordering is disabled. n1n Level 2 reordering is disabled. nn1 Level 3 reordering is disabled.

Table continues on the next page...

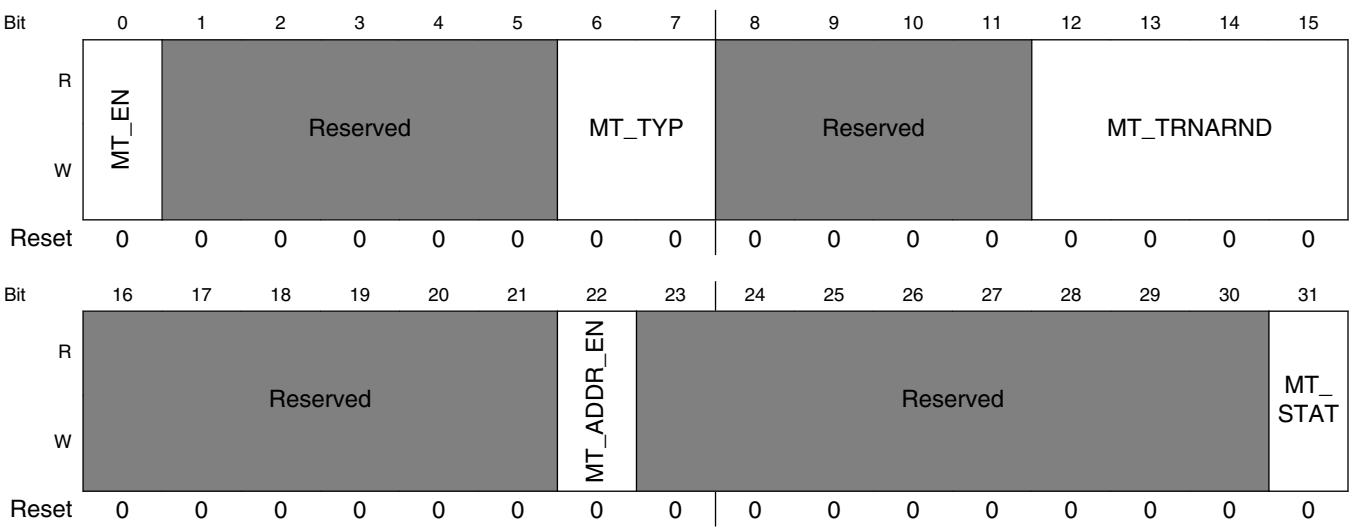
DDR_DDR_EOR field descriptions (continued)

Field	Description
8–10 -	This field is reserved. Reserved, should be cleared.
11 WR_REOD_DIS	Write reorder disable. 0 Write reordering is enabled. 1 Write reordering is disabled.
12–31 -	This field is reserved. Reserved, should be cleared.

12.4.41 DDR Memory Test Control Register (DDR_DDR_MTCR)

The DDR Memory Test Control Register provides the enable and controls for an automatic memory test.

Address: 8000h base + D00h offset = 8D00h



DDR_DDR_MTCR field descriptions

Field	Description
0 MT_EN	Memory Test Enable. This bit can be set by software to enable the memory test. Based on the value of MT_TYP, the controller will either issue writes only, reads only, or it will issue writes and reads.The memory controller will issue transactions throughout all of memory, as defined by the CSn_CONFIG and CSn_BNDS registers. The memory controller will check the data during this test. In addition, the ERR_DETECT register should be read by software if ECC is enabled after the memory test to ensure there were no ECC errors. If an ECC error is detected, the error capture registers will hold the address, data, and attributes captured for the first fail. If there is no ECC error and the test failed (for a data miscompare), then the capture registers will hold information for the transaction that caused the first data

Table continues on the next page...

DDR_DDR_MTCR field descriptions (continued)

Field	Description
	miscompare. Note that transactions with ECC errors will take priority in the capture registers. Hardware will clear MT_EN after the memory test is complete. This bit can be set before DDR_SDRAM_CFG[MEM_EN] is set, or it can be set again after the memory controller has been enabled. **NOTE:** The memory test may not be used if memory controller interleaving has been enabled via CS0_CONFIG[INTLV_EN] and DDR_MTCR[MT_TRNARND]=0000. 0 Memory test has been disabled. 1 Memory test has been enabled. Hardware will clear this bit when it is complete.
1–5 -	This field is reserved. Reserved
6–7 MT_TYP	Memory Test Type. This field will determine if the memory test will issue writes only, reads only, or both writes and reads. Note that the 'read only' test should not be used unless memory has already been initialized. 00 Memory test will issue writes and reads. 01 Memory test will issue writes only. 10 Memory test will issue reads only. 11 Reserved
8–11 -	This field is reserved. Reserved
12–15 MT_TRNARND	Memory Test Turnaround. This field determines how many writes will be issued during the memory test before the reads to the same addresses will be issued. This can be used to allow longer streams of writes/reads, and it can be used to test the write->read and read->write turnarounds in a stressful manner. This field is only relevant if MT_TYP is set to 2'b00. 0000 Entire memory will be written before read transactions are issued. 0001 Total write/read streams will be 1 transaction each. 0010 Total write/read streams will be 2 transactions each. 0011 Total write/read streams will be 4 transactions each. 0100-1111 Reserved
16–21 -	This field is reserved. Reserved
22 MT_ADDR_EN	Memory Test Address Range Enable. If this bit is set, then the address range defined in the DDR_MT_ST_EXT_ADDR, DDR_MT_ST_ADDR, DDR_MT_END_EXT_ADDR, and DDR-MT_END_ADDR registers will be used. 0 Full memory range defined by CSn_BNDS registers will be used for the memory test. 1 Memory range defined by DDR_MT_ST_EXT_ADDR, DDR_MT_ST_ADDR, DDR_MT_END_EXT_ADDR, and DDR_MT_END_ADDR is used.
23–30 -	This field is reserved. Reserved
31 MT_STAT	Memory Test Status. After hardware clears MT_EN, this bit will be set if there was a fail. If there is a fail during the memory test (that is, a data miscompare), then this bit will be set at the same time that MT_EN is cleared. Software can clear this bit after it has been set. 0 No fail has been detected. 1 A data miscompare was detected during the memory test.

12.4.42 DDR Memory Test Pattern n Register (DDR_DDR_MTP)

The DDR memory test pattern n register provides the data pattern that will be written during the nth set of 32-bits of each 40-byte memory test pattern. This is used when DDR_MTCR[MT_EN] is set to enable the memory write/read test.

NOTE

Memory test read compares data that is written in this register.

Address: 8000h base + D20h offset = 8D20h

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	DDR_PATT																															
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

DDR_DDR_MTP field descriptions

Field	Description
0–31 DDR_PATT	This 32-bit pattern will be used during the memory test if enabled via DDR_MTCR[MT_EN]. This memory test will write/read a programmable 40-byte pattern. The 10 DDR_MTPn registers will create the 40-byte pattern that is used.

12.4.43 DDR Memory Test Start Extended Address (DDR_DDR_MT_ST_EXT_ADDR)

The DDR memory test start extended address register provides the extended address that will be used with DDR_MTCR[MT_ADDR_EN] is set.

Address: 8000h base + D60h offset = 8D60h

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																								MT_ST_EXT_ADDR							
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

DDR_DDR_MT_ST_EXT_ADDR field descriptions

Field	Description
0–23 -	This field is reserved. Reserved
24–31 MT_ST_EXT_ ADDR	This field represents the starting extended address that will be used when MTCR[MT_ADDR_EN] is set.

12.4.44 DDR Memory Test Start Address (DDR_DDR_MT_ST_ADDR)

The DDR memory test start address register provides the address that will be used with DDR_MTCR[MT_ADDR_EN] is set.

Address: 8000h base + D64h offset = 8D64h

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	MT_ST_ADDR																															
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

DDR_DDR_MT_ST_ADDR field descriptions

Field	Description
0–31 MT_ST_ADDR	This field represents the starting address that will be used when MTCR[MT_ADDR_EN] is set.

12.4.45 DDR Memory Test End Extended Address (DDR_DDR_MT_END_EXT_ADDR)

The DDR memory test end extended address register provides the extended address that will be used with DDR_MTCR[MT_ADDR_EN] is set.

Address: 8000h base + D68h offset = 8D68h

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																								MT_END_EXT_ADDR							
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

DDR_DDR_MT_END_EXT_ADDR field descriptions

Field	Description
0–23 -	This field is reserved. Reserved
24–31 MT_END_EXT_ ADDR	This field represents the ending extended address that will be used when MTCR[MT_ADDR_EN] is set.

12.4.46 DDR Memory Test End Address (DDR_DDR_MT_END_ADDR)

The DDR memory test end address register provides the address that will be used with DDR_MTCR[MT_ADDR_EN] is set.

Address: 8000h base + D6Ch offset = 8D6Ch

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	MT_END_ADDR																															
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

DDR_DDR_MT_END_ADDR field descriptions

Field	Description
0–31 MT_END_ADDR	This field represents the ending address that will be used when MTCR[MT_ADDR_EN] is set.

12.4.47 Memory data path error injection mask high (DDR_DATA_ERR_INJECT_HI)

Address: 8000h base + E00h offset = 8E00h

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	EIMH																															
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

DDR_DATA_ERR_INJECT_HI field descriptions

Field	Description
0–31 EIMH	Error injection mask high data path. Used to test ECC by forcing errors on the high word of the data path. Setting a bit causes the corresponding data path bit to be inverted on memory bus writes.

12.4.48 Memory data path error injection mask low (DDR_DATA_ERR_INJECT_LO)

Address: 8000h base + E04h offset = 8E04h

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	EIML																															
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

DDR_DATA_ERR_INJECT_LO field descriptions

Field	Description
0–31 EIML	Error injection mask low data path. Used to test ECC by forcing errors on the low word of the data path. Setting a bit causes the corresponding data path bit to be inverted on memory bus writes.

12.4.49 Memory data path error injection mask ECC (DDR_ECC_ERR_INJECT)

The memory data path error injection mask ECC register sets the ECC mask, enables errors to be written to ECC memory, and allows the ECC byte to mirror the most significant data byte. In addition, a single address parity error may be injected through this register.

Address: 8000h base + E08h offset = 8E08h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Reserved															APIEN
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						EMB	EIEN	EEIM							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DDR_ECC_ERR_INJECT field descriptions

Field	Description
0–14 -	This field is reserved. Reserved
15 APIEN	Address parity error injection enable. This bit will be cleared by hardware after a single address parity error has been injected. 0 Address parity error injection disabled. 1 Address parity error injection enabled.
16–21 -	This field is reserved. Reserved
22 EMB	ECC mirror byte 0 Mirror byte functionality disabled. 1 Mirror the most significant data path byte onto the ECC byte.

Table continues on the next page...

DDR_ECC_ERR_INJECT field descriptions (continued)

Field	Description
23 EIEN	Error injection enable 0 Error injection disabled. 1 Error injection enabled. This applies to the data mask bits, the ECC mask bits, and the ECC mirror bit. Note that error injection should not be enabled until the memory controller has been enabled via DDR_SDRAM_CFG[MEM_EN].
24–31 EEIM	ECC error injection mask. Setting a mask bit causes the corresponding ECC bit to be inverted on memory bus writes.

12.4.50 Memory data path read capture high (DDR_CAPTURE_DATA_HI)

The memory data path read capture high register stores the high word of the read data path during error capture.

Address: 8000h base + E20h offset = 8E20h

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	ECHD																															
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

DDR_CAPTURE_DATA_HI field descriptions

Field	Description
0–31 ECHD	Error capture high data path. Captures the high word of the data path when errors are detected.

12.4.51 Memory data path read capture low (DDR_CAPTURE_DATA_LO)

The memory data path read capture low register stores the low word of the read data path during error capture.

Address: 8000h base + E24h offset = 8E24h

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	ECLD																															
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

DDR_CAPTURE_DATA_LO field descriptions

Field	Description
0–31 ECLD	Error capture low data path. Captures the low word of the data path when errors are detected.

12.4.52 Memory data path read capture ECC (DDR_CAPTURE_ECC)

The memory data path read capture ECC register stores the ECC syndrome bits that were on the data bus when an error was detected.

Address: 8000h base + E28h offset = 8E28h

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	ECE																															
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

DDR_CAPTURE_ECC field descriptions

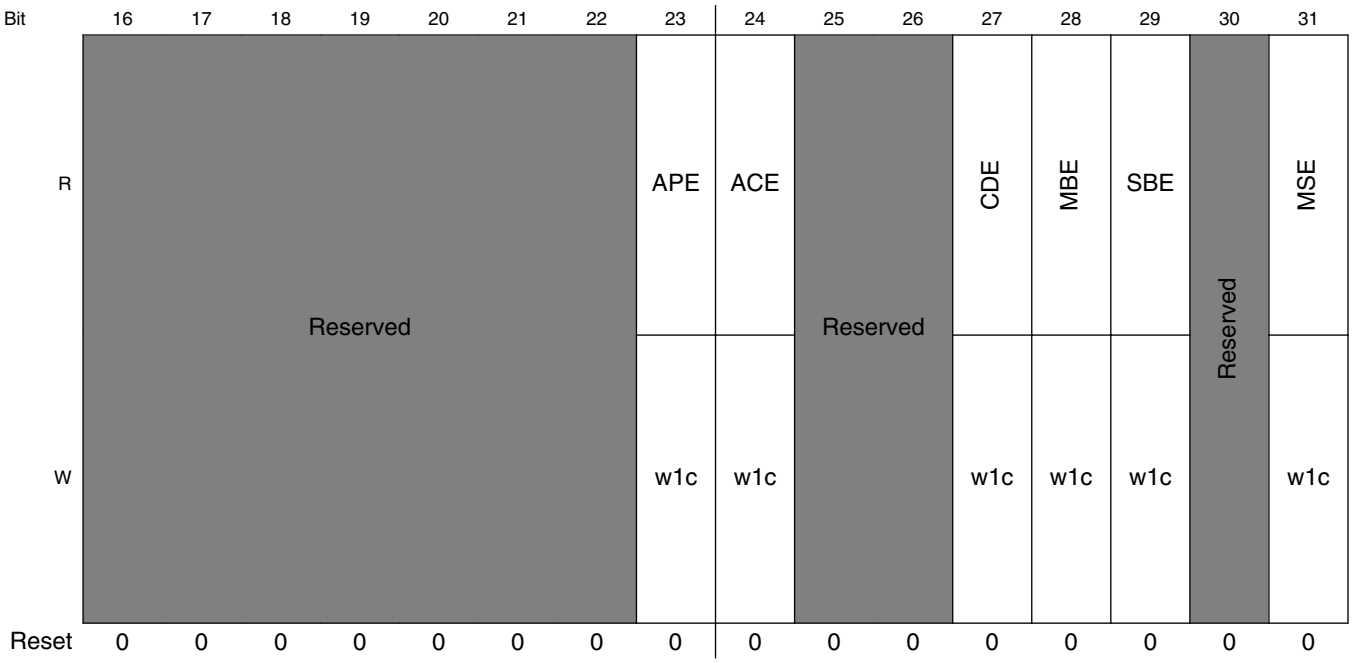
Field	Description
0–31 ECE	Error capture ECC. Captures the ECC bits on the data path whenever errors are detected. 64-bit mode: In 64-bit mode, only 24:31 should be used, although 0:7, 8:15, and 16:23 show the 8-bit ECC code replicated. • 0:7 should be ignored • 8:15 should be ignored • 16:23 should be ignored • 24:31 all 64 bits 32-bit mode: • 0:7 should be ignored • 8:15 8-bit ECC for the 32 bits in beats 0, 2, 4, 6 in 32-bit bus mode • 16:23 should be ignored • 24:31 8-bit ECC for the 32 bits in beats 1, 3, 5, 7 in 32-bit bus mode

12.4.53 Memory error detect (DDR_ERR_DETECT)

The memory error detect register stores the detection bits for multiple memory errors, single- and multiple-bit ECC errors, and memory select errors. It is a read/write register. A bit can be cleared by writing a one to the bit. System software can determine the type of memory error by examining the contents of this register. If an error is disabled with ERR_DISABLE, the corresponding error is never detected or captured in ERR_DETECT.

Address: 8000h base + E40h offset = 8E40h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MME	Reserved														
W	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



DDR_ERR_DETECT field descriptions

Field	Description
0 MME	Multiple memory errors. This bit is cleared by software writing a 1. 0 Multiple memory errors of the same type were not detected. 1 Multiple memory errors of the same type were detected.
1–22 -	This field is reserved. Reserved
23 APE	Address parity error. This bit is cleared by software writing a 1. 0 An address parity error has not been detected. 1 An address parity error has been detected.
24 ACE	Automatic calibration error. This bit is cleared by software writing a 1. 0 An automatic calibration error has not been detected. 1 An automatic calibration error has been detected.
25–26 -	This field is reserved. Reserved
27 CDE	Corrupted data error. This bit is cleared by software writing a 1. 0 A corrupted data error has not been detected. 1 A corrupted data error has been detected. This bit will be set if the actual ECC is inverted from the expected ECC during a read command. The memory controller will intentionally invert the ECC code if DDR_SDRAM_CFG_2[CD_DIS] is cleared when corrupted data is written to memory. The ERR_DETECT[MBE] bit will also be set when a corrupted data error is detected. Note it is also possible for a 2-bit data error to cause the ECC code to become inverted, which would either mask a corrupted data error or cause a normal multi-bit error to also appear as a corrupted data error.
28 MBE	Multiple-bit error. This bit is cleared by software writing a 1.

Table continues on the next page...

DDR_ERR_DETECT field descriptions (continued)

Field	Description
	0 A multiple-bit error has not been detected. 1 A multiple-bit error has been detected.
29 SBE	Single-bit ECC error. This bit is cleared by software writing a 1. 0 The number of single-bit ECC errors detected has not crossed the threshold set in ERR_SBE[SBET]. 1 The number of single-bit ECC errors detected crossed the threshold set in ERR_SBE[SBET].
30 -	This field is reserved. Reserved
31 MSE	Memory select error. This bit is cleared by software writing a 1. 0 A memory select error has not been detected. 1 A memory select error has been detected.

12.4.54 Memory error disable (DDR_ERR_DISABLE)

The memory error disable register allows selective disabling of the DDR controller's error detection circuitry. Disabled errors are not detected or reported.

Address: 8000h base + E44h offset = 8E44h

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							APED	ACED	Reserved		CDED	MBED	SBED	Reserved	MSED
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

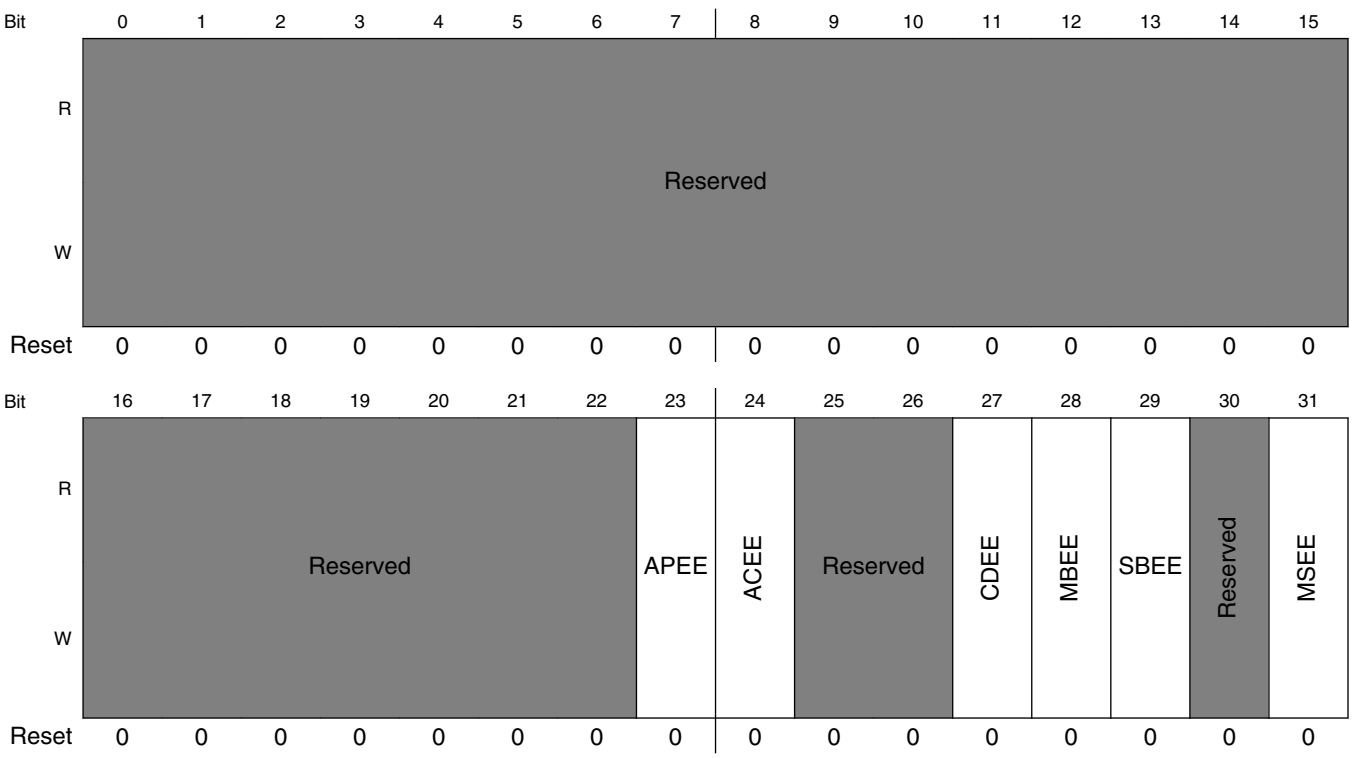
DDR_ERR_DISABLE field descriptions

Field	Description
0–22 -	This field is reserved. Reserved
23 APED	Address parity error disable 0 Address parity errors are detected if (DDR_SDRAM_CFG2[AP_EN] is set. They are reported if ERR_INT_EN[APEE] is set. 1 Address parity errors are not detected or reported.
24 ACED	Automatic calibration error disable 0 Automatic calibration errors are enabled. 1 Automatic calibration errors are disabled.
25–26 -	This field is reserved. Reserved
27 CDED	Corrupted data error disable 0 Corrupted data error checking is enabled. 1 Corrupted data error checking is disabled.
28 MBED	Multiple-bit ECC error disable 0 Multiple-bit ECC errors are detected if DDR_SDRAM_CFG[ECC_EN] is set. They are reported if ERR_INT_EN[MBEE] is set. See Error Management for more information.MBED must be zero and ERR_INT_EN[MBEE] and ECC_EN must be one to ensure that an interrupt is generated. 1 Multiple-bit ECC errors are not detected or reported.
29 SBED	Single-bit ECC error disable 0 Single-bit ECC errors are enabled. 1 Single-bit ECC errors are disabled.
30 -	This field is reserved. Reserved
31 MSED	Memory select error disable 0 Memory select errors are enabled. 1 Memory select errors are disabled.

12.4.55 Memory error interrupt enable (DDR_ERR_INT_EN)

The memory error interrupt enable register enables ECC interrupts or memory select error interrupts. When an enabled interrupt condition occurs, the internal *int_b* signal is asserted to the programmable interrupt controller (PIC).

Address: 8000h base + E48h offset = 8E48h



DDR_ERR_INT_EN field descriptions

Field	Description
0–22 -	This field is reserved. Reserved
23 APEE	Address parity error interrupt enable 0 Address parity errors cannot generate interrupts. 1 Address parity errors generate interrupts.
24 ACEE	Automatic calibration error interrupt enable 0 Automatic calibration errors cannot generate interrupts. 1 Automatic calibration errors generate interrupts.
25–26 -	This field is reserved. Reserved

Table continues on the next page...

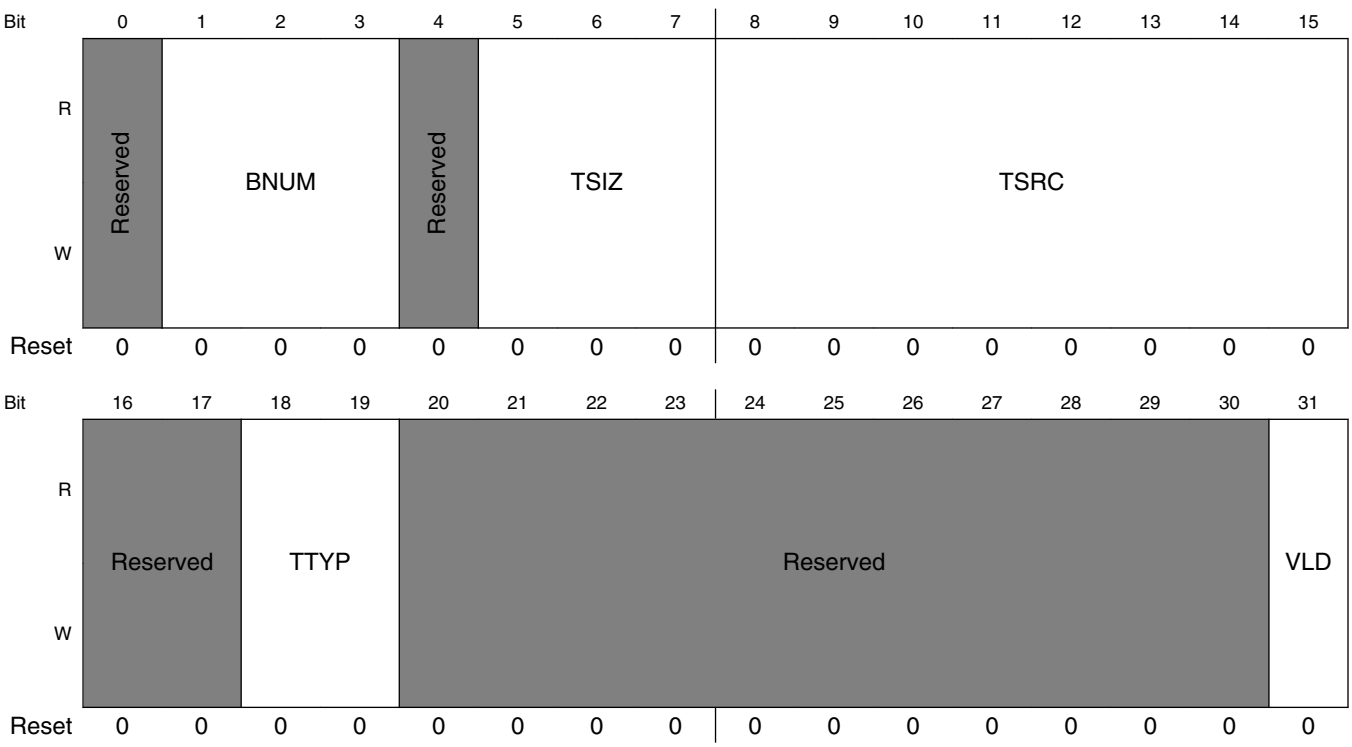
DDR_ERR_INT_EN field descriptions (continued)

Field	Description
27 CDEE	Corrupted data error interrupt enable 0 Corrupted data errors cannot generate interrupts. 1 Corrupted data errors generate interrupts.
28 MBEE	Multiple-bit ECC error interrupt enable. See Error Management for more information. Note that uncorrectable read errors may cause an interrupt. ERR_DISABLE[MBED] must be zero and MBEE and DDR_SDRAM_CFG[ECC_EN] must be set to ensure that an interrupt is generated. 0 Multiple-bit ECC errors cannot generate interrupts. 1 Multiple-bit ECC errors generate interrupts.
29 SBEE	Single-bit ECC error interrupt enable 0 Single-bit ECC errors cannot generate interrupts. 1 Single-bit ECC errors generate interrupts.
30 -	This field is reserved. Reserved
31 MSEE	Memory select error interrupt enable 0 Memory select errors do not cause interrupts. 1 Memory select errors generate interrupts.

12.4.56 Memory error attributes capture (DDR_CAPTURE_ATTRIBUTES)

The memory error attributes capture register sets attributes for errors including type, size, source, and others.

Address: 8000h base + E4Ch offset = 8E4Ch



DDR_CAPTURE_ATTRIBUTES field descriptions

Field	Description
0 -	This field is reserved. Reserved
1–3 BNUM	Data beat number. Captures the doubleword number for the detected error. Relevant only for ECC errors.
4 -	This field is reserved. Reserved
5–7 TSIZ	Transaction size for the error. Captures the transaction size in double words. 000 8 double words 001 1 double word 010 2 double words 011 3 double words 100 4 double words

Table continues on the next page...

DDR_CAPTURE_ATTRIBUTES field descriptions (continued)

Field	Description
	101 5 double word 110 6 double words 111 7 double words
8–15 TSRC	Transaction source for the error. See Global Source and Target IDs for the defined encodings.
16–17 -	This field is reserved. Reserved
18–19 TTYP	Transaction type for the error. 00 Reserved 01 Write 10 Read 11 Read-modify-write
20–30 -	This field is reserved. Reserved
31 VLD	Valid. Set as soon as valid information is captured in the error capture registers.

12.4.57 Memory error address capture (DDR_CAPTURE_ADDRESS)

The memory error address capture register holds the 32 lsbs of a transaction when a DDR ECC error is detected.

Address: 8000h base + E50h offset = 8E50h

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	CADDR																															
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

DDR_CAPTURE_ADDRESS field descriptions

Field	Description
0–31 CADDR	Captured address. Captures the 32 lsbs of the transaction address when an error is detected.

12.4.58 Memory error extended address capture (DDR_CAPTURE_EXT_ADDRESS)

The memory error extended address capture register holds the four most significant transaction bits when an error is detected.

Address: 8000h base + E54h offset = 8E54h

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																								CEADDR							
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

DDR_CAPTURE_EXT_ADDRESS field descriptions

Field	Description
0–23 -	This field is reserved. Reserved
24–31 CEADDR	Captured extended address. Captures the 8 msbs of the transaction address when an error is detected

12.4.59 Single-Bit ECC memory error management (DDR_ERR_SBE)

The single-bit ECC memory error management register stores the threshold value for reporting single-bit errors and the number of single-bit errors counted since the last error report. When the counter field reaches the threshold, it wraps back to the reset value (0). If necessary, software must clear the counter after it has managed the error.

Address: 8000h base + E58h offset = 8E58h

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								SBET								Reserved								SBEC							
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

DDR_ERR_SBE field descriptions

Field	Description
0–7 -	This field is reserved. Reserved
8–15 SBET	Single-bit error threshold. Establishes the number of single-bit errors that must be detected before an error condition is reported.

Table continues on the next page...

DDR_ERR_SBE field descriptions (continued)

Field	Description
16–23 -	This field is reserved. Reserved
24–31 SBEC	Single-bit error counter. Indicates the number of single-bit errors detected and corrected since the last error report. If single-bit error reporting is enabled, an error is reported and an interrupt is generated when this value equals SBET. SBEC is automatically cleared when the threshold value is reached.

12.5 DDR Functional Description

The DDR SDRAM controller controls processor and I/O interactions with system memory. It provides support for JEDEC-compliant DDR3/3L SDRAMs. The memory system allows a wide range of memory devices to be mapped to any arbitrary chip select, and support is provided for registered DIMMs and unbuffered DIMMs. However, registered DIMMs cannot be mixed with unbuffered DIMMs.

The figure below is a high-level block diagram of the DDR memory controller. Requests are received from the internal mastering device and the address is decoded to generate the physical bank, logical bank, row, and column addresses. The transaction is compared with values in the row open table to determine if the address maps to an open page. If the transaction does not map to an open page, an active command is issued.

The memory interface supports as many as four physical banks of 64-/72-bit wide or 32-/40bit bit wide memory. Bank sizes up to 4 Gbytes are supported, providing up to a maximum of 16 Gbytes of DDR main memory.

Programmable parameters allow for a variety of memory organizations and timings. Optional error checking and correcting (ECC) protection is provided for the DDR SDRAM data bus. Using ECC, the DDR memory controller detects and corrects all single-bit errors within the 64- or 32- -bit data bus, detects all double-bit errors within the 64- or 32- bit data bus, and detects all errors within a nibble. The controller allows as many as 32 pages to be open simultaneously. The amount of time (in clock cycles) the pages remain open is programmable with DDR_SDRAM_INTERVAL[BSTOPRE].

Note that the examples below show x8 memory devices.

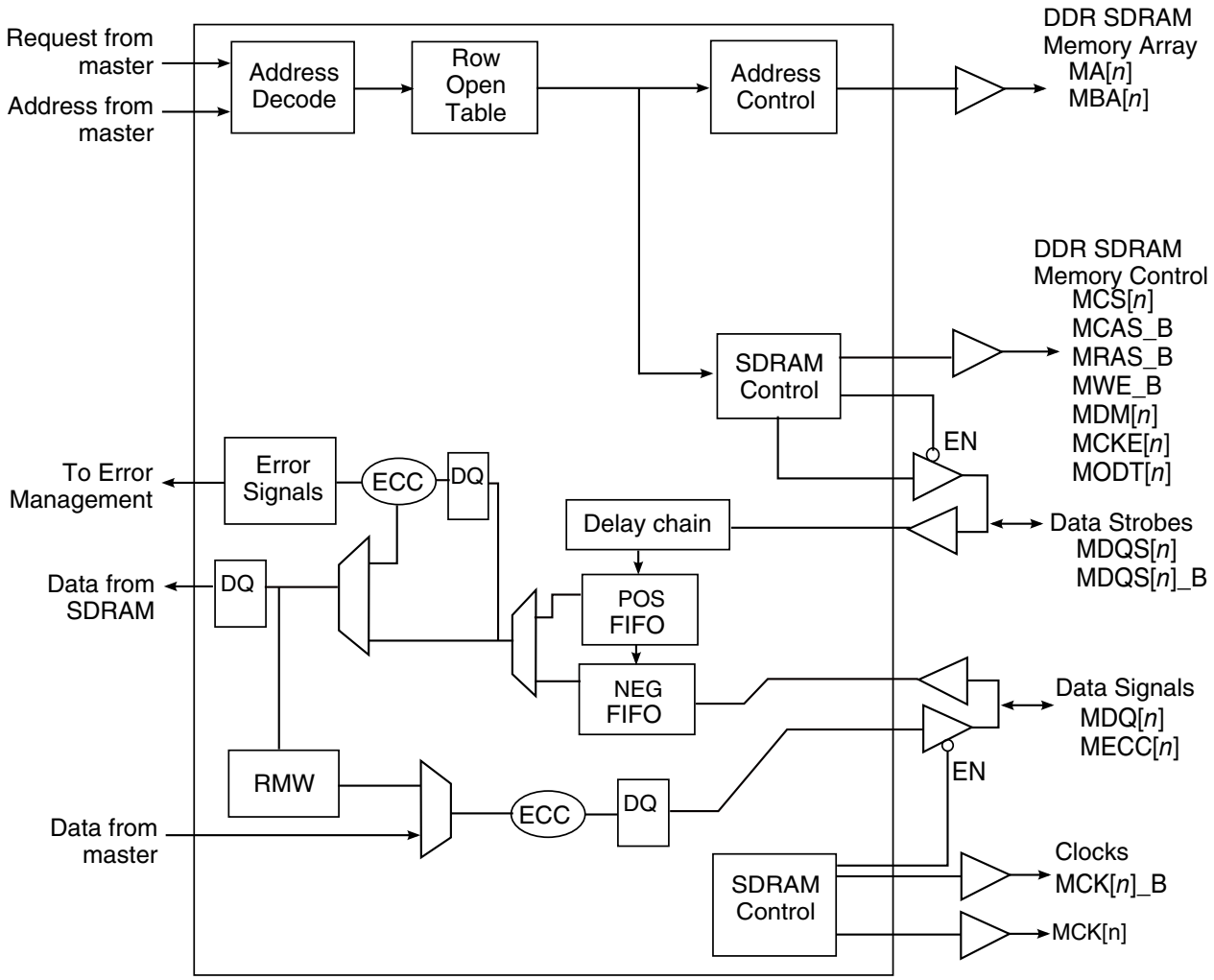


Figure 12-73. DDR Memory Controller Block Diagram

Read and write accesses to memory are burst oriented; accesses start at a selected location and continue for a programmed number of higher locations (4 or 8) in a programmed sequence. Accesses to closed pages start with the registration of an ACTIVE command followed by a READ or WRITE. (Accessing open pages does not require an ACTIVE command.) The address bits registered coincident with the activate command specifies the logical bank and row to be accessed. The address coincident with the READ or WRITE command specify the logical bank and starting column for the burst access.

The data interface is source synchronous, meaning whatever sources the data also provides a clocking signal to synchronize data reception. These bidirectional data strobes (MDQS[0:8]) are inputs to the controller during reads and outputs during writes. The DDR SDRAM specification requires the data strobe signals to be centered within the data tenure during writes and to be offset by the controller to the center of the data tenure during reads. This delay is implemented in the controller for both reads and writes.

When ECC is enabled, 1 clock cycle is added to the read path to check ECC and correct single-bit errors. ECC generation does not add a cycle to the write path.

The address and command interface is also source synchronous, although 1/8 cycle adjustments are provided for adjusting the clock alignment.

The figure below shows an example DDR SDRAM configuration with four logical banks.

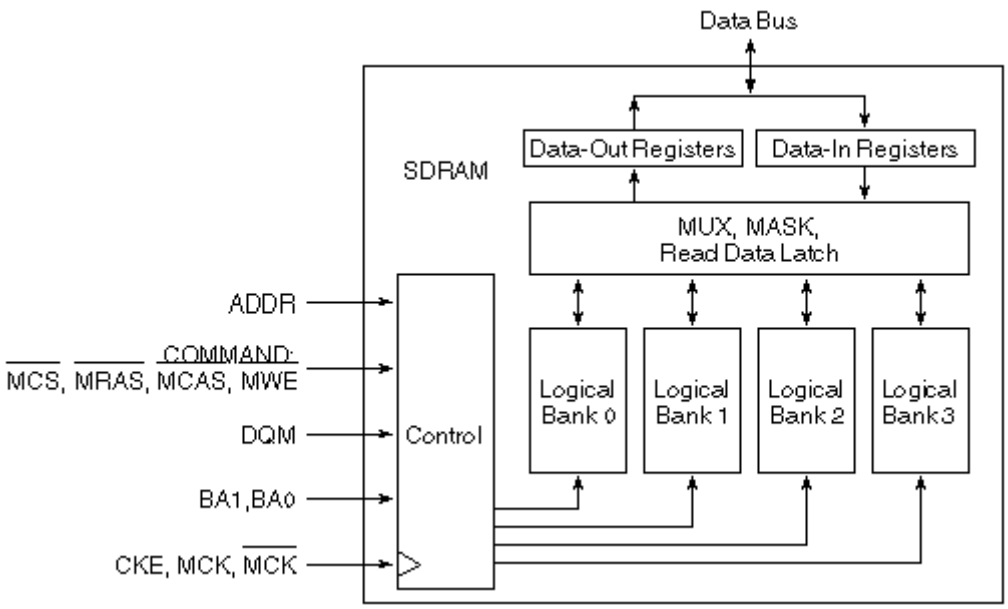


Figure 12-74. Typical Dual Data Rate SDRAM Internal Organization

The figure below shows some typical signal connections.

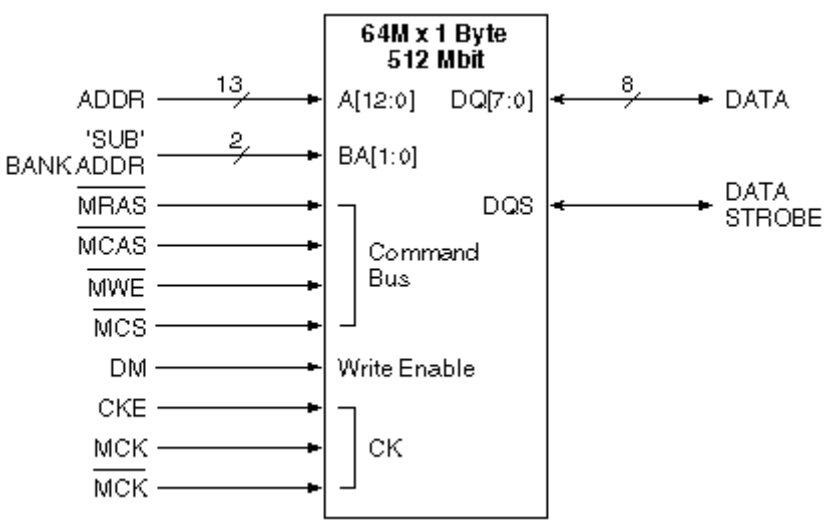
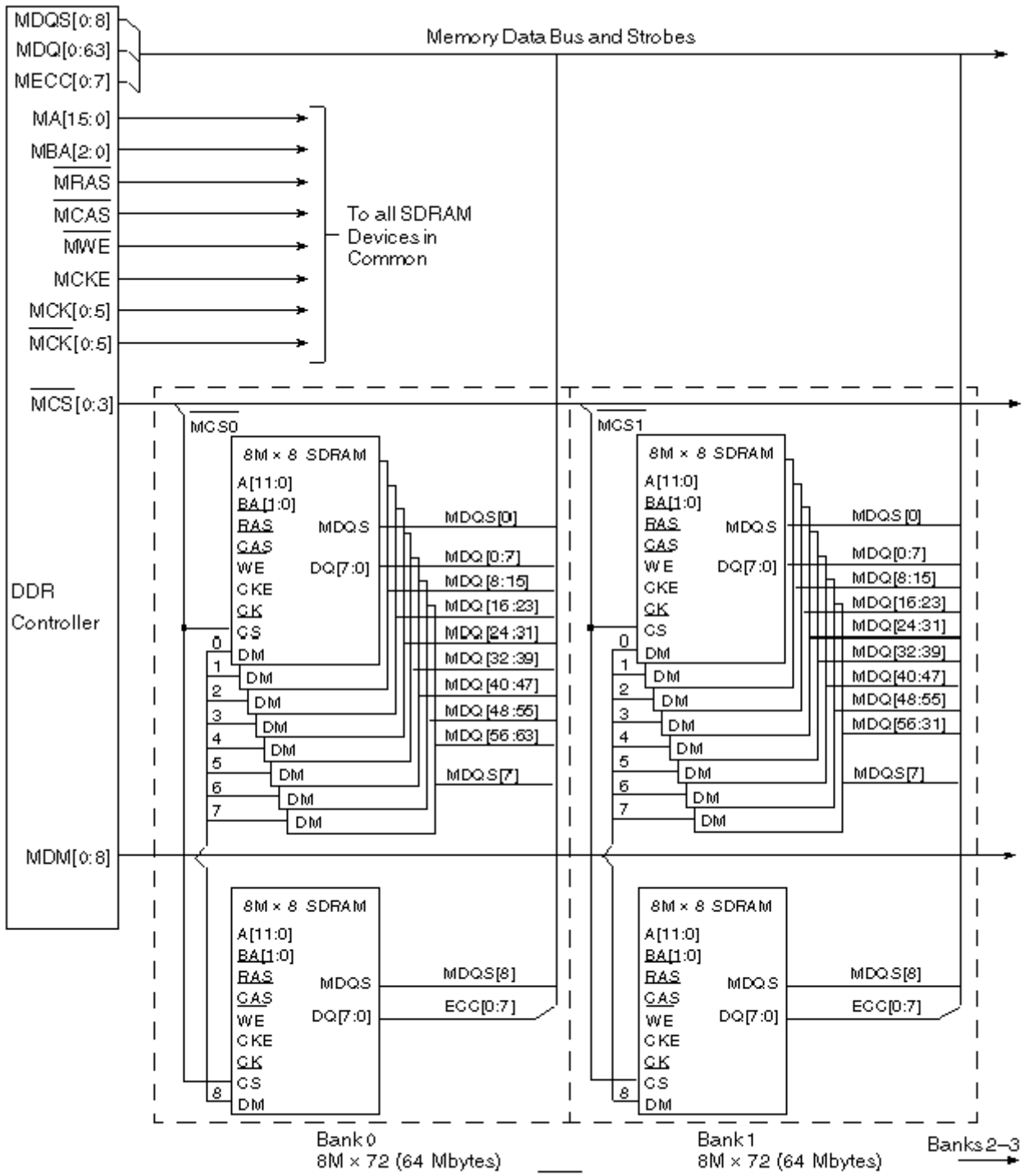


Figure 12-75. Typical DDR SDRAM Interface Signals

The figure below shows an example DDR SDRAM configuration with four physical banks each comprised of nine 8M x 8 DDR modules for a total of 256 Mbytes of system memory. One of the nine modules is used for the memory's ECC checking function. Certain address and control lines may require buffering. Analysis of the device's AC timing specifications, desired memory operating frequency, capacitive loads, and board routing loads can assist the system designer in deciding signal buffering requirements. The DDR memory controllerdrives 16 address pins, but in this example the DDR SDRAM devices use only 12 bits.



1. All signals are connected in common (in parallel) except for MCS[0:3], MCK[0:5], MDM[0:8], and the data bus signals.
2. Each of the MCS[0:3] signals correspond with a separate physical bank of memory.
3. Buffering may be needed if large memory arrays are used.
4. MCK[0:5] may be apportioned among all memory devices. Complementary bus is not shown.

Figure 12-76. Example 256-Mbyte DDR SDRAM Configuration

Error Management explains how the DDR memory controller handles errors.

12.5.1 DDR SDRAM interface operation

The DDR memory controller supports many different DDR SDRAM configurations. SDRAMs with different sizes can be used in the same system. 16 multiplexed address signals and three logical bank select signals support device densities from 64 Mbits to 8 Gbits. Four chip select (CS_B) signals support up to two DIMMs of memory. The DDR SDRAM physical banks can be built from standard memory modules or directly-attached memory devices. The data path to individual physical banks is 64 or 32 bits wide, 72 or 40 bits with ECC. The DDR memory controller supports physical bank sizes from 16 MB to 4 GB. The physical banks can be constructed using x 8, x16 (with 1 DQS per data byte) memory devices. The memory technologies supported are 64 Mbits, 128 Mbits, 256 Mbits, 512 Mbits, 1 Gbit , 2 Gbits, 4 Gbits and 8Gbits. Nine data qualifier (DQM) signals provide byte selection for memory accesses.

NOTE

An 8-bit DDR SDRAM device has a DQM signal and eight data signals (DQ[0:7]). A 16-bit DDR SDRAM device has two DQM signals associated with specific halves of the 16 data signals (DQ[0:7] and DQ[8:15]).

When ECC is enabled, all memory accesses are performed on double-word boundaries (that is, all DQM signals are set simultaneously). However, when ECC is disabled, the memory system uses the DQM signals for byte lane selection when using x8 or x16 devices.

This table shows the DDR memory controller's relationships between data byte lane0-7 , MDM[0:7], MDQS[0:7], and MDQ[0:63] when DDR SDRAM memories are used with x8 or x16 devices.

Table 12-78. Byte-lane-to-data relationship for x8 and x16 devices

Data byte lane	Data bus mask	Data bus strobe	Data bus 64-bit mode
0 (MSB)	MDM[0]	MDQS[0]/MDQS[0]_B	MDQ[0:7]
1	MDM[1]	MDQS[1]/MDQS[1]_B	MDQ[8:15]
2	MDM[2]	MDQS[2]/MDQS[2]_B	MDQ[16:23]
3	MDM[3]	MDQS[3]/MDQS[3]_B	MDQ[24:31]
4	MDM[4]	MDQS[4]/MDQS[4]_B	MDQ[32:39]
5	MDM[5]	MDQS[5]/MDQS[5]_B	MDQ[40:47]
6	MDM[6]	MDQS[6]/MDQS[6]_B	MDQ[48:55]
7 (LSB)	MDM[7]	MDQS[7]/MDQS[7]_B	MDQ[56:63]

12.5.1.1 Supported DDR SDRAM Organizations

Although the DDR memory controller multiplexes row and column address bits onto 16 16 memory address signals and 3 logical bank select signals, a physical bank may be implemented with memory devices requiring fewer than 31 address bits. The physical bank may be configured to provide from 12 to 16 row address bits, plus 3 logical bank-select bits and from 8-11 column address bits.

The table below describes DDR SDRAM device configurations supported by the DDR memory controller.

NOTE

DDR SDRAM is limited to 30 total address bits.

Table 12-79. Supported DDR3 SDRAM Device Configurations

SDRAM Device	Device Configuration	Row x Column x Sub-bank Bits	64-Bit Bank Size	Four Banks of Memory
512 Mbits	128 Mbits x 4	13 x 11 x 3	1 Gbyte	4 Gbytes
512 Mbits	64 Mbits x 8	13 x 10 x 3	512 Mbytes	2 Gbytes
512 Mbits	32 Mbits x 16	12 x 10 x 3	256 Mbytes	1 Gbyte
1 Gbits	256 Mbits x 4	14 x 11 x 3	2 Gbytes	4 Gbytes
1 Gbits	128 Mbits x 8	14 x 10 x 3	1 Gbyte	4 Gbytes
1 Gbits	64 Mbits x 16	13 x 10 x 3	512 Mbytes	2 Gbytes
2 Gbits	512 Mbits x 4	15 x 11 x 3	4 Gbytes	16 Gbytes
2 Gbits	256 Mbits x 8	15 x 10 x 3	2 Gbytes	8 Gbytes
2 Gbits	128 Mbits x 16	14 x 10 x 3	1 Gbyte	4 Gbytes
4 Gbits	1 Gbit x 4	16 x 11 x 3	8 Gbytes	32 Gbytes
4 Gbits	512 Mbits x 8	16 x 10 x 3	4 Gbytes	16 Gbytes
4 Gbits	256 Mbits x 16	15 x 10 x 3	2 Gbytes	8 Gbytes
8 Gbits	2 Gbits x 4	16 x 12 x 3	16 Gbytes	64 Gbytes
8 Gbits	1 Gbit x 8	16 x 11 x 3	8 Gbytes	32 Gbytes
8 Gbits	512 Mbits x 16	16 x 10 x 3	4 Gbytes	16 Gbytes

If a transaction request is issued to the DDR memory controller and the address does not lie within any of the programmed address ranges for an enabled chip select, a memory select error is flagged. Errors are described in detail in Error Management."

By using a memory-polling algorithm at power-on reset or by querying the JEDEC serial presence detect capability of memory modules, system firmware uses the memory-boundary registers to configure the DDR memory controller to map the size of each bank

in memory. The memory controller uses its bank map to assert the appropriate MCSn_B signal for memory accesses according to the provided bank starting and ending addresses. The memory banks are not required to be mapped to a contiguous address space.

12.5.2 DDR SDRAM Address Multiplexing

The tables below show the address bit encodings for each DDR SDRAM configuration. The address presented at the memory controller signals MA[15 :0] use MA[15] as the msb and MA[0] as the lsb. Also, MA[10] is used as the auto-precharge bit in DDR3 mode for reads and writes, so the column address can never use MA[10].

Table 12-80. DDR3 Address Multiplexing for 64-Bit Data Bus with Interleaving and Partial Array Self Refresh Disabled

Row x Col		msb	Address from Core Master																												lsb
		8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37-39
16 x 10 x 3	MRAS_B	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0														
	MBA																	2	1	0											
	MCAS_B																				9	8	7	6	5	4	3	2	1	0	
15 x 10 x 3	MRAS_B		14	13	12	11	10	9	8	7	6	5	4	3	2	1	0														
	MBA																	2	1	0											
	MCAS_B																				9	8	7	6	5	4	3	2	1	0	

Table continues on the next page...

Table 12-80. DDR3 Address Multiplexing for 64-Bit Data Bus with Interleaving and Partial Array Self Refresh Disabled (continued)

Row x Col		msb	Address from Core Master																												lsb
		8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37-39
14 x 10 x 3	MRAS_B			13	12	11	10	9	8	7	6	5	4	3	2	1	0														
	MBA																	2	1	0											
	MCAS_B																				9	8	7	6	5	4	3	2	1	0	
13 x 10 x 3	MRAS_B				12	11	10	9	8	7	6	5	4	3	2	1	0														
	MBA																	2	1	0											
	MCAS_B																				9	8	7	6	5	4	3	2	1	0	
12 x 10 x 3	MRAS_B						11	10	9	8	7	6	5	4	3	2	1	0													
	MBA																		2	1	0										
	MCAS_B																					8	7	6	5	4	3	2	1	0	

Table 12-81. DDR3 Address Multiplexing for 32-Bit Data Bus with Interleaving and Partial Array Self Refresh Disabled

Row x Col		msb	Address from Core Master																														lsb
		8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38-39
16 x 10 x 3	MRAS_B		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0															
	MBA																		2	1	0												
	MCAS_B																					9	8	7	6	5	4	3	2	1	0		
15 x 10 x 3	MRAS_B			14	13	12	11	10	9	8	7	6	5	4	3	2	1	0															
	MBA																		2	1	0												
	MCAS_B																					9	8	7	6	5	4	3	2	1	0		
14 x 10 x 3	MRAS_B				13	12	11	10	9	8	7	6	5	4	3	2	1	0															
	MBA																		2	1	0												
	MCAS_B																					9	8	7	6	5	4	3	2	1	0		
13 x 10 x 3	MRAS_B					12	11	10	9	8	7	6	5	4	3	2	1	0															
	MBA																		2	1	0												
	MCAS_B																					9	8	7	6	5	4	3	2	1	0		
12 x 10 x 3	MRAS_B						11	10	9	8	7	6	5	4	3	2	1	0															
	MBA																		2	1	0												
	MCAS_B																					9	8	7	6	5	4	3	2	1	0		

Chip select interleaving is supported for the memory controller, and is programmed in DDR_SDRAM_CFG[BA_INTLV_CTL]. Interleaving is supported between chip selects 0 and 1 or chip selects 2 and 3. In addition, interleaving between all four chip selects can be enabled. When interleaving is enabled, the chip selects being interleaved must use the same size of memory. If two chip selects are interleaved, then 1 extra bit in the address decode is used for the interleaving to determine which chip select to access. If four chip selects are interleaved, then two extra bits are required in the address decode.

Table 12-82 illustrates examples of address decode when interleaving between two chip selects, and Table 12-83 shows examples of address decode when interleaving between four chip selects.

Table 12-82. Example of Address Multiplexing for 32/64-Bit Data Bus Interleaving Between Two Banks with Partial Array Self Refresh Disabled

Row x Col		msb	Address from Core Master																												lsb
		8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37-39
14 x 10 x 3	MRAS_B		13	12	11	10	9	8	7	6	5	4	3	2	1	0	CSSEL														
	MBA																	2	1	0											
	MCAS_B																				9	8	7	6	5	4	3	2	1	0	
13 x 10 x 3	MRAS_B			12	11	10	9	8	7	6	5	4	3	2	1	0	CSSEL														
	MBA																	2	1	0											
	MCAS_B																				9	8	7	6	5	4	3	2	1	0	

Table 12-83. Example of Address Multiplexing for 64-Bit Data Bus Interleaving Between Four Banks with Partial Array Self Refresh Disabled

Row x Col		msb	Address from Core Master																												lsb
		8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37-39
14 x 10 x 3	MRAS_B	13	12	11	10	9	8	7	6	5	4	3	2	1	0	CS SEL															
	MBA																	2	1	0											
	MCAS_B																				9	8	7	6	5	4	3	2	1	0	
13 x 10 x 3	MRAS_B		12	11	10	9	8	7	6	5	4	3	2	1	0	CS SEL															
	MBA																	2	1	0											
	MCAS_B																				9	8	7	6	5	4	3	2	1	0	

Partial Array Self Refresh (PASR) can be enabled for any chip select using the CS*n*_CONFIG_2[PASR_CFG] fields. If PASR is enabled for a given chip select, then the sub-bank and row decode will be swapped, and the sub-bank will be decoded as the most significant portion of the DRAM address, as shown in Table 12-84. If chip select interleaving and PASR are enabled for a chip select, then the interleaved chip select bit will be placed immediately to the left of the column decode, as shown in Table 12-85.

Table 12-84. DDR2 Address Multiplexing with Partial Array Self Refresh Enabled

Row x Col		msb	Address from Core Master																														lsb
		8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38-39
16 x 10 x 3	MRAS_B					15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0											
	MBA		2	1	0																											
	MCAS_B																					9	8	7	6	5	4	3	2	1	0	
15 x 10 x 3	MRAS_B						14	13	12	11	10	9	8	7	6	5	4	3	2	1	0											
	MBA			2	1	0																										
	MCAS_B																					9	8	7	6	5	4	3	2	1	0	
14 x 10 x 3	MRAS_B							13	12	11	10	9	8	7	6	5	4	3	2	1	0											
	MBA				2	1	0																									
	MCA																					9	8	7	6	5	4	3	2	1	0	

Table continues on the next page...

Table 12-84. DDR2 Address Multiplexing with Partial Array Self Refresh Enabled (continued)

Row x Col		msb	Address from Core Master																														lsb
		8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38-39
	S_B																															
13 x 10 x 3	MRAS_B								12	11	10	9	8	7	6	5	4	3	2	1	0											
	MBA					2	1	0																								
	MCAS_B																					9	8	7	6	5	4	3	2	1	0	
12 x 10 x 3	MRAS_B									11	10	9	8	7	6	5	4	3	2	1	0											
	MBA						2	1	0																							
	MCAS_B																					9	8	7	6	5	4	3	2	1	0	

Table 12-85. Example of Address Multiplexing for 64-bit Data Bus Interleaving Between Two Banks with Partial Array Self Refresh Enabled

Row x Col		msb		Address from Core Master																												lsb
		8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37-39
14 x 10	MRAS_B					13	12	11	10	9	8	7	6	5	4	3	2	1	0	CSSEL											
x3	MBA		2	1	0																										
	MCAS_B																				9	8	7	6	5	4	3	2	1	0	
13 x 10	MRAS_B						12	11	10	9	8	7	6	5	4	3	2	1	0	CSSEL											
x 3	MBA			2	1	0																									
	MCAS_B																				9	8	7	6	5	4	3	2	1	0	

12.5.3 JEDEC Standard DDR SDRAM Interface Commands

The following section describes the commands and timings the controller uses when operating in DDR3 mode.

All read or write accesses to DDR SDRAM are performed by the DDR memory controller using JEDEC standard DDR SDRAM interface commands. The SDRAM device samples command and address inputs on rising edges of the memory clock; data is sampled using both the rising and falling edges of DQS. Data read from the DDR SDRAM is also sampled on both edges of DQS.

The following DDR SDRAM interface commands (summarized in Table 12-86) are provided by the DDR controller. All actions for these commands are described from the perspective of the SDRAM device.

- Row activate-Latches row address and initiates memory read of that row. Row data is latched in SDRAM sense amplifiers and must be restored by a precharge command before another row activate occurs.
- Precharge-Restores data from the sense amplifiers to the appropriate row. Also initializes the sense amplifiers in preparation for reading another row in the memory array, (performing another activate command). Precharge must occur after read or write, if the row address changes on the next open page mode access.
- Read-Latches column address and transfers data from the selected sense amplifier to the output buffer as determined by the column address. During each succeeding

clock edge, additional data is driven without additional read commands. The amount of data transferred is determined by the burst size which defaults to 4.

- Write-Latches column address and transfers data from the data pins to the selected sense amplifier as determined by the column address. During each succeeding clock edge, additional data is transferred to the sense amplifiers from the data pins without additional write commands. The amount of data transferred is determined by the data masks and the burst size, which is set to four by the DDR memory controller.
- Refresh (similar to MCAS_B before MRAS_B)-Causes a row to be read in all logical banks (JEDEC SDRAM) as determined by the refresh row address counter. This refresh row address counter is internal to the SDRAM. After being read, the row is automatically rewritten in the memory array. All logical banks must be in a precharged state before executing a refresh. The memory controller also supports posted refreshes, where several refreshes may be executed at once, and the refresh interval may be extended.
- Mode register set (for configuration)-Allows setting of DDR SDRAM options. These options are: MCAS_B latency, additive latency (for DDR3), write recovery (for DDR3), burst type, and burst length. MCAS_B latency may be chosen as provided by the preferred SDRAM (some SDRAMs provide MCAS_B latency {1,2,3}, some provide MCAS_B latency {1,2,3,4,5}, etc.). Burst type is always sequential. Although some SDRAMs provide burst lengths of 1, 2, 4, 8, and page size, this memory controller supports a burst length of 4. A burst length of 8 is supported for DDR3 memory only. The mode register set command is performed by the DDR memory controller during system initialization. Parameters such as mode register data, MCAS_B latency, burst length, and burst type, are set by software in DDR_SDRAM_MODE[SDMODE] and transferred to the SDRAM array by the DDR memory controller after DDR_SDRAM_CFG[MEM_EN] is set. If DDR_SDRAM_CFG[BI] is set to bypass the automatic initialization, then the MODE registers can be configured through software via use of the DDR_SDRAM_MD_CNTL register.
- Self refresh (for long periods of standby)-Used when the device is in standby for very long periods of time. Automatically generates internal refresh cycles to keep the data in all memory banks refreshed. Before execution of this command, the DDR controller will place all logical banks in a precharged state.

Table 12-86. DDR SDRAM Command Table

Operation	CKE Prev.	CKE Current	MCS_B	MRAS_B	MCAS_B	MWE_B	MBA	MA10	MA
Activate	H	H	L	L	H	H	Logical bank select	Row	Row
Precharge select logical bank	H	H	L	L	H	L	Logical bank select	L	X
Precharge all logical banks	H	H	L	L	H	L	X	H	X

Table continues on the next page...

Table 12-86. DDR SDRAM Command Table (continued)

Operation	CKE Prev.	CKE Current	MCS _B	MRAS _B	MCAS _B	MWE _B	MBA	MA10	MA
Read	H	H	L	H	L	H	Logical bank select	L	Column
Read with auto-precharge	H	H	L	H	L	H	Logical bank select	H	Column
Write	H	H	L	H	L	L	Logical bank select	L	Column
Write with auto-precharge	H	H	L	H	L	L	Logical bank select	H	Column
Mode register set	H	H	L	L	L	L	Opcode	Opcode	Opcode and mode
Auto refresh	H	H	L	L	L	H	X	X	X
Self refresh	H	L	L	L	L	H	X	X	X

12.5.4 DDR SDRAM Interface Timing

For single-beat reads, the DDR memory controller performs a four- (or eight-) beat burst read, but ignores the last three (or seven) beats. Single-beat writes are performed by masking the last three (or seven) beats of the four- (or eight-) beat burst using the data mask MDM[0:8]. If ECC is disabled, writes smaller than double words are performed by appropriately activating the data mask. If ECC is enabled, the controller performs a read-modify write.

NOTE

If a second read or write is pending, reads shorter than four beats are not terminated early even if some data is irrelevant.

To accommodate available memory technologies across a wide spectrum of operating frequencies, the DDR memory controller allows the setting of the intervals defined in the table below with granularity of one memory clock cycle, except for CASLAT, which can be programmed with 1/2 clock granularity.

Table 12-87. DDR SDRAM Interface Timing Intervals

Timing Intervals	Definition
ACTTOACT	The number of clock cycles from a bank-activate command until another bank-activate command within a physical bank. This interval is listed in the AC specifications of the SDRAM as t_{RRD}.
ACTTOPRE	The number of clock cycles from an activate command until a precharge command is allowed. This interval is listed in the AC specifications of the SDRAM as t_{RAS}.
ACTTORW	The number of clock cycles from an activate command until a read or write command is allowed. This interval is listed in the AC specifications of the SDRAM as t_{RCD}.
BSTOPRE	The number of clock cycles to maintain a page open after an access. The page open duration counter is reloaded with BSTOPRE each time the page is accessed (including page hits). When the counter expires, the open page is closed with a SDRAM precharge bank command as soon as possible.

Table continues on the next page...

Table 12-87. DDR SDRAM Interface Timing Intervals (continued)

Timing Intervals	Definition
CASLAT	Used in conjunction with additive latency to obtain the READ latency. The number of clock cycles between the registration of a READ command by the SDRAM and the availability of the first piece of output data. If a READ command is registered at clock edge *n*, and the read latency is *m* clocks, the data is available nominally coincident with clock edge *n* + *m*.
PRETOACT	The number of clock cycles from a precharge command until an activate or a refresh command is allowed. This interval is listed in the AC specifications of the SDRAM as t_{RP}.
REFINT	Refresh interval. Represents the number of memory bus clock cycles between refresh cycles. Depending on DDR_SDRAM_CFG_2[NUM_PR], some number of rows are refreshed in each SDRAM bank during each refresh cycle. The value of REFINT depends on the specific SDRAMs used and the frequency of the interface as t_{RP}.
REFREC	The number of clock cycles from the refresh command until an activate command is allowed. This can be calculated by referring to the AC specification of the SDRAM device. The AC specification indicates a maximum refresh to activate interval in nanoseconds. This field is also combined with TIMING_CFG_3[EXT_REFREC]
WR_DATA_DELAY	Provides different options for the timing between a write command and the write data strobe. This allows write data to be sent later than the nominal time to meet the SDRAM timing requirement between the registration of a write command and the reception of a data strobe associated with the write command. The specification dictates that the data strobe may not be received earlier than 75% of a cycle, or later than 125% of a cycle, from the registration of a write command. This parameter is not defined in the SDRAM specification. It is implementation-specific, defined for the DDR memory controller in TIMING_CFG_2.
WRREC	The number of clock cycles from the last beat of a write until a precharge command is allowed. This interval, write recovery time, is listed in the AC specifications of the SDRAM as t_{WR}.
WRTORD	Last write pair to read command. Controls the number of clock cycles from the last write data pair to the subsequent read command to the same bank as t_{WTR}.

The value of the above parameters (in whole clock cycles) must be set by boot code at system start-up (in the TIMING_CFG_0, TIMING_CFG_1, TIMING_CFG_2, and TIMING_CFG_3 registers as described in DDR SDRAM timing configuration 0 (DDR_TIMING_CFG_0)DDR SDRAM timing configuration 1 (DDR_TIMING_CFG_1)DDR SDRAM timing configuration 2 (DDR_TIMING_CFG_2) and DDR SDRAM timing configuration 3 (DDR_TIMING_CFG_3)) and be kept in the DDR memory controller configuration register space.

The following figures show SDRAM timing for various types of accesses. System software is responsible (at reset) for optimally configuring SDRAM timing parameters. The programmable timing parameters apply to both read and write timing configuration. The configuration process must be completed and the DDR SDRAM initialized before any accesses to SDRAM are attempted.

Figure 12-77 through Figure 12-79 show DDR SDRAM timing for various types of accesses; see Figure 12-78 for a burst read operation, Figure 12-77 for a single-beat write operation, and Figure 12-79 for a burst write operation. Note that all signal transitions occur on the rising edge of the memory bus clock and that single-beat read operations are

identical to burst-reads. These figures assume the CLK_ADJUST is set to 1/2 DRAM cycle, an additive latency of 0 DRAM cycles is used, and the write latency is 1 DRAM cycle.

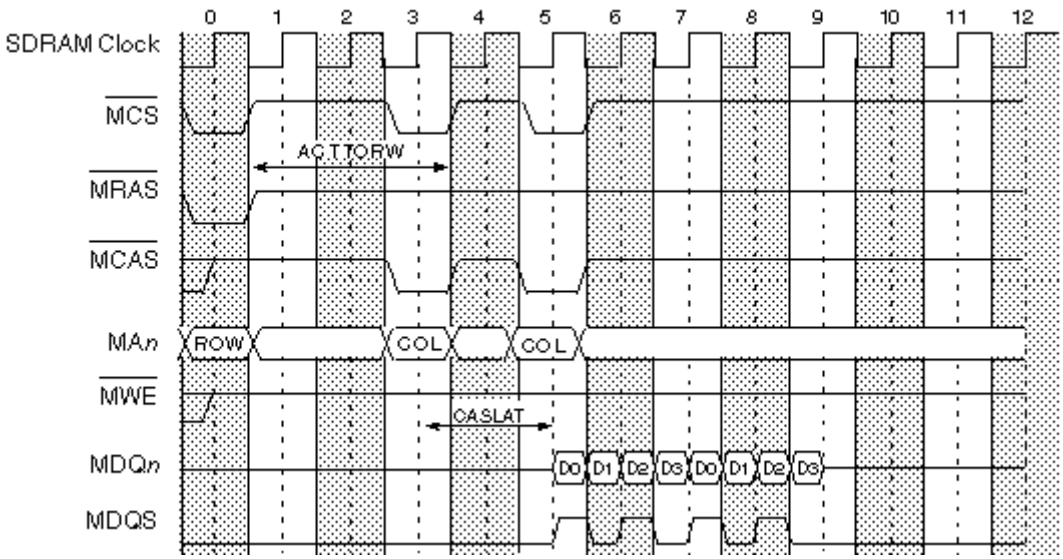


Figure 12-77. DDR SDRAM Burst Read Timing-ACTTORW = 3, MCAS_B Latency = 2

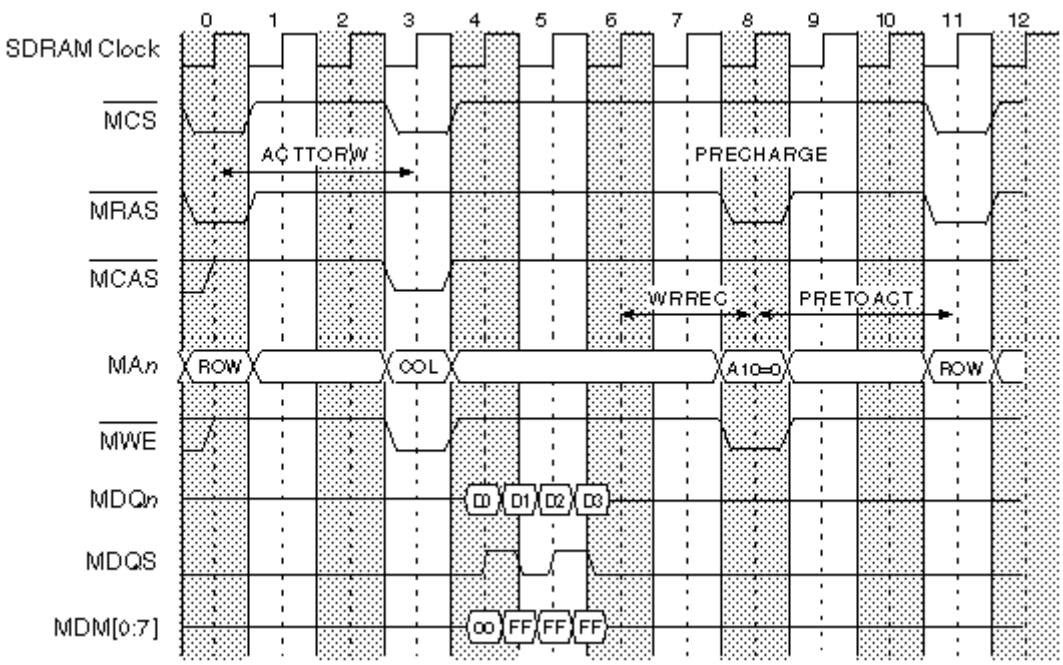


Figure 12-78. DDR SDRAM Single-Beat (Double Word) Write Timing-ACTTORW = 3

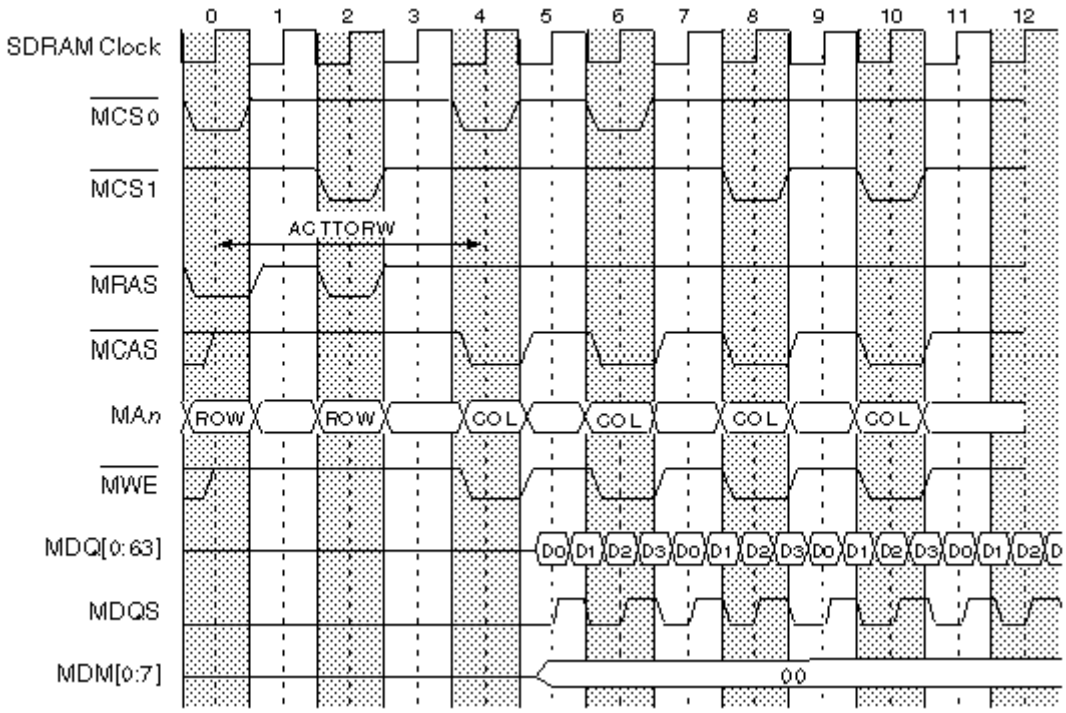


Figure 12-79. DDR SDRAM 4-Beat Burst Write Timing-ACTTORW = 4

12.5.4.1 Clock Distribution

- If running with many devices, zero-delay PLL clock buffers, JEDEC-JESD82 standard, should be used. These buffers were designed for DDR applications.
- A 72 bit x 64 Mbytes DDR bank has 9-byte-wide DDR chips, resulting in 18 DDR chips in a two-bank system. In this case, each MCK/MCK_B signal pair should drive exactly nine devices.
- PCB traces for DDR clock signals should be short, all on the same layer, and of equal length and loading.
- DDR SDRAM manufacturers provide detailed information on PCB layout and termination issues.

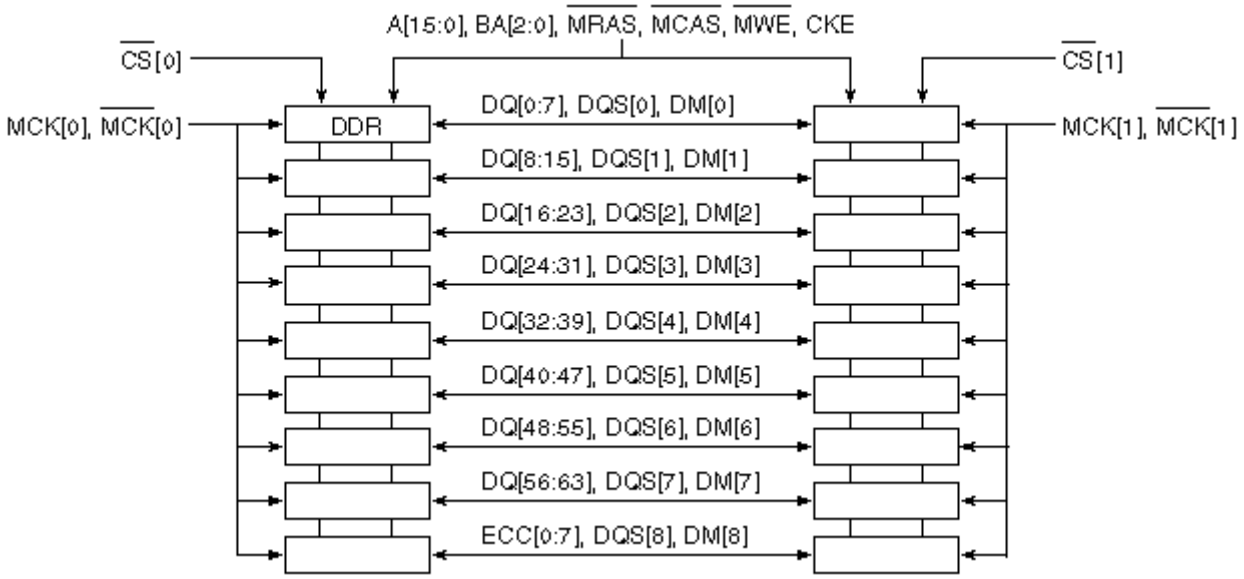


Figure 12-80. DDR SDRAM Clock Distribution Example for x8 DDR SDRAMs

12.5.5 DDR SDRAM Mode-Set Command Timing

The DDR memory controller transfers the mode register set commands to the SDRAM array, and it uses the setting of TIMING_CFG_0[MRS_CYC] for the Mode Register Set cycle time.

The figure below shows the timing of the mode-set command. The first transfer corresponds to the ESDMODE code; the second corresponds to SDMODE. The Mode Register Set cycle time is set to 2 DRAM cycles.

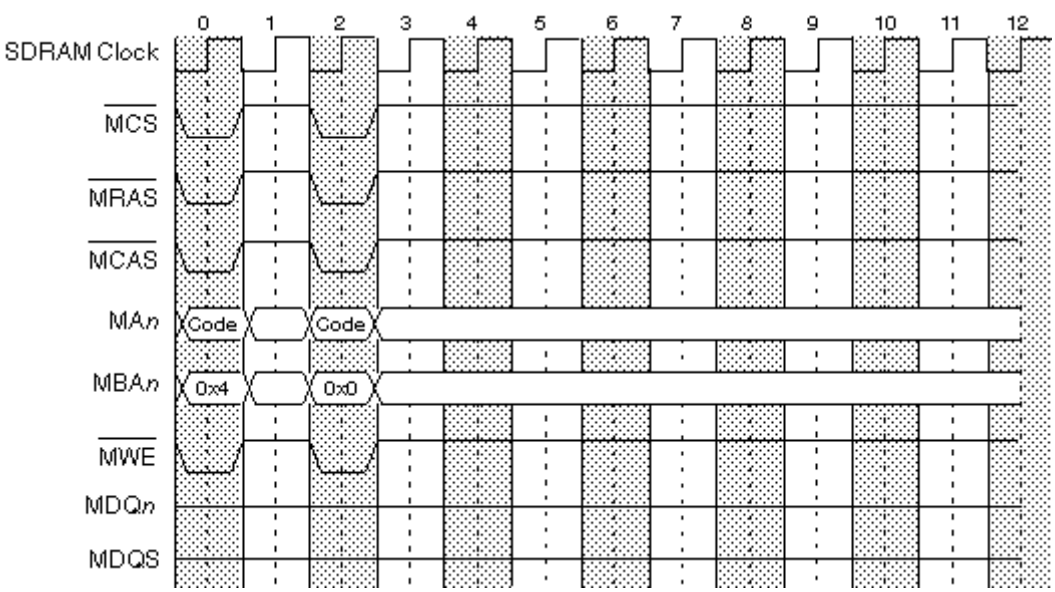


Figure 12-81. DDR SDRAM Mode-Set Command Timing

12.5.6 DDR SDRAM Registered DIMM Mode

To reduce loading, registered DIMMs latch the DDR SDRAM control signals internally before using them to access the array. Setting DDR_SDRAM_CFG[RD_EN] compensates for this delay on the DIMMs' control bus by delaying the data and data mask writes (on SDRAM buses) by an extra SDRAM clock cycle.

NOTE

Application system board must assert the reset signal on DDR memory devices until software is able to program the DDR memory controller configuration registers, and must deassert the reset signal on DDR memory devices before DDR_SDRAM_CFG[MEM_EN] is set. This ensures that the DDR memory devices are held in reset until a stable clock is provided and, further, that a stable clock is provided before memory devices are released from reset.

The figure below shows the registered DDR SDRAM DIMM single-beat write timing.

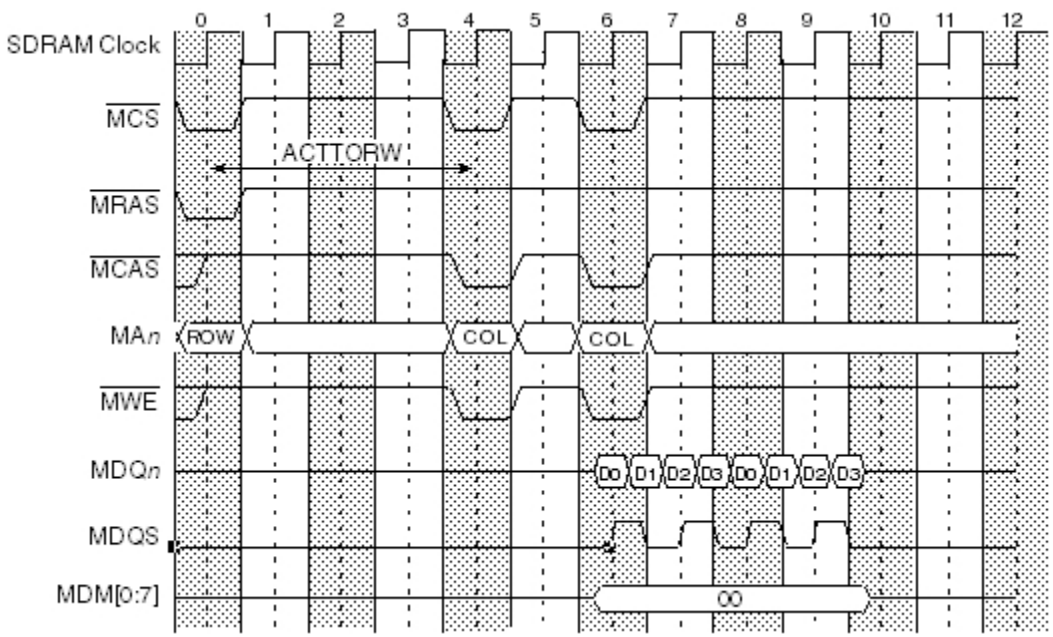


Figure 12-82. Registered DDR SDRAM DIMM Burst Write Timing

12.5.7 DDR SDRAM Write Timing Adjustments

The DDR memory controller facilitates system design flexibility by providing a write timing adjustment parameter, write data delay, (TIMING_CFG_2[WR_DATA_DELAY]) for data and DQS. The DDR SDRAM specification requires DQS be received no sooner than 75% of an SDRAM clock period- and no later than 125% of a clock period-from the capturing clock edge of the command/ address at the SDRAM. The WR_DATA_DELAY parameter may be used to meet this timing requirement for a variety of system configurations, ranging from a system with one DIMM to a fully populated system with two DIMMs.
TIMING_CFG_2[WR_DATA_DELAY] specifies how much to delay the launching of DQS and data from the first clock edge occurring one SDRAM clock cycle after the command is launched. The delay increment step sizes are in 1/4 SDRAM clock periods starting with the default value of 0.

The figure below shows the use of the WR_DATA_DELAY parameter.

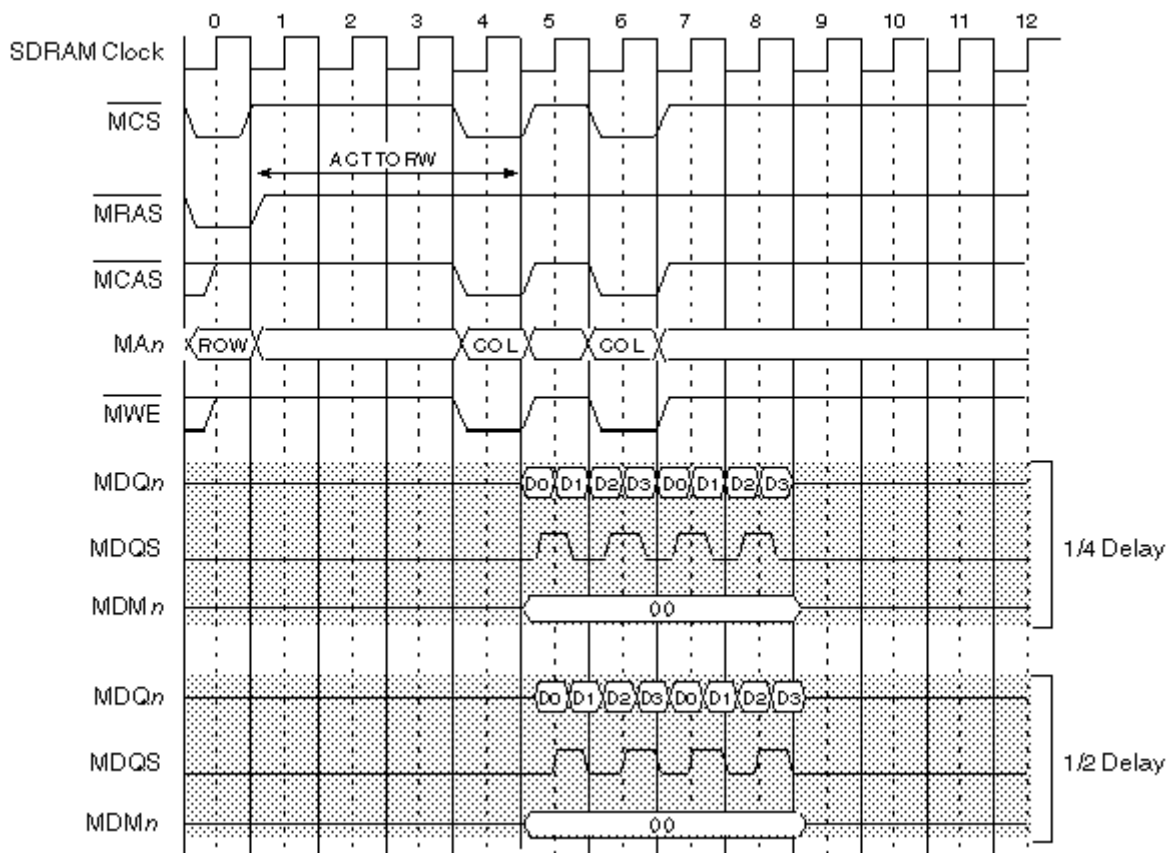


Figure 12-83. Write Timing Adjustments Example for Write Latency = 1

12.5.8 DDR SDRAM Refresh

The DDR memory controller supports auto-refresh and self-refresh. Auto refresh is used during normal operation and is controlled by the DDR_SDRAM_INTERVAL[REFINT] value; self-refresh is used only when the DDR memory controller is set to enter a sleep power management state. The REFINT value, which represents the number of memory bus clock cycles between refresh cycles, must allow for possible outstanding transactions to complete before a refresh request is sent to the memory after the REFINT value is reached. If a memory transaction is in progress when the refresh interval is reached, the refresh cycle waits for the transaction to complete. In the worst case, the refresh cycle must wait the number of bus clock cycles required by the longest programmed access. To ensure that the latency caused by a memory transaction does not violate the device refresh period, it is recommended that the programmed value of REFINT be less than that required by the SDRAM.

When a refresh cycle is required, the DDR memory controller does the following:

1. Completes all current memory requests.
2. Closes all open pages with a PRECHARGE-ALL command to each DDR SDRAM bank with an open page (as indicated by the row open table).
3. Issues one or more auto-refresh commands to each DDR SDRAM bank (as identified by its chip select) to refresh one row in each logical bank of the selected physical bank.

The auto-refresh commands are staggered across the four possible banks to reduce the system's instantaneous power requirements. Two sets of auto refresh commands will be issued on consecutive cycles when the memory is fully populated with two DIMMs. The initial PRECHARGE-ALL commands are also staggered in two groups for convenience. It is important to note that when entering self-refresh mode, only one refresh command is issued simultaneously to all physical banks. For this entire refresh sequence, no cycle optimization occurs for the usual case where fewer than four banks are installed. After the refresh sequence completes, any pending memory request is initiated after an inactive period specified by TIMING_CFG_1 [REFREC] and TIMING_CFG_3[EXT_REFREC]. In addition, posted refreshes are supported to allow the refresh interval to be set to a larger value.

12.5.8.1 DDR SDRAM Refresh Timing

Refresh timing for the DDR SDRAM is controlled by the programmable timing parameter TIMING_CFG_1 [REFREC], which specifies the number of memory bus clock cycles from the refresh command until a logical bank activate command is allowed. The DDR memory controller implements bank staggering for refreshes, as shown in Figure 12-84 (TIMING_CFG_1 [REFREC] = 10 in this example).

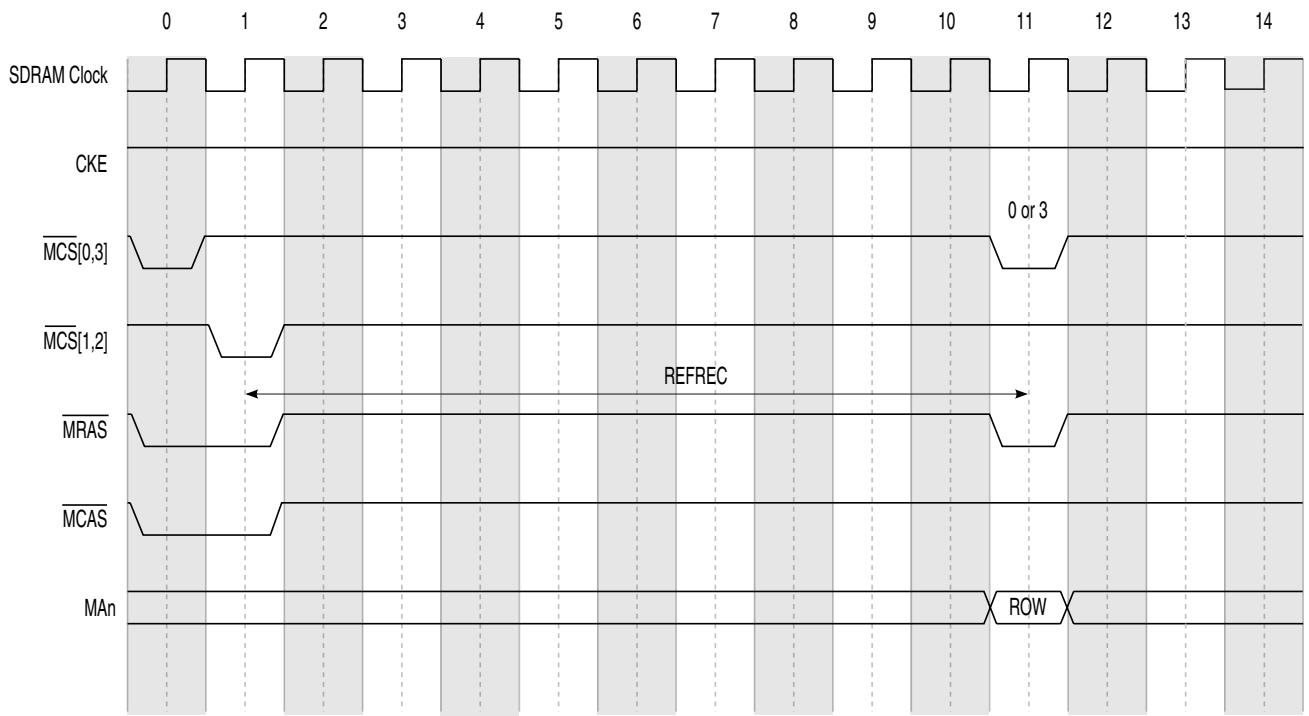


Figure 12-84. DDR SDRAM Bank Staggered Auto Refresh Timing

NOTE

Although the figure shows the controller activating bank 0 or 3 after the refresh recovery interval has expired in cycle 11, the controller could activate bank 1 or 2 instead of bank 0 or 3.

System software is responsible for optimal configuration of TIMING_CFG_1 [REFREC] and TIMING_CFG_3[EXT_REFREC] at reset. Configuration must be completed before DDR SDRAM accesses are attempted.

12.5.8.2 DDR SDRAM Refresh and Power-Saving Modes

In full-on mode, the DDR memory controller supplies the normal auto refresh to SDRAM. In sleep mode, the DDR memory controller can be configured to take advantage of self-refreshing SDRAMs or to provide no refresh support. Self-refresh support is enabled with the SREN memory control parameter.

Table 12-88 summarizes the refresh types available in each power-saving mode.

Table 12-88. DDR SDRAM Power-Saving Modes Refresh Configuration

Power Saving Mode	Refresh Type	SREN
Sleep	Self	1
	None	-

Note that in the absence of refresh support, system software must preserve DDR SDRAM data (such as by copying the data to disk) before entering the power-saving mode.

The dynamic power-saving mode uses the CKE DDR SDRAM pin to dynamically power down when there is no system memory activity. The CKE pin is negated when both of the following conditions are met:

- No memory refreshes are scheduled
- No memory accesses are scheduled

CKE is reasserted when a new access or refresh is scheduled or the dynamic power mode is disabled. This mode is controlled with DDR_SDRAM_CFG[DYN_PWR_MGMT].

Dynamic power management mode offers tight control of the memory system's power consumption by trading power for performance through the use of CKE. Powering up the DDR SDRAM when a new memory reference is scheduled causes an access latency penalty, depending on whether active or precharge powerdown is used, along with the settings of TIMING_CFG_0[ACT_PD_EXIT] and TIMING_CFG_0[PRE_PD_EXIT]. A penalty of 1 cycle is shown in the figure below .

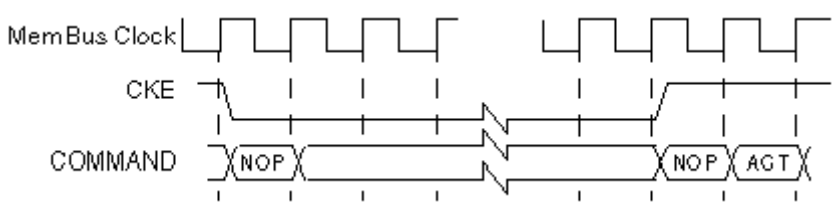


Figure 12-85. DDR SDRAM Power-Down Mode

12.5.8.2.1 Self-Refresh in Sleep Mode

The entry and exit timing for self-refreshing SDRAMs is shown in Figure 12-86 and Figure 12-87.

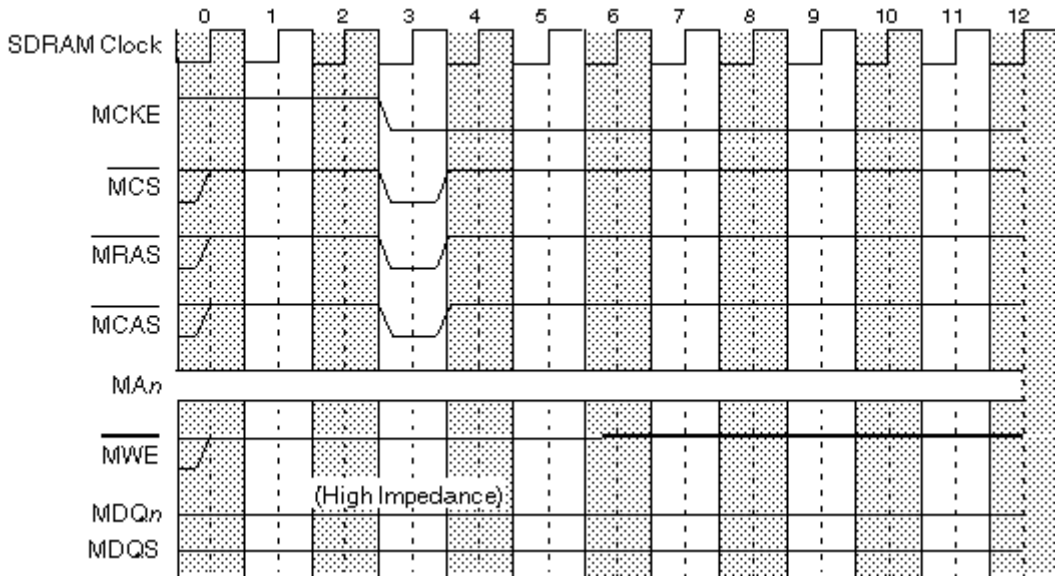


Figure 12-86. DDR SDRAM Self-Refresh Entry Timing

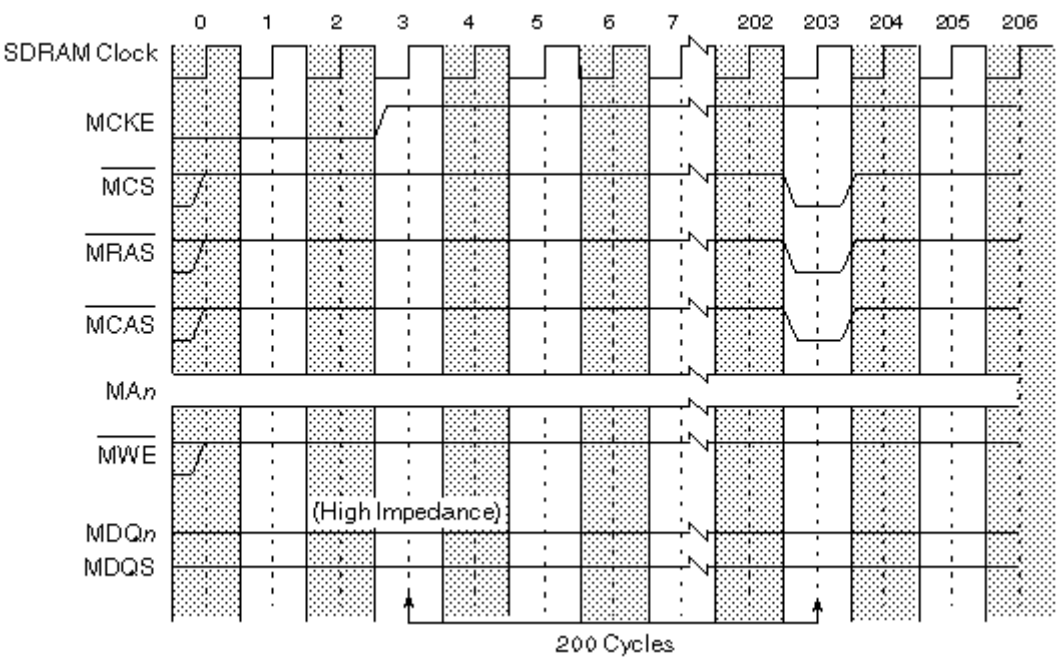


Figure 12-87. DDR SDRAM Self-Refresh Exit Timing

12.5.9 DDR Data Beat Ordering

Transfers to and from memory are always performed in four- or eight-beat bursts (four beats = 32 bytes when a 64-bit bus is used). For transfer sizes other than four or eight beats, the data transfers are still operated as four- or eight-beat bursts. If ECC is enabled and either the access is not doubleword aligned or the size is not a multiple of a doubleword, a full read-modify-write is performed for a write to SDRAM. If ECC is disabled or both the access is doubleword aligned with a size that is a multiple of a doubleword, the data masks (MDM[0:8] (MDM[0:4] for 32-bit bus) can be used to prevent the writing of unwanted data to SDRAM. The DDR memory controller also uses data masks to prevent all unintended full double words from writing to SDRAM. For example, if a write transaction is desired with a size of one double word (8 bytes), then the second, third, and fourth beats of data are not written to DRAM.

All writes for DDR3 mode will be aligned to beat 0 of the DRAM.

12.5.10 Page Mode and Logical Bank Retention

The DDR memory controller supports an open/closed page mode with an allowable open page for each logical bank of DRAM used. In closed page mode for DDR SDRAMs, the DDR memory controller uses the SDRAM auto-precharge feature, which allows the controller to indicate that the page must be automatically closed by the DDR SDRAM after the READ or WRITE access. This is performed by using MA[10] of the address during the COMMAND phase of the access to enable auto-precharge. Auto-precharge is non-persistent in that it is either enabled or disabled for each individual READ or WRITE command. It can, however, be enabled or disabled separately for each chip select.

When the DDR memory controller operates in open page mode, it retains the currently active SDRAM page by not issuing a precharge command. The page remains opens until one of the following conditions occurs:

- Refresh interval is met.
- The user-programmable DDR_SDRAM_INTERVAL[BSTOPRE] value is exceeded.
- There is a logical bank row collision with another transaction that must be issued.

Page mode can dramatically reduce access latencies for page hits. Depending on the memory system design and timing parameters, using page mode can save two to three clock cycles for subsequent burst accesses that hit in an active page. Also, better performance can be obtained by using more banks, especially in systems which use many different channels. Page mode is disabled by clearing DDR_SDRAM_INTERVAL[BSTOPRE] or setting CS n_CONFIG[AP_ nEN].

12.5.11 Error Checking and Correcting (ECC)

The DDR memory controller supports error checking and correcting (ECC) for the data path between the core master and system memory. The memory detects all double-bit errors, detects all multi-bit errors within a nibble, and corrects all single-bit errors. Other errors may be detected, but are not guaranteed to be corrected or detected. Double-bit errors are always reported when error reporting is enabled. When a single-bit error occurs, the single-bit error counter register is incremented, and its value compared to the single-bit error trigger register. An error is reported when these values are equal. The single-bit error registers can be programmed such that minor memory faults are corrected and ignored, but a catastrophic memory failure generates an interrupt.

For writes that are smaller than 64 bits, the DDR memory controller performs a double-word read from system memory of the address for the write (checking for errors), and merges the write data with the data read from memory. Then, a new ECC code is generated for the merged double word. The data and ECC code is then written to memory. If a multi-bit error is detected on the read, the transaction completes the read-modify-write to keep the DDR memory controller from hanging. However, the corrupt data is masked on the write, so the original contents in SDRAM remain unchanged.

The syndrome encodings for the ECC code are shown in Table 12-89 and Table 12-90.

Table 12-89. DDR SDRAM ECC Syndrome Encoding

Data Bit	Syndrome Bit									Data Bit	Syndrome Bit							
	0	1	2	3	4	5	6	7			0	1	2	3	4	5	6	7
0	•	•						•		32			•	•				•
1	•		•					•		33			•		•			•
2	•			•				•		34	•		•		•			
3	•				•			•		35		•	•		•			
4	•	•				•				36			•	•		•		
5	•		•			•				37			•		•	•		
6	•			•		•				38	•		•		•	•		•
7	•				•	•				39		•	•		•	•		•
8	•	•					•			40			•	•			•	
9	•		•				•			41			•		•		•	
10	•			•			•			42	•		•		•		•	•
11	•				•		•			43		•	•		•		•	•
12	•	•				•	•	•		44			•	•		•	•	•
13	•		•			•	•	•		45			•		•	•	•	•
14	•			•		•	•	•		46	•		•		•	•	•	
15	•				•	•	•	•		47		•	•		•	•	•	

Table continues on the next page...

Table 12-89. DDR SDRAM ECC Syndrome Encoding (continued)

Data Bit	Syndrome Bit								Data Bit	Syndrome Bit							
	0	1	2	3	4	5	6	7		0	1	2	3	4	5	6	7
16		•	•					•	48		•				•	•	
17		•		•				•	49			•			•	•	
18		•			•			•	50				•		•	•	
19	•	•			•				51	•					•	•	
20		•	•			•			52		•				•		•
21		•		•		•			53			•			•		•
22		•			•	•			54				•		•		•
23	•	•			•	•		•	55	•					•		•
24		•	•				•		56		•					•	•
25		•		•			•		57			•				•	•
26		•			•		•		58				•			•	•
27	•	•			•		•	•	59	•						•	•
28		•	•			•	•	•	60				•	•		•	
29		•		•		•	•	•	61	•			•	•		•	•
30		•			•	•	•	•	62		•		•	•		•	•
31	•	•			•	•	•		63			•	•	•		•	•

Table 12-90. DDR SDRAM ECC Syndrome Encoding (Check Bits)

Check Bit	Syndrome Bit							
	0	1	2	3	4	5	6	7
0	•							
1		•						
2			•					
3				•				
4					•			
5						•		
6							•	
7								•

12.5.12 Error Management

The DDR memory controller detects four different kinds of errors: training, single-bit, multi-bit, and memory select errors. The following discussion assumes all the relevant error detection, correction, and reporting functions are enabled as described in Memory error interrupt enable (DDR_ERR_INT_EN)Memory error disable (DDR_ERR_DISABLE) and Memory error detect (DDR_ERR_DETECT)."

Single-bit errors are counted and reported based on the ERR_SBE value. When a single-bit error is detected, the DDR memory controller does the following:

- Corrects the data
- Increments the single-bit error counter ERR_SBE[SBEC]
- Generates an critical interrupt if the counter value ERR_SBE[SBEC] equals the programmable threshold ERR_SBE[SBET]
- Completes the transaction normally

If a multi-bit error is detected for a read, the DDR memory controller logs the error and generates an error interrupt (if enabled, as described in Memory error disable (DDR_ERR_DISABLE)"). Another error the DDR memory controller detects is a memory select error, which causes the DDR memory controller to log the error and generate an critical interrupt (if enabled, as described in Memory error detect (DDR_ERR_DETECT)"). This error is detected if the address from the memory request does not fall into any of the enabled, programmed chip select address ranges. For all memory select errors, the DDR memory controller does not issue any transactions onto the pins after the first read has returned data strobes. If the DDR memory controller is not using sample points, then a dummy transaction is issued to DDR SDRAM with the first enabled chip select. In this case, the source port on the pins is forced to 0x1F to show the transaction is not real. Table 12-91 shows the errors with their descriptions. The final error the memory controller detects is the automatic calibration error. This error is set if the memory controller detects an error during its training sequence.

Table 12-91. Memory Controller Errors

Category	Error	Descriptions	Action	Detect Register
Notification	Single-bit ECC threshold	The number of ECC errors has reached the threshold specified in the ERR_SBE.	The error is reported via machine check or an error interrupt if enabled.	The error control register only logs read versus write, not full type
Access Error	Multi-bit ECC error	A multi-bit ECC error is detected during a read, or read-modify-write memory operation.		
	Memory select error	Read, or write, address does not fall within the address range of any of the memory banks.		

12.5.13 DDR Rapid Clear of Memory

Upon Hard Fail condition, the security monitor drives an interrupt to the DDR controller indicating the memory needs to be cleared, the DDR controller will clear all of memory defined by the CS*n*_BNDS registers for all enabled chip selects. DDRDSR_2[RPD_ST] will be set once the DDR controller begins clearing memory. At this time, writes to the DDR CCSR space are not allowed to modify the register values. Once the rapid clear of memory sequence is complete, writes to the DDR registers may proceed. The DDRDSR_2[RPD_EN] bit is provided so software can determine when the rapid clear of memory sequence is complete and can therefore update registers again. Software can clear the RPD_ST and RPD_EN bits after the rapid memory clear is complete.

12.6 Initialization/Application Information

NOTE

The DDRC must be programmed before the corresponding Local Access Windows are programmed and enabled for the DDR target(s).

System software must configure the DDR memory controller, using a memory polling algorithm at system start-up, to correctly map the size of each bank in memory. Then, the DDR memory controller uses its bank map to assert the appropriate MCSn_B signal for memory accesses according to the provided bank depths. System software must also configure the DDR memory controller at system start-up to appropriately multiplex the row and column address bits for each bank. Refer to row-address configuration in Chip select n configuration (DDR_CS*n*_CONFIG). Address multiplexing occurs according to these configuration bits.

At system reset, initialization software (boot code) must set up the programmable parameters in the memory interface configuration registers. See DDR Memory Map/ Register Definition for more detailed descriptions of the configuration registers. These parameters are shown in the table below.

Table 12-92. Memory Interface Configuration Register Initialization Parameters

Name	Description	Parameter	Section/Page
CS n_BNDS	Chip select memory bounds	SA n EA n	Chip select n memory bounds (DDR_CS*n*_BNDS)

Table continues on the next page...

Table 12-92. Memory Interface Configuration Register Initialization Parameters (continued)

Name	Description	Parameter		Section/Page
CS n_CONFIG	Chip select configuration	CS_ n_EN AP_ n_EN ODT_RD_CFG ODT_WR_CFG	BA_BITS_CS_ n ROW_BITS_CS_ n COL_BITS_CS_ n	Chip select n configuration (DDR_CS*n*_CONFIG)
CS*n*_CONFIG_2	Chip select configuration 2	PASR_CFG		Chip select n configuration 2 (DDR_CS*n*_CONFIG_2)
TIMING_CFG_3	Extended timing parameters for fields in TIMING_CFG_1	EXT_PRETOACT EXT_ACTTOPRE EXT_ACTTORW EXT_REFREC EXT_CASLAT EXT_WRREC CNTL_ADJ		DDR SDRAM timing configuration 3 (DDR_TIMING_CFG_3)
TIMING_CFG_0	Timing configuration	RWT WRT RRT WWT	ACT_PD_EXIT PRE_PD_EXIT MRS_CYC	DDR SDRAM timing configuration 0 (DDR_TIMING_CFG_0)
TIMING_CFG_1	Timing configuration	PRETOACT ACTTOPRE ACTTORW CASLAT	REFREC WRREC ACTTOACT WRTORD	DDR SDRAM timing configuration 1 (DDR_TIMING_CFG_1)
TIMING_CFG_2	Timing configuration	ADD_LAT CPO WR_LAT RD_TO_PRE	WR_DATA_DELAY CKE_PLS FOUR_ACT	DDR SDRAM timing configuration 2 (DDR_TIMING_CFG_2)
DDR_SDRAM_CFG	Control configuration	SREN ECC_EN RD_EN SDRAM_TYPE DYN_PWR 32_BE 8_BE DBW	2T_EN 3T_EN BA_INTLV_CTL HSE BI	DDR SDRAM control configuration (DDR_DDR_SDRAM_CFG)
DDR_SDRAM_CFG_2	Control configuration	SR_IE ODT_CFG	NUM_PR OBC_CFG AP_EN D_INIT	DDR SDRAM control configuration 2 (DDR_DDR_SDRAM_CFG_2)

Table continues on the next page...

Table 12-92. Memory Interface Configuration Register Initialization Parameters (continued)

Name	Description	Parameter	Section/Page
		RCW_EN MD_EN	
DDR_SDRAM_MODE	Mode configuration	ESDMODE SDMODE	DDR SDRAM mode configuration (DDR_DDR_SDRAM_MODE)
DDR_SDRAM_MODE_2	Mode configuration	ESDMODE2 ESDMODE3	DDR SDRAM mode configuration 2 (DDR_DDR_SDRAM_MODE_2)
DDR_SDRAM_INTERVAL	Interval configuration	REFINT BSTOPRE	DDR SDRAM interval configuration (DDR_DDR_SDRAM_INTERVAL)
DDR_DATA_INIT	Data initialization configuration register	INIT_VALUE	DDR SDRAM data initialization (DDR_DDR_DATA_INIT)
DDR_SDRAM_CLK_CNTL	Clock adjust	CLK_ADJUST	DDR SDRAM clock control (DDR_DDR_SDRAM_CLK_CNTL)
DDR_INIT_ADDR	Initialization address	INIT_ADDR	DDR training initialization address (DDR_DDR_INIT_ADDR)
DDR_INIT_EXT_ADDR	Extended initialization address	INIT_EXT_ADDR	DDR training initialization extended address (DDR_DDR_INIT_EXT_ADDRESS)
TIMING_CFG_4	Timing configuration	RWT WRT RRT WWT EXT_RWT EXT_WRT EXT_RRT EXT_WWT EXT_REFINT DLL_LOCK	DDR SDRAM timing configuration 4 (DDR_TIMING_CFG_4)

Table continues on the next page...

Table 12-92. Memory Interface Configuration Register Initialization Parameters (continued)

Name	Description	Parameter	Section/Page
TIMING_CFG_5	Timing configuration	RODT_ON RODT_OFF WODT_ON WODT_OFF	DDR SDRAM timing configuration 5 (DDR_TIMING_CFG_5)
DDR_ZQ_CNTL	ZQ calibration control	ZQ_EN ZQINIT ZQOPER ZQCS	DDR ZQ calibration control (DDR_DDR_ZQ_CNTL)
DDR_WRLVL_CNTL	Write leveling control	WRLVL_EN WRLVL_MRD WRLVL_ODTEN WRLVL_DQSEN WRLVL_SMPL WRLVL_WLR WRLVL_START	DDR write leveling control (DDR_DDR_WRLVL_CNTL)
DDR_WRLVL_CNTL_2	Write leveling control	WRLVL_START_1 WRLVL_START_2 WRLVL_START_3 WRLVL_START_4	DDR write leveling control 2 (DDR_DDR_WRLVL_CNTL_2)
DDR_WRLVL_CNTL_3	Write leveling control	WRLVL_START_5 WRLVL_START_6 WRLVL_START_7 WRLVL_START_8	DDR write leveling control 3 (DDR_DDR_WRLVL_CNTL_3)
DDR_SR_CNTR	Self refresh control	SR_IT	DDR Self Refresh Counter (DDR_DDR_SR_CNTR)
DDR_SDRAM_RCW_1	Register control words configuration	RCW0 RCW1 RCW2 RCW3 RCW4 RCW5 RCW6 RCW7	DDR Register Control Words 1 (DDR_DDR_SDRAM_RCW_1)
DDR_SDRAM_RCW_2	Register control words configuration	RCW8 RCW9 RCW10 RCW11	DDR Register Control Words 2 (DDR_DDR_SDRAM_RCW_2)

Table continues on the next page...

Table 12-92. Memory Interface Configuration Register Initialization Parameters (continued)

Name	Description	Parameter		Section/Page
		RCW12 RCW13 RCW14 RCW15		
DDRCDR_1	Driver control	DHC_EN ODT DSO_C_EN DSO_D_EN	DSO_CPZ DSO_CNZ DSO_DPZ DSO_DNZ	DDR Control Driver Register 1 (DDR_DDRCDR_1)
DDRCDR_2	Driver control	DSO_CLK_EN DSO_CLKPZ DSO_CLKNZ		DDR Control Driver Register 2 (DDR_DDRCDR_2)

12.6.1 Programming Summary

Depending on the memory type used, certain fields must be programmed differently. The table below illustrates the differences in certain fields for different memory types. Note: This table does not list all fields that must be programmed.

Table 12-93. DDR3 Programming Summary

Parameter	Description	Summary	Section
AP*n* _EN	Chip Select *n*Auto Precharge Enable	Can be used to place chip select *n*in auto precharge mode	
ODT_RD_CFG	Chip Select ODT Read Configuration	Can be enabled to assert ODT if desired. This could be set differently depending on system topology. However, systems with only 1 chip select will typically not use ODT when issuing reads to the memory.	
ODT_WR_CFG	Chip Select ODT Write Configuration	Can be enabled to assert ODT if desired. This could be set differently depending on system topology. However, ODT will typically be set to assert for the chip select that is getting written to (value would be set to 001).	
ODT_PD_EXIT	ODT Powerdown Exit	Should be set to 0001 for DDR3. The powerdown times (t_{XP} and t_{XPDLL}) required for DDR3 are controlled via TIMING_CFG_0[ACT_PD_EXIT] and TIMING_CFG_0[PRE_PD_EXIT].	
PRETOACT	Precharge to Activate Timing	Should be set according to the specifications for the memory used (t_{RP})	
ACTTOPRE	Activate to Precharge Timing	Should be set, along with the Extended Activate to Precharge Timing, according to the specifications for the memory used (t_{RAS})	
ACTTORW	Activate to Read/Write Timing	Should be set according to the specifications for the memory used (t_{RCD})	

Table continues on the next page...

Table 12-93. DDR3 Programming Summary (continued)

Parameter	Description	Summary	Section
CASLAT	CAS Latency	Should be set, along with the Extended CAS Latency, to the desired CAS latency	
REFREC	Refresh Recovery	Should be set, along with the Extended Refresh Recovery, to the specifications for the memory used (T_{RFC})	
WRREC	Write Recovery	Should be set according to the specifications for the memory used (t_{WR}). If DDR_SDRAM_CFG_2[OBC_CFG] is set, then this should be programmed to t_{WR} + 2 DRAM cycles.	
ACTTOACT	Activate A to Activate B	Should be set according to the specifications for the memory used (t_{RRD})	
WRTORD	Write to Read Timing	Should be set according to the specifications for the memory used (t_{WTR}) If DDR_SDRAM_CFG_2[OBC_CFG] is set, then this should be programmed to t_{WTR} + 2 DRAM cycles.	
ADD_LAT	Additive Latency	Should be set to the desired additive latency. This must be set to a value less than TIMING_CFG_1[ACTTORW]	
WR_LAT	Write Latency	Should be set to the desired write latency. Note that DDR3 SDRAMs do not necessarily require the write latency to equal the CAS latency minus 1 cycle. The minimum WR_LAT that can be used in 1T timing mode is 5 cycles if DDR_RATE=0	
RD_TO_PRE	Read to Precharge Timing	Should be set according to the specifications for the memory used (t_{RTP}). Time between read and precharge for non-zero value of additive latency (AL) is a minimum of AL + t_{RTP} cycles. If DDR_SDRAM_CFG_2[OBC_CFG] is set, then this should be programmed to t_{RTP} + 2 DRAM cycles.	
CKE_PLS	Minimum CKE Pulse Width	Should be set according to the specifications for the memory used (t_{CKE})	
FOUR_ACT	Four Activate Window	Should be set according to the specifications for the memory used (t_{FAW}).	
RD_EN	Registered DIMM Enable	If registered DIMMs are used, then this field should be set to 1	
8_BE	8-beat burst enable	If a 64-bit bus is used, this should be set to 0. Otherwise, this should be set to 1. If this is set to 0, then other requirements in TIMING_CFG_4 will be needed to ensure t_{CCD} is met.	
2T_EN	2T Timing Enable	In heavily loaded systems, this can be set to 1 to gain extra timing margin on the interface at the cost of address/ command bandwidth.	
ODT_CFG	ODT Configuration	Can be set for termination at the IOs according to system topology. Typically, if ODT is enabled, then the internal IOs should be set up for termination only during reads to DRAM.	
OBC_CFG	On-The-Fly Burst Chop Configuration	Can be set to 1 if on-the-fly burst chop will be used. This is expected to give the best performance in DDR3 mode. This feature can only be used if a 64-bit data bus is used.	
RWT	Read-to-write turnaround for same chip select (in TIMING_CFG_4)	This can be used to force a longer read-to-write turnaround time when accessing the same chip select. This is useful for burst chop mode, as there are some timing requirements to the same chip select that still must be met.	

Table continues on the next page...

Table 12-93. DDR3 Programming Summary (continued)

Parameter	Description	Summary	Section
WRT	Write-to-read turnaround for same chip select (in TIMING_CFG_4)	This could be used to force a certain turnaround time between a write and read to the same chip select. This is useful for burst chop mode. However, it is expected that TIMING_CFG_1[WRTORD] will be programmed appropriately such that TIMING_CFG_4[WRT] can be set to 0000.	
RRT	Read-to-read turnaround for same chip select (in TIMING_CFG_4)	Should typically be set to 0100 in burst chop mode (on-the-fly or fixed).	
WWT	Write-to-write turnaround for same chip select (in TIMING_CFG_4)	Should typically be set to 0100 in burst chop mode (on-the-fly or fixed).	
ZQ_EN	ZQ Calibration Enable	Should be set to 1. The other fields in DDR_ZQ_CNTL should also be programmed appropriately based on the DRAM specifications.	
WRLVL_EN	Write Leveling Enable	Can be set to 1 if write leveling is desired. Otherwise the value used in TIMING_CFG_2[WR_DATA_DELAY] will be used to shift all bytes during writes to DRAM. If write leveling will be used, all other fields in DDR_WRLVL_CNTL should be programmed appropriately based on the DRAM specifications.	
BSTOPRE	Burst To Precharge Interval	Can be set to any value, depending on the application. Auto precharge can be enabled by setting this field to all 0s.	

12.6.2 DDR SDRAM Initialization Sequence

After configuration of all parameters is complete, system software must set DDR_SDRAM_CFG[MEM_EN] to enable the memory interface. Note that 200 µs (500 µs for DDR3) must elapse after DRAM clocks are stable (DDR_SDRAM_CLK_CNTL[CLK_ADJUST] is set and any chip select is enabled) before MEM_EN can be set, so a delay loop in the initialization code may be necessary if software is enabling the memory controller. If DDR_SDRAM_CFG[BI] is not set, the DDR memory controller will conduct an automatic initialization sequence to the memory, which will follow the memory specifications. If the bypass initialization mode is used, then software can initialize the memory through the DDR_SDRAM_MD_CNTL register.

12.6.3 Using Forced Self-Refresh Mode to Implement a Battery-Backed RAM System

This section describes the options offered by this device to support battery-backed main memory.

12.6.3.1 Hardware Based Self-Refresh Scheme

An external voltage sense device can be connected to this device through one of the external interrupt lines IRQ*n*. The external interrupt from the voltage sensor would then be steered through the programmable interrupt controller (MPIC) to the internal SoC interrupt event signal, *sie0*. Note that the *sie0* signal must remain high until power is removed.

If DDR_SDRAM_CFG_2[SR_IE] is set, the *sie0* signal from the interrupt controller is then automatically detected by the DDR controller, which immediately causes main memory to enter self-refresh mode. See DDR SDRAM control configuration 2 (DDR_DDR_SDRAM_CFG_2) for further information on this bit.

These fields in the appropriate registers in the MPIC must be set for self refresh to function:

- EIVPR*n*[PRIORITY] should be set to 0xF (highest priority)
- EILRn[INTTGT] should be set to route the incoming signal to *sie0*

See EIVPR and EILR for descriptions of these registers.

12.6.3.2 Software Based Self-Refresh Scheme

The DDR controller also has a software-programmable bit, DDR_SDRAM_CFG_2[FRC_SR], that immediately puts main memory into self-refresh mode. See DDR SDRAM control configuration 2 (DDR_DDR_SDRAM_CFG_2) for a description of this register.

It is expected that a critical interrupt routine triggered by an external voltage sensing device will have time to set this bit.

12.6.3.3 Bypassing Re-initialization During Battery-Backed Operation

The DDR controller offers an initialization bypass feature (DDR_SDRAM_CFG[BI]), which system designers may use to prevent re-initialization of main memory during system power-on following an abnormal shutdown. See DDR SDRAM control configuration (DDR_DDR_SDRAM_CFG) for information on this bit and DDR training initialization address (DDR_DDR_INIT_ADDR) for a discussion of avoiding possible ECC errors in this mode.

Note that the DDR controller will automatically wait 200 DRAM cycles before issuing any command after the assertion of MCKE[0:3] when this mode is used.

Chapter 13
Integrated Flash Controller (IFC)

13.1 IFC overview

The integrated flash controller (IFC) is used to interface with external asynchronous NAND flash, asynchronous NOR flash, SRAM, generic ASIC memories, and EPROM.

It has eight chip-selects to which a maximum of eight flash devices can be attached, although only one can be accessed at any given time.

The IFC handles pin multiplexing to the internal system bus based on the selected controller (NAND, NOR, GPCM, or generic ASIC). To save pins at the chip level, multiplexing of address pins can be done on the data bus using an external address valid signal (AVD/ALE). The BCH error-correction algorithm is used to correct the error bits while reading from a NAND device.

This figure shows a block diagram of the IFC.

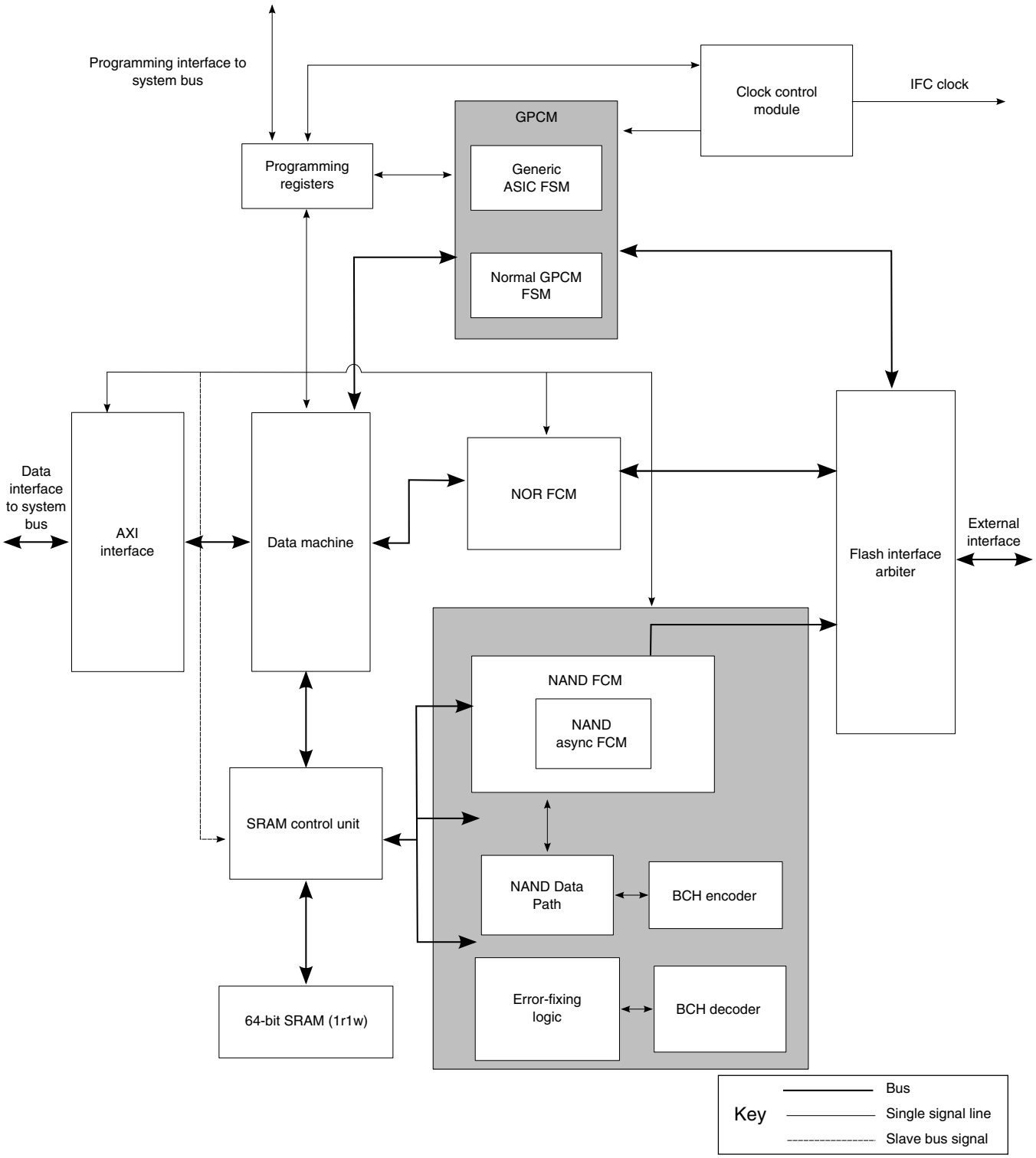


Figure 13-1. IFC block diagram

13.1.1 IFC features summary

The IFC supports both general and controller-specific features.

The general features of the IFC include the following:

- Flash controller with eight chip-selects
- Supports error and debug registers
- Functional muxing of pins between NAND, NOR, and GPCM
- Supports memory banks of sizes up to 4 GB
- Write-protection capability (only for NAND and NOR)
- Provision of software reset
- External transceiver enable/disable control on a per-bank basis

13.1.1.1 NAND flash features

The IFC features a NAND flash controller.

- x8/x16 NAND flash interface
- Support for ONFI-2.2 asynchronous interface (8-/16-bit) and mandatory commands
- BCH code for 4-bit and 8-bit error correction per sector of 512 bytes (using GF-2^{13} Galois field) and 24 and 40-bit ECC per sector of 1 KB (using GF- 2^{14} Galois field):
  - For a 512-byte page and 16-byte spare region, 4-bit error correction is supported
  - For a 2 KB page and 64-byte spare region, 4-bit error correction is supported
  - For 4 KB page width:
    - For 128-byte spare region, 4-bit error correction is supported
    - For 210-, 218-, and 224-byte spare regions, 4- or 8-bit error correction is supported
    - Spare size of a minimum of 176 bytes (8 for BBI + 4x42 for ECC bytes), 4-, 8-, or 24-bit BCH
    - Spare size of a minimum of 288 bytes (8 for BBI + 4x70 for ECC bytes), 4-, 8-, 24-, or 40-bit BCH
  - For an 8 KB page width:
    - At least 136(8 + 128) bytes of spare region is required for bad block information and ECC bytes, 4-bit BCH
    - 8 KB page, at least 264(8 + 256) bytes of spare region required for bad block information and ECC bytes, 4-bit or 8-bit BCH
    - Spare size of a minimum of 344 bytes (8 for BBI + 8x42 for ECC bytes), 4-, 8-, or 24-bit BCH
    - Spare size of a minimum of 568 bytes (8 for BBI + 8x70 for ECC bytes), 4-, 8-, 24-, or 40-bit BCH
  - For all ECC modes (4/8/24/40), parity bytes are stored at offset 08h in a spare region
- ECC generation/checking is optional
- Flexible timing control to allow interfacing with proprietary NAND devices

- Supports SLC and MLC flash devices with configurable page sizes of up to 8 KB
- Supports advance NAND commands such as cache, copy-back, and multi-plane programming
- Supports two >RDY_B/BSY_B signals
- Programmable command and data transfer sequences of up to 15 steps
- Configurable block-size constraints to a multiple of 32 pages, up to 256 pages. Multiplying factors are taken as power of 2.
- Interrupt for error handling and flash command completion event
- Internal SRAM of 9 KB
  - Memory-mapped
  - Can be configured as boot RAM
  - Acts as data buffer for normal operation
- Boot chip-select (CS0) available after system reset, with a boot block size of 8 KB for execute-in-place boot loading from NAND flash (in asynchronous mode)
- Supports flash devices of a magnitude of terabytes

13.1.1.2 NOR flash features

The IFC features a NOR flash controller.

- Compatible with asynchronous NOR flash (synchronous burst-read not supported)
- Provides memory-mapped interfacing to NOR
- Supports an address data multiplexed (ADM) NOR device
- Flexible timing control allows interfacing with proprietary NOR devices (WE_B controlled writes only)
- Boot chip-select (CS0) is available at system reset
- Data bus width of 8/16

13.1.1.3 GPCM and GASIC features

The IFC features a GPCM. GPCM features comprise of both normal GPCM and generic ASIC features.

Normal GPCM features include the following:

- Supports x8-/16-bit devices
- Compatible with general-purpose addressable device,such as SRAM and ROM
- External clock is supported with programmable division ratio (2, 3, 4, ... up to 16)
- Output enable signal (OE_B)
- Write enable signal (WE_B)
- Even/Odd parity on data bus supported

- External access termination signal (IFCTA_B)
- Burst support in GPCM

Generic ASIC features include the following:

- Supports x8-/16-bit devices
- The following address and data sequences are supported on the I/O bus:
  - 16-bit I/O: AADD
  - 8-bit I/O: AAAADDDD
- Configurable even/odd parity on address/data bus
- Parity-error detection

13.1.2 IFC modes of operation

The IFC contains one NAND, one NOR flash controller, and one GPCM/generic-ASIC controller.

It can be programmed such that all eight memory banks can work with the NAND, NOR, and GPCM/generic-ASIC depending on the requirement. Only one memory bank can be active at any given time.

13.2 External signal descriptions

This section provides both general and detailed information on the chip's external signals.

This table provides an overview of the chip's external signals, and the following table provides a much more detailed description of the signals.

Table 13-1. Signal properties

Chip signal name	IFC signal name	Description / Function	Number of signals	I/O
IFC_AD[0:15]	AD[0:15]	Multiplexed address/data bus. Bidirectional Data Bus for NOR, during AVD assertion this bus will carry address bits. Address LSB or MSB will be governed by CSOR[ADM_SHFT_MODE] Bidirectional Data Bus for GPCM, for external address latching this bus is used to carry address MSB's. Bidirectional shared address/data bus for GASIC.	16	I/O
IFC_A[16:31]	ADDR[16:31]	Dedicated address bus	16	IO

Table continues on the next page...

Table 13-1. Signal properties (continued)

Chip signal name	IFC signal name	Description / Function	Number of signals	I/O
IFC_PAR[0:1]	IFC_PAR[0:1]	Parity address and data (GPCM mode) Parity data (GASIC mode)	2	I/O
IFC_CS[0:7] _B	CE[0:7]_B	Chip-select	8	O
IFC_WE[0]_B	IFC_WE[0]_B / GPWE[0] / SOF_L	NAND Write Enable NOR Write Enable Start of frame indication for Generic ASIC		O
IFC_BCTL	BCTL	Data buffer control	1	O
IFC_TE	TE	External Transceiver Enable/Disable Control	1	I/O
IFC_AVD	AVD / IFC_WE[2]_B	External Address Valid Signal for NOR GPCM Write Byte Select 2 (IFC_WE2_B)	1	O
IFC_CLE	CLE / IFC_WE1_B	NAND Command Latch Enable GPCM Write Byte Select 1 (IFC_WE1_B)	1	O
IFC_OE_B	OE_B GPOE_B RW_L_B NDRE_B	NOR Output Enable GPCM Output Enable GASIC Read/Write Indication NAND Read Enable	1	O
IFC_WP[0]_B	WP[0]_B / NDWP_B	Per chip select based NAND Write Protect	4	O
IFC_RB[0:1]_B	RB[0:4]_B / IFCTA[0:7] / RDY_L[0:7]	Ready/busy for CS[0:7]_B NAND Ready/Busy input, NOR Ready Busy Input, GPCM External Termination, GASIC Ready Indication.	5	I
IFC_PERR_B	PERR_B	Parity error	1	I
IFC_CLK[0:1]	CLK[0:1]	External IFC clock	2	O

This table describes the external signals in detail.

Table 13-2. Detailed signal description

Signal	I/O	Description	
IFC_AD[0:*n*]	I/O	Multiplexed address/data bus.	
		State meaning	Asserted/Negated - During the assertion of AVD, AD[0:*n*] are driven with the address for the access to follow. External logic should propagate the address on AD[0:*n*] while AVD is asserted, and latch the address upon negation of AVD. After AVD is negated, AD[0:*n*] are either driven by write data or are switched to an input to sample read data driven by an external device. Following the last data transfer of a write access, AD[0:*n*] are released to a high-impedance state.
IFC_A[*n*:*m*]	O	Non-multiplexed address bus.	
		State meaning	Asserted/Negated - These signals provide the non-multiplexed lower portion of the address bus. In multiplex mode, the latched AD signals provide the most significant address bits and the A signals provide the least significant address bits.
IFC_AVD	O	Address valid signal for the address latch in NOR devices.	

Table continues on the next page...

Table 13-2. Detailed signal description (continued)

Signal	I/O	Description	
		State meaning	External logic should propagate the address on AD while AVD is asserted, and latch the address upon the negation of AVD.
IFC_CS[0:*n*]_B	O	Chip-Select; mutually exclusive chip-selects are available.	
		State meaning	Asserted/Negated - Used to enable specific memory device connected to the IFC. CS_B[0] corresponds to the chip-select for memory bank 0, which has the memory type and attributes defined by BR0 and FTIM0_CS0.
IFC_WE[0]_B	O	NAND write enable, NOR write enable, GPCM write byte select 0, GASIC start of frame.	
		State meaning	Asserted/Negated - Used to control the data write cycles on NOR/NAND flash. For GPCM it validates its corresponding byte number on the data bus. For GASIC it indicates start of a transaction.
IFC_CLE	O	NAND Command latch enable, GPCM write byte select 1 which is IFC_WE[1]_B.	
		State meaning	Asserted/Negated - It enables the command cycle on NAND flash. For GPCM it validates 1st byte on data bus.
IFC_OE_B	O	NAND read enable, NOR output enable, GPCM output enable, GASIC Read/Write Enable and is valid only when SOF_L_B is asserted.	
		State meaning	Asserted/Negated - It enables data read cycles on NOR, NAND, and GPCM devices. For GASIC, it indicates whether the transaction is read or write. High value on this indicates read operation while low value indicates write.
IFC_WP_B	O	Per chip select based non muxed NAND Write Protect Output, GPCM write byte select its corresponding byte number on the data bus.	
		State meaning	Asserted/Negated - It protects NAND flash from accidental erasure or program when asserted low. For GPCM it validates the corresponding byte number on data bus.
IFC_RB_B	I	NAND ready busy for all the chip-selects, NOR Read Busy, GPCM External Termination of Access, GASIC Ready Indication. Valid after the completion of address phase till it asserts.	
		State meaning	Asserted/Negated - It indicated the ready busy status of flash operation. It is used to stall the controller operation when flash is indicating busy. For GPCM this input is used for termination of current access. For GASIC it indicates that the current GASIC transaction is accepted by device and the host can start the next transaction.
IFC_BCTL	O	Data buffer control. Buffer control is disabled by FTIM0_CS0[BCTLD] field.	
		State meaning	Asserted/Negated - The BCTL pin normally functions as a write/read control for a device (NOR) connected to the AD signals. Note that an external data buffer must not drive the AD lines in conflict with the IFC when BCTL is high, because BCTL remains high after reset and during address phases.
IFC_TE	I/O	External transceiver enable/disable. This pin acts as an input during reset and should be weakly pulled up/down so as to disable the external transceiver during that time. After reset, the IFC drives TE with the value configures in CSPRn[TE] as per the selected chip select. When none of the CS pins are selected, TE is driven High-Z from the IFC, so as to disable the external transceiver by driving the value tied to the board to it.	
		State meaning	Asserted/Negated - The TE pin functions as an enable/disable for an external transceiver connected to the AD lines. In asserted state, it enables the external transceiver and when negated, disables the transceiver.
IFC_PAR[0:*n*]	I/O	GPCM data bus parity, GPCM address bus parity.	
		State meaning	Asserted/Negated - For GPCM during Write a parity bit is generated for each data byte sent over the bus. Unused byte lanes have undefined parity.

Table continues on the next page...

Table 13-2. Detailed signal description (continued)

Signal	I/O	Description	
			For GASIC a parity bit is sent for each address and data byte sent over the AD bus.
IFC_PERR_B	I	GASIC parity error indication. PERR_B is sampled a configurable number of IFC_CLK cycles after the assertion of SOF_L.	
		State meaning	Asserted/Negated - Parity Error is indicated by GASIC for wrong parity sent for address and data bytes.
IFC_CLK[0:*n*]	O	IFC external clock.	
		State meaning	This is external divided clock derived from the IFC module input clock. Division ratio is programmable.

13.2.1 Internal connectivity of the write protect and ready/busy signals

The IFC supports one write protect and one ready/busy signal per chip select (up to eight chip selects). This limited availability of write protect and ready/busy pins means that the available write protect and ready/busy signals are mapped to the eight chip selects that the IFC supports.

T2080 supports one write protect signal IFC_WP_B[0] and two ready/busy signals IFC_RB_B[0:1].

The single write protect signal supported is a logical OR of all eight write protects output from the IFC. This means that if software enables write protection for any one of the chip selects, the single write protect signal IFC_WP_B[0] is asserted and all NAND flash devices monitoring that signal will be “write protected.”

RB_B[1] is routed to the ready/busy input of the IFC which corresponds to chip select 1. The first ready/busy RB_B[0] is routed to the ready/busy inputs of the IFC block which correspond to chip selects 0 and 2-7. This means that a busy condition on any one of the (up to) seven devices associated with chip selects 0 and 2-7 will be seen by software as a busy condition for all seven of those chip selects.

13.3 IFC memory map/register definition

The IFC is allocated 8 KB of memory-mapped space. The memory map is divided into two parts (4 KB each):

- Common registers shared by NAND, NOR, and GPCM FCM
- Specific registers defined for the NAND FCM, NOR FCM (IFC global), and GPCM FCM (IFC run time) exclusively

IFC memory map

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
12_4000	IFC revision control register (IFC_REV)	32	R	See section	13.3.1/602
12_400C	Extended Chip Select Property Registers (IFC_CSPR0_EXT)	32	R/W	0000_0000h	13.3.2/603
12_4010	Chip-select property register n (IFC_CSPR0)	32	R/W	0000_0000h	13.3.3/603
12_4018	Extended Chip Select Property Registers (IFC_CSPR1_EXT)	32	R/W	0000_0000h	13.3.2/603
12_401C	Chip-select property register n (IFC_CSPR1)	32	R/W	0000_0000h	13.3.3/603
12_4024	Extended Chip Select Property Registers (IFC_CSPR2_EXT)	32	R/W	0000_0000h	13.3.2/603
12_4028	Chip-select property register n (IFC_CSPR2)	32	R/W	0000_0000h	13.3.3/603
12_4030	Extended Chip Select Property Registers (IFC_CSPR3_EXT)	32	R/W	0000_0000h	13.3.2/603
12_4034	Chip-select property register n (IFC_CSPR3)	32	R/W	0000_0000h	13.3.3/603
12_403C	Extended Chip Select Property Registers (IFC_CSPR4_EXT)	32	R/W	0000_0000h	13.3.2/603
12_4040	Chip-select property register n (IFC_CSPR4)	32	R/W	0000_0000h	13.3.3/603
12_4048	Extended Chip Select Property Registers (IFC_CSPR5_EXT)	32	R/W	0000_0000h	13.3.2/603
12_404C	Chip-select property register n (IFC_CSPR5)	32	R/W	0000_0000h	13.3.3/603
12_4054	Extended Chip Select Property Registers (IFC_CSPR6_EXT)	32	R/W	0000_0000h	13.3.2/603
12_4058	Chip-select property register n (IFC_CSPR6)	32	R/W	0000_0000h	13.3.3/603
12_4060	Extended Chip Select Property Registers (IFC_CSPR7_EXT)	32	R/W	0000_0000h	13.3.2/603
12_4064	Chip-select property register n (IFC_CSPR7)	32	R/W	0000_0000h	13.3.3/603
12_40A0	Address mask register (IFC_AMASK0)	32	R/W	0000_0000h	13.3.4/605
12_40AC	Address mask register (IFC_AMASK1)	32	R/W	0000_0000h	13.3.4/605
12_40B8	Address mask register (IFC_AMASK2)	32	R/W	0000_0000h	13.3.4/605

Table continues on the next page...

IFC memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
12_40C4	Address mask register (IFC_AMASK3)	32	R/W	0000_0000h	13.3.4/605
12_40D0	Address mask register (IFC_AMASK4)	32	R/W	0000_0000h	13.3.4/605
12_40DC	Address mask register (IFC_AMASK5)	32	R/W	0000_0000h	13.3.4/605
12_40E8	Address mask register (IFC_AMASK6)	32	R/W	0000_0000h	13.3.4/605
12_40F4	Address mask register (IFC_AMASK7)	32	R/W	0000_0000h	13.3.4/605
12_4130	Chip-Select Option Register - NAND Flash Mode (IFC_CSOR0_NAND)	32	R/W	0000_0000h	13.3.5/607
12_4130	Chip-Select Option Register - NOR Flash Mode (IFC_CSOR0_NOR)	32	R/W	0000_000Ch	13.3.6/609
12_4130	Chip-Select Option Register - GPCM (IFC_CSOR0_GPCM)	32	R/W	See section	13.3.7/612
12_4134	Extended Chip Select Option Register - NAND Flash Mode (IFC_CSOR0_EXT)	32	R/W	0010_0000h	13.3.8/616
12_413C	Chip-Select Option Register - NAND Flash Mode (IFC_CSOR1_NAND)	32	R/W	0000_0000h	13.3.5/607
12_413C	Chip-Select Option Register - NOR Flash Mode (IFC_CSOR1_NOR)	32	R/W	0000_000Ch	13.3.6/609
12_413C	Chip-Select Option Register - GPCM (IFC_CSOR1_GPCM)	32	R/W	See section	13.3.7/612
12_4140	Extended Chip Select Option Register - NAND Flash Mode (IFC_CSOR1_EXT)	32	R/W	0010_0000h	13.3.8/616
12_4148	Chip-Select Option Register - NAND Flash Mode (IFC_CSOR2_NAND)	32	R/W	0000_0000h	13.3.5/607
12_4148	Chip-Select Option Register - NOR Flash Mode (IFC_CSOR2_NOR)	32	R/W	0000_000Ch	13.3.6/609
12_4148	Chip-Select Option Register - GPCM (IFC_CSOR2_GPCM)	32	R/W	See section	13.3.7/612
12_414C	Extended Chip Select Option Register - NAND Flash Mode (IFC_CSOR2_EXT)	32	R/W	0010_0000h	13.3.8/616
12_4154	Chip-Select Option Register - NAND Flash Mode (IFC_CSOR3_NAND)	32	R/W	0000_0000h	13.3.5/607
12_4154	Chip-Select Option Register - NOR Flash Mode (IFC_CSOR3_NOR)	32	R/W	0000_000Ch	13.3.6/609
12_4154	Chip-Select Option Register - GPCM (IFC_CSOR3_GPCM)	32	R/W	See section	13.3.7/612
12_4158	Extended Chip Select Option Register - NAND Flash Mode (IFC_CSOR3_EXT)	32	R/W	0010_0000h	13.3.8/616
12_4160	Chip-Select Option Register - NAND Flash Mode (IFC_CSOR4_NAND)	32	R/W	0000_0000h	13.3.5/607
12_4160	Chip-Select Option Register - NOR Flash Mode (IFC_CSOR4_NOR)	32	R/W	0000_000Ch	13.3.6/609
12_4160	Chip-Select Option Register - GPCM (IFC_CSOR4_GPCM)	32	R/W	See section	13.3.7/612
12_4164	Extended Chip Select Option Register - NAND Flash Mode (IFC_CSOR4_EXT)	32	R/W	0010_0000h	13.3.8/616
12_416C	Chip-Select Option Register - NAND Flash Mode (IFC_CSOR5_NAND)	32	R/W	0000_0000h	13.3.5/607

Table continues on the next page...

IFC memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
12_416C	Chip-Select Option Register - NOR Flash Mode (IFC_CSOR5_NOR)	32	R/W	0000_000Ch	13.3.6/609
12_416C	Chip-Select Option Register - GPCM (IFC_CSOR5_GPCM)	32	R/W	See section	13.3.7/612
12_4170	Extended Chip Select Option Register - NAND Flash Mode (IFC_CSOR5_EXT)	32	R/W	0010_0000h	13.3.8/616
12_4178	Chip-Select Option Register - NAND Flash Mode (IFC_CSOR6_NAND)	32	R/W	0000_0000h	13.3.5/607
12_4178	Chip-Select Option Register - NOR Flash Mode (IFC_CSOR6_NOR)	32	R/W	0000_000Ch	13.3.6/609
12_4178	Chip-Select Option Register - GPCM (IFC_CSOR6_GPCM)	32	R/W	See section	13.3.7/612
12_417C	Extended Chip Select Option Register - NAND Flash Mode (IFC_CSOR6_EXT)	32	R/W	0010_0000h	13.3.8/616
12_4184	Chip-Select Option Register - NAND Flash Mode (IFC_CSOR7_NAND)	32	R/W	0000_0000h	13.3.5/607
12_4184	Chip-Select Option Register - NOR Flash Mode (IFC_CSOR7_NOR)	32	R/W	0000_000Ch	13.3.6/609
12_4184	Chip-Select Option Register - GPCM (IFC_CSOR7_GPCM)	32	R/W	See section	13.3.7/612
12_4188	Extended Chip Select Option Register - NAND Flash Mode (IFC_CSOR7_EXT)	32	R/W	0010_0000h	13.3.8/616
12_41C0	Flash Timing Register 0 for Chip-Select n - NAND Flash Mode (IFC_FTIM0_CS0_NAND)	32	R/W	0000_0000h	13.3.9/616
12_41C0	Flash Timing Register 0 for CSn - NOR Flash Mode (IFC_FTIM0_CS0_NOR)	32	R/W	0000_0000h	13.3.10/ 618
12_41C0	Flash Timing Register 0 for CSn - Normal GPCM Mode (IFC_FTIM0_CS0_GPCM)	32	R/W	0000_0000h	13.3.11/ 620
12_41C4	Flash Timing Register 1 for Chip-Select n - NAND Flash Mode (IFC_FTIM1_CS0_NAND)	32	R/W	0000_0000h	13.3.12/ 621
12_41C4	Flash Timing Register 1 for CSn - NOR Flash Mode (IFC_FTIM1_CS0_NOR)	32	R/W	0000_0000h	13.3.13/ 622
12_41C4	Flash Timing Register 1 for CSn - Normal GPCM Mode (IFC_FTIM1_CS0_GPCM)	32	R/W	0000_0000h	13.3.14/ 623
12_41C8	Flash Timing Register 2 for Chip-Select n - NAND Flash Mode (IFC_FTIM2_CS0_NAND)	32	R/W	0000_0000h	13.3.15/ 624
12_41C8	Flash Timing Register 2 for CSn - NOR Flash Mode (IFC_FTIM2_CS0_NOR)	32	R/W	0000_0000h	13.3.16/ 625
12_41C8	Flash Timing Register 2 for CSn - Normal GPCM Mode (IFC_FTIM2_CS0_GPCM)	32	R/W	0000_0000h	13.3.17/ 626
12_41CC	Flash Timing Register 3 for Chip-Select n - NAND Flash Mode (IFC_FTIM3_CS0_NAND)	32	R/W	0000_0000h	13.3.18/ 627
12_41CC	Flash Timing Register 3 for CSn - NOR Flash Mode (IFC_FTIM3_CS_NOR)	32	R/W	0000_0000h	13.3.19/ 628
12_41CC	Flash Timing Register 3 for CSn - Normal GPCM Mode (IFC_FTIM3_CS0_GPCM)	32	R/W	0000_0000h	13.3.20/ 628

Table continues on the next page...

IFC memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
12_41F0	Flash Timing Register 0 for Chip-Select n - NAND Flash Mode (IFC_FTIM0_CS1_NAND)	32	R/W	0000_0000h	13.3.9/616
12_41F0	Flash Timing Register 0 for CSn - NOR Flash Mode (IFC_FTIM0_CS1_NOR)	32	R/W	0000_0000h	13.3.10/ 618
12_41F0	Flash Timing Register 0 for CSn - Normal GPCM Mode (IFC_FTIM0_CS1_GPCM)	32	R/W	0000_0000h	13.3.11/ 620
12_41F4	Flash Timing Register 1 for Chip-Select n - NAND Flash Mode (IFC_FTIM1_CS1_NAND)	32	R/W	0000_0000h	13.3.12/ 621
12_41F4	Flash Timing Register 1 for CSn - NOR Flash Mode (IFC_FTIM1_CS1_NOR)	32	R/W	0000_0000h	13.3.13/ 622
12_41F4	Flash Timing Register 1 for CSn - Normal GPCM Mode (IFC_FTIM1_CS1_GPCM)	32	R/W	0000_0000h	13.3.14/ 623
12_41F8	Flash Timing Register 2 for Chip-Select n - NAND Flash Mode (IFC_FTIM2_CS1_NAND)	32	R/W	0000_0000h	13.3.15/ 624
12_41F8	Flash Timing Register 2 for CSn - NOR Flash Mode (IFC_FTIM2_CS1_NOR)	32	R/W	0000_0000h	13.3.16/ 625
12_41F8	Flash Timing Register 2 for CSn - Normal GPCM Mode (IFC_FTIM2_CS1_GPCM)	32	R/W	0000_0000h	13.3.17/ 626
12_41FC	Flash Timing Register 3 for Chip-Select n - NAND Flash Mode (IFC_FTIM3_CS1_NAND)	32	R/W	0000_0000h	13.3.18/ 627
12_41FC	Flash Timing Register 3 for CSn - Normal GPCM Mode (IFC_FTIM3_CS1_GPCM)	32	R/W	0000_0000h	13.3.20/ 628
12_4220	Flash Timing Register 0 for Chip-Select n - NAND Flash Mode (IFC_FTIM0_CS2_NAND)	32	R/W	0000_0000h	13.3.9/616
12_4220	Flash Timing Register 0 for CSn - NOR Flash Mode (IFC_FTIM0_CS2_NOR)	32	R/W	0000_0000h	13.3.10/ 618
12_4220	Flash Timing Register 0 for CSn - Normal GPCM Mode (IFC_FTIM0_CS2_GPCM)	32	R/W	0000_0000h	13.3.11/ 620
12_4224	Flash Timing Register 1 for Chip-Select n - NAND Flash Mode (IFC_FTIM1_CS2_NAND)	32	R/W	0000_0000h	13.3.12/ 621
12_4224	Flash Timing Register 1 for CSn - NOR Flash Mode (IFC_FTIM1_CS2_NOR)	32	R/W	0000_0000h	13.3.13/ 622
12_4224	Flash Timing Register 1 for CSn - Normal GPCM Mode (IFC_FTIM1_CS2_GPCM)	32	R/W	0000_0000h	13.3.14/ 623
12_4228	Flash Timing Register 2 for Chip-Select n - NAND Flash Mode (IFC_FTIM2_CS2_NAND)	32	R/W	0000_0000h	13.3.15/ 624
12_4228	Flash Timing Register 2 for CSn - NOR Flash Mode (IFC_FTIM2_CS2_NOR)	32	R/W	0000_0000h	13.3.16/ 625
12_4228	Flash Timing Register 2 for CSn - Normal GPCM Mode (IFC_FTIM2_CS2_GPCM)	32	R/W	0000_0000h	13.3.17/ 626
12_422C	Flash Timing Register 3 for Chip-Select n - NAND Flash Mode (IFC_FTIM3_CS2_NAND)	32	R/W	0000_0000h	13.3.18/ 627
12_422C	Flash Timing Register 3 for CSn - Normal GPCM Mode (IFC_FTIM3_CS2_GPCM)	32	R/W	0000_0000h	13.3.20/ 628

Table continues on the next page...

IFC memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
12_4250	Flash Timing Register 0 for Chip-Select n - NAND Flash Mode (IFC_FTIM0_CS3_NAND)	32	R/W	0000_0000h	13.3.9/616
12_4250	Flash Timing Register 0 for CSn - NOR Flash Mode (IFC_FTIM0_CS3_NOR)	32	R/W	0000_0000h	13.3.10/ 618
12_4250	Flash Timing Register 0 for CSn - Normal GPCM Mode (IFC_FTIM0_CS3_GPCM)	32	R/W	0000_0000h	13.3.11/ 620
12_4254	Flash Timing Register 1 for Chip-Select n - NAND Flash Mode (IFC_FTIM1_CS3_NAND)	32	R/W	0000_0000h	13.3.12/ 621
12_4254	Flash Timing Register 1 for CSn - NOR Flash Mode (IFC_FTIM1_CS3_NOR)	32	R/W	0000_0000h	13.3.13/ 622
12_4254	Flash Timing Register 1 for CSn - Normal GPCM Mode (IFC_FTIM1_CS3_GPCM)	32	R/W	0000_0000h	13.3.14/ 623
12_4258	Flash Timing Register 2 for Chip-Select n - NAND Flash Mode (IFC_FTIM2_CS3_NAND)	32	R/W	0000_0000h	13.3.15/ 624
12_4258	Flash Timing Register 2 for CSn - NOR Flash Mode (IFC_FTIM2_CS3_NOR)	32	R/W	0000_0000h	13.3.16/ 625
12_4258	Flash Timing Register 2 for CSn - Normal GPCM Mode (IFC_FTIM2_CS3_GPCM)	32	R/W	0000_0000h	13.3.17/ 626
12_425C	Flash Timing Register 3 for Chip-Select n - NAND Flash Mode (IFC_FTIM3_CS3_NAND)	32	R/W	0000_0000h	13.3.18/ 627
12_425C	Flash Timing Register 3 for CSn - Normal GPCM Mode (IFC_FTIM3_CS3_GPCM)	32	R/W	0000_0000h	13.3.20/ 628
12_4280	Flash Timing Register 0 for Chip-Select n - NAND Flash Mode (IFC_FTIM0_CS4_NAND)	32	R/W	0000_0000h	13.3.9/616
12_4280	Flash Timing Register 0 for CSn - NOR Flash Mode (IFC_FTIM0_CS4_NOR)	32	R/W	0000_0000h	13.3.10/ 618
12_4280	Flash Timing Register 0 for CSn - Normal GPCM Mode (IFC_FTIM0_CS4_GPCM)	32	R/W	0000_0000h	13.3.11/ 620
12_4284	Flash Timing Register 1 for Chip-Select n - NAND Flash Mode (IFC_FTIM1_CS4_NAND)	32	R/W	0000_0000h	13.3.12/ 621
12_4284	Flash Timing Register 1 for CSn - NOR Flash Mode (IFC_FTIM1_CS4_NOR)	32	R/W	0000_0000h	13.3.13/ 622
12_4284	Flash Timing Register 1 for CSn - Normal GPCM Mode (IFC_FTIM1_CS4_GPCM)	32	R/W	0000_0000h	13.3.14/ 623
12_4288	Flash Timing Register 2 for Chip-Select n - NAND Flash Mode (IFC_FTIM2_CS4_NAND)	32	R/W	0000_0000h	13.3.15/ 624
12_4288	Flash Timing Register 2 for CSn - NOR Flash Mode (IFC_FTIM2_CS4_NOR)	32	R/W	0000_0000h	13.3.16/ 625
12_4288	Flash Timing Register 2 for CSn - Normal GPCM Mode (IFC_FTIM2_CS4_GPCM)	32	R/W	0000_0000h	13.3.17/ 626
12_428C	Flash Timing Register 3 for Chip-Select n - NAND Flash Mode (IFC_FTIM3_CS4_NAND)	32	R/W	0000_0000h	13.3.18/ 627
12_428C	Flash Timing Register 3 for CSn - Normal GPCM Mode (IFC_FTIM3_CS4_GPCM)	32	R/W	0000_0000h	13.3.20/ 628

Table continues on the next page...

IFC memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
12_42B0	Flash Timing Register 0 for Chip-Select n - NAND Flash Mode (IFC_FTIM0_CS5_NAND)	32	R/W	0000_0000h	13.3.9/616
12_42B0	Flash Timing Register 0 for CSn - NOR Flash Mode (IFC_FTIM0_CS5_NOR)	32	R/W	0000_0000h	13.3.10/ 618
12_42B0	Flash Timing Register 0 for CSn - Normal GPCM Mode (IFC_FTIM0_CS5_GPCM)	32	R/W	0000_0000h	13.3.11/ 620
12_42B4	Flash Timing Register 1 for Chip-Select n - NAND Flash Mode (IFC_FTIM1_CS5_NAND)	32	R/W	0000_0000h	13.3.12/ 621
12_42B4	Flash Timing Register 1 for CSn - NOR Flash Mode (IFC_FTIM1_CS5_NOR)	32	R/W	0000_0000h	13.3.13/ 622
12_42B4	Flash Timing Register 1 for CSn - Normal GPCM Mode (IFC_FTIM1_CS5_GPCM)	32	R/W	0000_0000h	13.3.14/ 623
12_42B8	Flash Timing Register 2 for Chip-Select n - NAND Flash Mode (IFC_FTIM2_CS5_NAND)	32	R/W	0000_0000h	13.3.15/ 624
12_42B8	Flash Timing Register 2 for CSn - NOR Flash Mode (IFC_FTIM2_CS5_NOR)	32	R/W	0000_0000h	13.3.16/ 625
12_42B8	Flash Timing Register 2 for CSn - Normal GPCM Mode (IFC_FTIM2_CS5_GPCM)	32	R/W	0000_0000h	13.3.17/ 626
12_42BC	Flash Timing Register 3 for Chip-Select n - NAND Flash Mode (IFC_FTIM3_CS5_NAND)	32	R/W	0000_0000h	13.3.18/ 627
12_42BC	Flash Timing Register 3 for CSn - Normal GPCM Mode (IFC_FTIM3_CS5_GPCM)	32	R/W	0000_0000h	13.3.20/ 628
12_42E0	Flash Timing Register 0 for Chip-Select n - NAND Flash Mode (IFC_FTIM0_CS6_NAND)	32	R/W	0000_0000h	13.3.9/616
12_42E0	Flash Timing Register 0 for CSn - NOR Flash Mode (IFC_FTIM0_CS6_NOR)	32	R/W	0000_0000h	13.3.10/ 618
12_42E0	Flash Timing Register 0 for CSn - Normal GPCM Mode (IFC_FTIM0_CS6_GPCM)	32	R/W	0000_0000h	13.3.11/ 620
12_42E4	Flash Timing Register 1 for Chip-Select n - NAND Flash Mode (IFC_FTIM1_CS6_NAND)	32	R/W	0000_0000h	13.3.12/ 621
12_42E4	Flash Timing Register 1 for CSn - NOR Flash Mode (IFC_FTIM1_CS6_NOR)	32	R/W	0000_0000h	13.3.13/ 622
12_42E4	Flash Timing Register 1 for CSn - Normal GPCM Mode (IFC_FTIM1_CS6_GPCM)	32	R/W	0000_0000h	13.3.14/ 623
12_42E8	Flash Timing Register 2 for Chip-Select n - NAND Flash Mode (IFC_FTIM2_CS6_NAND)	32	R/W	0000_0000h	13.3.15/ 624
12_42E8	Flash Timing Register 2 for CSn - NOR Flash Mode (IFC_FTIM2_CS6_NOR)	32	R/W	0000_0000h	13.3.16/ 625
12_42E8	Flash Timing Register 2 for CSn - Normal GPCM Mode (IFC_FTIM2_CS6_GPCM)	32	R/W	0000_0000h	13.3.17/ 626
12_42EC	Flash Timing Register 3 for Chip-Select n - NAND Flash Mode (IFC_FTIM3_CS6_NAND)	32	R/W	0000_0000h	13.3.18/ 627
12_42EC	Flash Timing Register 3 for CSn - Normal GPCM Mode (IFC_FTIM3_CS6_GPCM)	32	R/W	0000_0000h	13.3.20/ 628

Table continues on the next page...

IFC memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
12_4310	Flash Timing Register 0 for Chip-Select n - NAND Flash Mode (IFC_FTIM0_CS7_NAND)	32	R/W	0000_0000h	13.3.9/616
12_4310	Flash Timing Register 0 for CSn - NOR Flash Mode (IFC_FTIM0_CS7_NOR)	32	R/W	0000_0000h	13.3.10/ 618
12_4310	Flash Timing Register 0 for CSn - Normal GPCM Mode (IFC_FTIM0_CS7_GPCM)	32	R/W	0000_0000h	13.3.11/ 620
12_4314	Flash Timing Register 1 for Chip-Select n - NAND Flash Mode (IFC_FTIM1_CS7_NAND)	32	R/W	0000_0000h	13.3.12/ 621
12_4314	Flash Timing Register 1 for CSn - NOR Flash Mode (IFC_FTIM1_CS7_NOR)	32	R/W	0000_0000h	13.3.13/ 622
12_4314	Flash Timing Register 1 for CSn - Normal GPCM Mode (IFC_FTIM1_CS7_GPCM)	32	R/W	0000_0000h	13.3.14/ 623
12_4318	Flash Timing Register 2 for Chip-Select n - NAND Flash Mode (IFC_FTIM2_CS7_NAND)	32	R/W	0000_0000h	13.3.15/ 624
12_4318	Flash Timing Register 2 for CSn - NOR Flash Mode (IFC_FTIM2_CS7_NOR)	32	R/W	0000_0000h	13.3.16/ 625
12_4318	Flash Timing Register 2 for CSn - Normal GPCM Mode (IFC_FTIM2_CS7_GPCM)	32	R/W	0000_0000h	13.3.17/ 626
12_431C	Flash Timing Register 3 for Chip-Select n - NAND Flash Mode (IFC_FTIM3_CS7_NAND)	32	R/W	0000_0000h	13.3.18/ 627
12_431C	Flash Timing Register 3 for CSn - Normal GPCM Mode (IFC_FTIM3_CS7_GPCM)	32	R/W	0000_0000h	13.3.20/ 628
12_4400	Ready busy status for each chip-select (IFC_RB_STAT)	32	R	See section	13.3.21/ 629
12_440C	General control register (IFC_GCR)	32	R/W	0000_0000h	13.3.22/ 630
12_4418	Common event and error status register (IFC_CM_EVTER_STAT)	32	R/W	0000_0000h	13.3.23/ 631
12_4424	Common event and error enable register (IFC_CM_EVTER_EN)	32	R/W	8000_0000h	13.3.24/ 632
12_4430	Common event and error interrupt enable register (IFC_CM_EVTER_INTR_EN)	32	R/W	0000_0000h	13.3.25/ 633
12_443C	Common transfer error attributes register 0 (IFC_CM_ERATTR0)	32	R	0000_0000h	13.3.26/ 633
12_4440	Common transfer error attributes register 1 (IFC_CM_ERATTR1)	32	R	0000_0000h	13.3.27/ 635
12_444C	Clock control register (IFC_CCR)	32	R/W	0300_8000h	13.3.28/ 635
12_4450	Clock status register (IFC_CSR)	32	R	See section	13.3.29/ 637
12_5000	NAND configuration register (IFC_NCFGR)	32	R/W	0000_0000h	13.3.30/ 638
12_5014	NAND Flash Command Register 0 (IFC_NAND_FCR0)	32	R/W	0000_0000h	13.3.31/ 640

Table continues on the next page...

IFC memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
12_5018	NAND flash command register 1 (IFC_NAND_FCR1)	32	R/W	0000_0000h	13.3.32/ 640
12_503C	Flash Row Address Register n (IFC_ROW0)	32	R/W	0000_0000h	13.3.33/ 641
12_5044	Flash COL Address Register n (IFC_COL0)	32	R/W	0000_0000h	13.3.34/ 641
12_5044	Flash COL Address Register for 2 KB Large-Page Device (IFC_COL0_2KB)	32	R/W	0000_0000h	13.3.35/ 643
12_5044	Flash COL Address Register for 4 KByte Large-Page Device (IFC_COL0_4KB)	32	R/W	0000_0000h	13.3.36/ 644
12_504C	Flash Row Address Register n (IFC_ROW1)	32	R/W	0000_0000h	13.3.33/ 641
12_5054	Flash COL Address Register n (IFC_COL1)	32	R/W	0000_0000h	13.3.34/ 641
12_5054	Flash COL Address Register for 2 KB Large-Page Device (IFC_COL1_2KB)	32	R/W	0000_0000h	13.3.35/ 643
12_5054	Flash COL Address Register for 4 KByte Large-Page Device (IFC_COL1_4KB)	32	R/W	0000_0000h	13.3.36/ 644
12_505C	Flash Row Address Register n (IFC_ROW2)	32	R/W	0000_0000h	13.3.33/ 641
12_5064	Flash COL Address Register n (IFC_COL2)	32	R/W	0000_0000h	13.3.34/ 641
12_5064	Flash COL Address Register for 2 KB Large-Page Device (IFC_COL2_2KB)	32	R/W	0000_0000h	13.3.35/ 643
12_5064	Flash COL Address Register for 4 KByte Large-Page Device (IFC_COL2_4KB)	32	R/W	0000_0000h	13.3.36/ 644
12_506C	Flash Row Address Register n (IFC_ROW3)	32	R/W	0000_0000h	13.3.33/ 641
12_5074	Flash COL Address Register n (IFC_COL3)	32	R/W	0000_0000h	13.3.34/ 641
12_5074	Flash COL Address Register for 2 KB Large-Page Device (IFC_COL3_2KB)	32	R/W	0000_0000h	13.3.35/ 643
12_5074	Flash COL Address Register for 4 KByte Large-Page Device (IFC_COL3_4KB)	32	R/W	0000_0000h	13.3.36/ 644
12_5108	Flash Byte Count Register for NAND Flash (IFC_NAND_BC)	32	R/W	0000_0000h	13.3.37/ 645
12_5110	NAND flash instruction register 0 (IFC_NAND_FIR0)	32	R/W	0000_0000h	13.3.38/ 646
12_5114	NAND flash instruction register 1 (IFC_NAND_FIR1)	32	R/W	0000_0000h	13.3.39/ 648
12_5118	NAND flash instruction register 2 (IFC_NAND_FIR2)	32	R/W	0000_0000h	13.3.40/ 649
12_515C	NAND chip-select register (IFC_NAND_CSEL)	32	R/W	0000_0000h	13.3.41/ 650

Table continues on the next page...

IFC memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
12_5164	NAND operation sequence start (IFC_NANDSEQ_STRT)	32	R/W	0000_0000h	13.3.42/ 650
12_516C	NAND event and error status register (IFC_NAND_EVTER_STAT)	32	w1c	0000_0000h	13.3.43/ 652
12_5174	NAND page read completion event status register (IFC_PGRDCMPL_EVT_STAT)	32	w1c	0000_0000h	13.3.44/ 654
12_5180	NAND event and error enable register (IFC_NAND_EVTER_EN)	32	R/W	AE00_0000h	13.3.45/ 656
12_518C	NAND event and error interrupt enable register (IFC_NAND_EVTER_INTR_EN)	32	R/W	0000_0000h	13.3.46/ 658
12_5198	NAND transfer error attributes register 0 (IFC_NAND_ERATTR0)	32	R	0000_0000h	13.3.47/ 659
12_519C	NAND transfer error attributes register 1 (IFC_NAND_ERATTR1)	32	R	0000_0000h	13.3.48/ 661
12_51E0	NAND flash status register (IFC_NAND_FSR)	32	R	0000_0000h	13.3.49/ 661
12_51E8	ECC status and result of flash operation register 0 (IFC_ECCSTAT0)	32	R	0000_0000h	13.3.50/ 662
12_51EC	ECC status and result of flash operation register 1 (IFC_ECCSTAT1)	32	R	0000_0000h	13.3.51/ 664
12_51F0	ECC status and result of flash operation register 2 (IFC_ECCSTAT2)	32	R	0000_0000h	13.3.52/ 665
12_51F4	ECC status and result of flash operation register 3 (IFC_ECCSTAT3)	32	R	0000_0000h	13.3.53/ 666
12_5278	NAND control register (IFC_NANDCR)	32	R/W	1E00_0000h	13.3.54/ 666
12_5284	NAND autoboot trigger register (IFC_NAND_AUTOBOOT_TRGR)	32	W	0000_0000h	13.3.55/ 667
12_528C	NAND flash memory data register (IFC_NAND_MDR)	32	R	0000_0000h	13.3.56/ 669
12_5400	NOR event and error status register (IFC_NOR_EVTER_STAT)	32	w1c	0000_0000h	13.3.57/ 670
12_540C	NOR event and error enable register (IFC_NOR_EVTER_EN)	32	R/W	8500_0000h	13.3.58/ 672
12_5418	NOR event and error interrupt enable register (IFC_NOR_EVTER_INTR_EN)	32	R/W	0000_0000h	13.3.59/ 673
12_5424	NOR transfer error attributes register 0 (IFC_NOR_ERATTR0)	32	R	0000_0000h	13.3.60/ 674
12_5428	NOR transfer error attribute register 1 (IFC_NOR_ERATTR1)	32	R	0000_0000h	13.3.61/ 676
12_542C	NOR transfer error attribute register 2 (IFC_NOR_ERATTR2)	32	R	0000_0000h	13.3.62/ 676
12_5440	NOR control register (IFC_NORCR)	32	R/W	000F_0000h	13.3.63/ 677

Table continues on the next page...

IFC memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
12_5800	GPCM event and error status register (IFC_GPCM_EVTER_STAT)	32	w1c	0000_0000h	13.3.64/ 678
12_580C	GPCM event and error enable register (IFC_GPCM_EVTER_EN)	32	R/W	0540_0000h	13.3.65/ 681
12_5818	GPCM event and error interrupt enable register (IFC_GPCM_EVTER_INTR_EN)	32	R/W	0000_0000h	13.3.66/ 682
12_5824	GPCM transfer error attributes register 0 (IFC_GPCM_ERATTR0)	32	R	0000_0000h	13.3.67/ 684
12_5828	GPCM transfer error attributes register 1 (IFC_GPCM_ERATTR1)	32	R	0000_0000h	13.3.68/ 685
12_582C	GPCM transfer error attributes register 2 (IFC_GPCM_ERATTR2)	32	R	0000_0000h	13.3.69/ 686
12_5830	GPCM status register (IFC_GPCM_STAT)	32	R	0000_0000h	13.3.70/ 688

13.3.1 IFC revision control register (IFC_REV)

This register represents the revision number of the IFC.

Address: 12_4000h base + 0h offset = 12_4000h

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				REV_MAJ				Reserved				REV_MIN				Reserved															
W																																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	n	n	n	n	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IFC_REV field descriptions

Field	Description
0–3 -	This field is reserved.
4–7 REV_MAJ	Major Revision. It represents the major revision of IFC
8–11 -	This field is reserved.
12–15 REV_MIN	Minor Revision. It represents the minor revision of IFC.
16–31 -	This field is reserved.

13.3.2 Extended Chip Select Property Registers (IFC_CSPR*n*_EXT)

The extended chip select property register (CSPRn_EXT) contains the extended base address, that is, the most significant bits (msb) of of the base address.

Address: 12_4000h base + Ch offset + (12d × 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	Reserved																								BA_EXT							
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

IFC_CSPR*n*_EXT field descriptions

Field	Description
0–23 -	This field is reserved.
24–31 BA_EXT	Extended Base Address: This field contains the msbs of the base address. Complete base address can be represented by concatenating CSPRn_EXT[BA_EXT] and CSPR*n*[BA] fields. Each chip selects's base register value is compared to the address on the AXI address bus to determine if the AXI master is accessing a memory bank controlled by the IFC. Address decoding is performed by using the address mask bit in AMASK*n*[AM] field more details of decoding is given in AMASK register description.

13.3.3 Chip-select property register n (IFC_CSPR*n*)

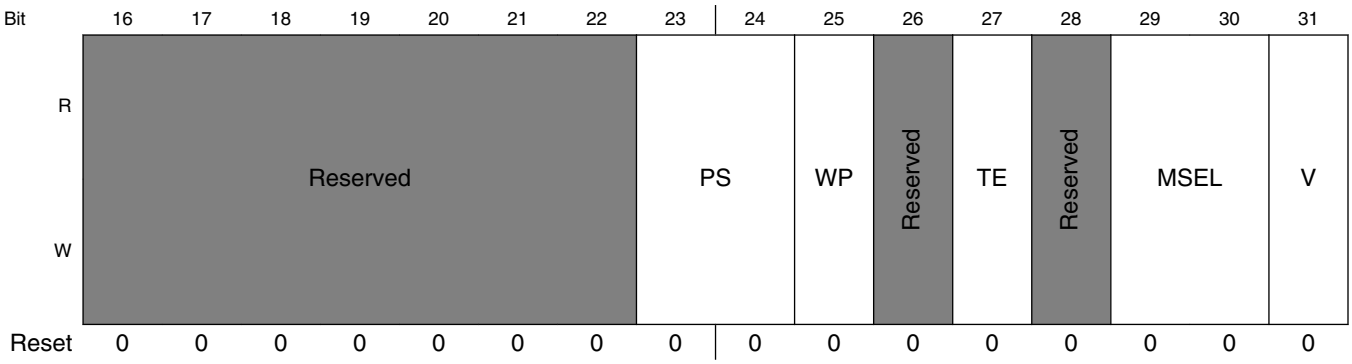
The chip-select property register (CSPR*n*) contains the base address and memory attributes for each bank.

NOTE

CSPR0: Only chip-select 0 is used for booting. PS, MSEL[1] is obtained from the configuration word when boot_load/ rcw_load comes. V will be set for CSPR0 when boot_load/ rcw_load comes with valid port size.

Address: 12_4000h base + 10h offset + (12d × i), where i=0d to 7d

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



IFC_CSPR*n* field descriptions

Field	Description
0–15 BA	Base Address: Each base register value is compared to the address on the AXI address bus to determine if the AXI master is accessing a memory bank controlled by the IFC. Used with the address mask bit.
16–22 -	This field is reserved.
23–24 PS	Port Size-Specifies the port size of this memory region. For BR0, PS is configured through reset configuration word as loaded during power on reset. For all other banks the value is reset to 00 (port size not defined). 00 Reserved 01 8 bit 10 16 bit 11 Reserved
25 WP	Write Protect: **NOTE:** 1. This bit is valid only for NAND and NOR; for GPCM this bit should not be set. **NOTE:** 2. If CS0 is used for booting from NAND or NOR, CSPR0[WP] will be set (write protected). Software must clear this bit after completing the boot operation to permit write operations on CS0 0 Read and write accesses are allowed. 1 Only read accesses are allowed. The memory controller does not assert chip-select on write cycles to this memory bank for NOR flash. (Refer Write protect for more details)
26 -	This field is reserved.
27 TE	External Transceiver Enable. It specifies the value that will be driven on TE pin when a particular CSn is selected. 0 Logic 0 will be driven on TE pin 1 Logic 1 will be driven on TE pin
28 -	This field is reserved.
29–30 MSEL	Machine Select 00 NOR flash 01 NAND flash 10 GPCM 11 Reserved

Table continues on the next page...

IFC_CSPR*n* field descriptions (continued)

Field	Description
31 V	Valid: Indicates that the contents of CSPRn are valid. 1 Bank is valid 0 Bank is invalid

13.3.4 Address mask register (IFC_AMASK*n*)

The address mask registers (AMASKn) contains the 16-bit address mask field for all four memory banks. The selected 16-bit address mask field masks corresponding CSPR*n*[BA] fields. The 16 lsbs of the 32-bit internal address do not participate in bank address matching in selecting a bank for access. Masking address bits independently allows external devices of different size address ranges to be used. Address mask bits can be set or cleared in any order in the field, allowing a resource to reside in more than one area of the address map.

The IFC only has 32 address pins at the external memory interface side. Since the system side address bus width is 40 bits, only the lower 32 bits will be given out. The upper 8 bits are used only for address decoding to identify the bank (chip select) on which access will be performed. There is no masking of the upper 8 bits of the system side address so it will be compared with CSPRn_EXT[BA_EXT] field as it is. Therefore, if the user programs the same value in the CSPRn_EXT[BA_EXT] field, the entire address range of the IFC will be 4 GB. If different values are programmed in the CSPRn_EXT[BA_EXT] field, the address range of the IFC will then be (number of chip selects) x 4 GB, that is, 32 GB.

The following logic is used for address decoding:

24-bit base address is formed as, BASE_ADDRn[0:23] = {CSPRn_EXT[BA_EXT], CSPRn[BA]}, n=0-7 (chip select)

- Select CSn (chip select n) if {BASE_ADDRn[0:7], (BASE_ADDRn[8:23] & AMn[0:15])} == {SYSTEM_ADDR[39:32], (SYSTEM_ADDR[31:16] & AMn[0:15])}, where
  - SYSTEM_ADDR is a 40-bit incoming address from the system side
  - Index 39-32 represents 8 address msbs
  - SYSTEM_ADDR[31:16] represent the next lower 16 address bits and the remaining 16 lsbs are not used in the bank selection
  - In the logic mentioned above, the 8 msbs are not masked and are used as is for comparison.

NOTE

Chip select 0 (CS0) is used for boot purpose; if the boot source is NOR, then it must be executed in place. During booting, the IFC registers (including BA_EXT, BA, and AMASK registers) also are modified. If there is race condition between the update of BA_EXT, BA, and AMASK registers, a chip select error may occur. To avoid this error and to map every transaction in CS0 during boot, the upper 8 bits of the system address and the base address, will be masked before comparision. Masking of these fields will be turned off only after the first write transaction comes to modify the AMASK0 register.

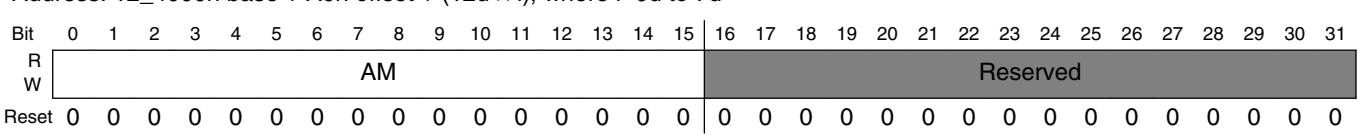
After reset, until a first write transaction comes to update AMASK0 register, all the transactions coming from the system side will always be mapped to CS0. After receiving the first write transaction to update AMASK0, the chip select decoding logic will work as per the above mentioned equation.

The table below shows memory bank size from 64 KB to 4 GB.

Table 13-25. Memory Bank Sizes in Relation to Address Mask

AM	Memory Bank Size
0000_0000_0000_0000	4 GBytes
1000_0000_0000_0000	2 Gbytes
1100_0000_0000_0000	1 Gbytes
1110_0000_0000_0000	512 Mbytes
1111_0000_0000_0000	256 Mbytes
1111_1000_0000_0000	128 Mbytes
1111_1100_0000_0000	64 Mbytes
1111_1110_0000_0000	32 Mbytes
1111_1111_0000_0000	16 Mbytes
1111_1111_1000_0000	8 Mbytes
1111_1111_1100_0000	4 Mbytes
1111_1111_1110_0000	2 Mbytes
1111_1111_1111_0000	1 Mbytes
1111_1111_1111_1000	512 Kbytes
1111_1111_1111_1100	256 Kbytes
1111_1111_1111_1110	128 Kbytes
1111_1111_1111_1111	64 Kbytes

Address: 12_4000h base + A0h offset + (12d × i), where i=0d to 7d



IFC_AMASK*n* field descriptions

Field	Description
0–15 AM	16-bit Address mask corresponding to memory bank
16–31 -	This field is reserved.

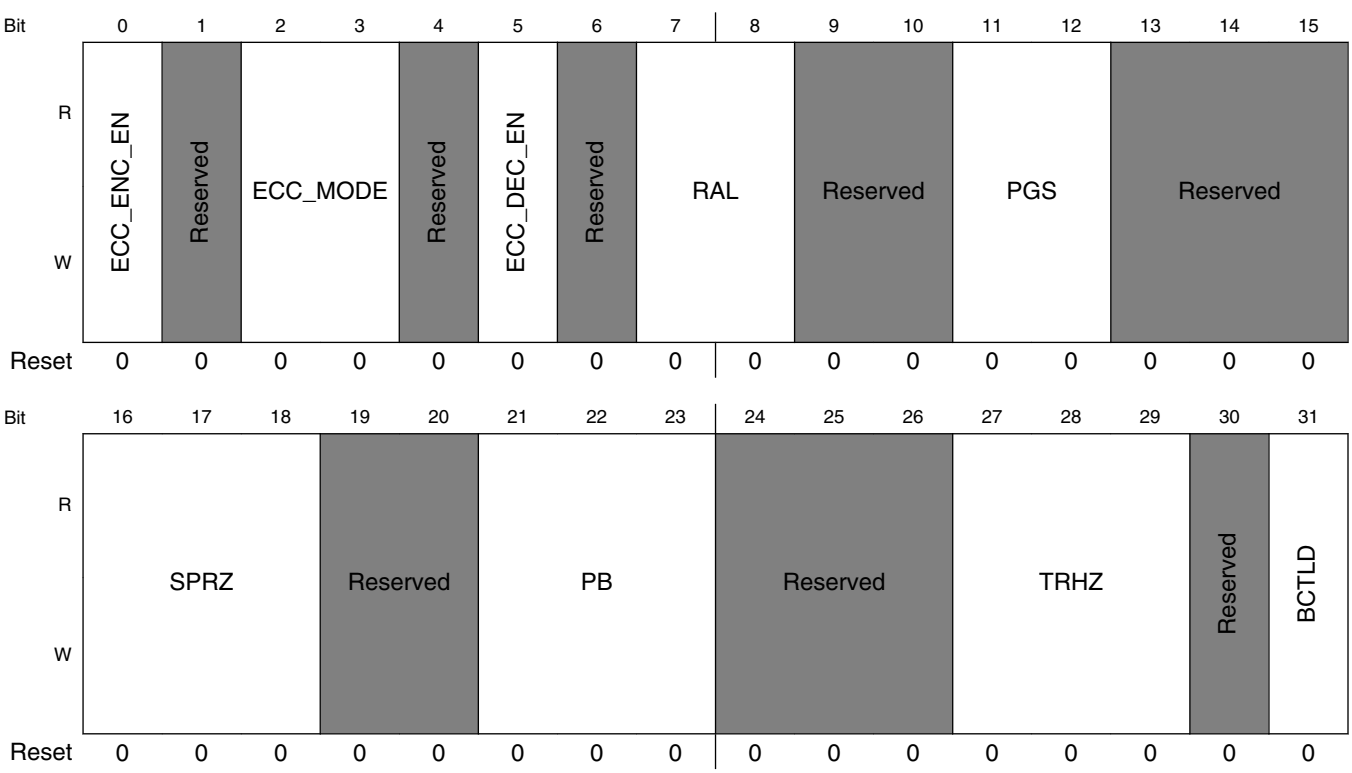
13.3.5 Chip-Select Option Register - NAND Flash Mode (IFC_CSOR*n*_NAND)

The following shows the CSOR*n* fields when CSPR*n*[MSEL] selects the NAND flash mode.

NOTE

- CSOR0 is the only chip-select used for booting.
- ECC_DEC_EN, ECC_MODE[1], RAL, and PGS are obtained from the configuration word when boot_load/ rcw_load occurs.
- The reset value of TRHZ is programmable by passing the parameter.

Address: 12_4000h base + 130h offset + (12d × i), where i=0d to 7d



IFC_CSOR*n*_NAND field descriptions

Field	Description
0 ECC_ENC_EN	ECC Encoder Enable/Disable bit: **NOTE:** The encoder should be enabled only when performing full page operations. In case of partial page operations, the encoder should be disabled. 0 ECC encoding disabled 1 ECC Encoding enabled
1 -	This field is reserved.
2–3 ECC_MODE	ECC Mode of operation 00 4 bit correction per 512 byte data sector 01 8 bit correction per 512 byte data sector 10 24 bit correction per 1Kbyte sector 11 40 bit correction per 1Kbyte sector
4 -	This field is reserved.
5 ECC_DEC_EN	ECC Decoding Enable/Disable bit **NOTE:** The decoder should be enabled only when performing full page operations. In case of partial page operations, the decoder should be disabled 0 ECC decoding disabled 1 ECC decoding enabled
6 -	This field is reserved.
7–8 RAL	Row Address Length: Number of address bytes issued during page address operation **NOTE:** Bytes for column address are determined by page size 00 1 byte 01 2 bytes 10 3 bytes 11 4 bytes
9–10 -	This field is reserved.
11–12 PGS	Page Size 00 512 Bytes 01 2 KB 10 4 KB 11 8 KB
13–15 -	This field is reserved.
16–18 SPRZ	Spare size **NOTE:** Depending on the ECC mode and page size, a fixed value of the spare region is selected during boot. For more information, see Booting methods. Others Reserved

Table continues on the next page...

IFC_CSOR*n*_NAND field descriptions (continued)

Field	Description
	000 16 Bytes 001 64 Bytes 010 128 Bytes 011 210 Bytes 100 218 Bytes 101 224 Bytes 110 Spare size information will be used from corresponding CSORn_EXT register.
19–20 -	This field is reserved.
21–23 PB	Pages per Block Others Reserved 000 32 Pages 001 64 Pages 010 128 Pages 011 256 Pages
24–26 -	This field is reserved.
27–29 TRHZ	Time for read enable high to output high impedance (Z). Number of clocks required for memory to go in high-Z after read enable deassertion. This field is used during last read data access. If the IFC is accessing the NAND flash for a read operation, then after the last byte is read the NAND FSM must wait for TRHZ clock cycles so that there is no contention on the external buffer. Other settings are Reserved. 000 Wait for 20 IP Clocks 001 Wait for 40 IP Clocks 010 Wait for 60 IP Clocks 011 Wait for 80 IP Clocks 100 Wait for 100 IP Clocks
30 -	This field is reserved.
31 BCTLD	Buffer control disable. Disables assertion of BCTL during access to the current memory bank. 0 BCTL is asserted upon access to the current memory bank. 1 BCTL is not asserted upon access to the current memory bank.

13.3.6 Chip-Select Option Register - NOR Flash Mode (IFC_CSOR*n*_NOR)

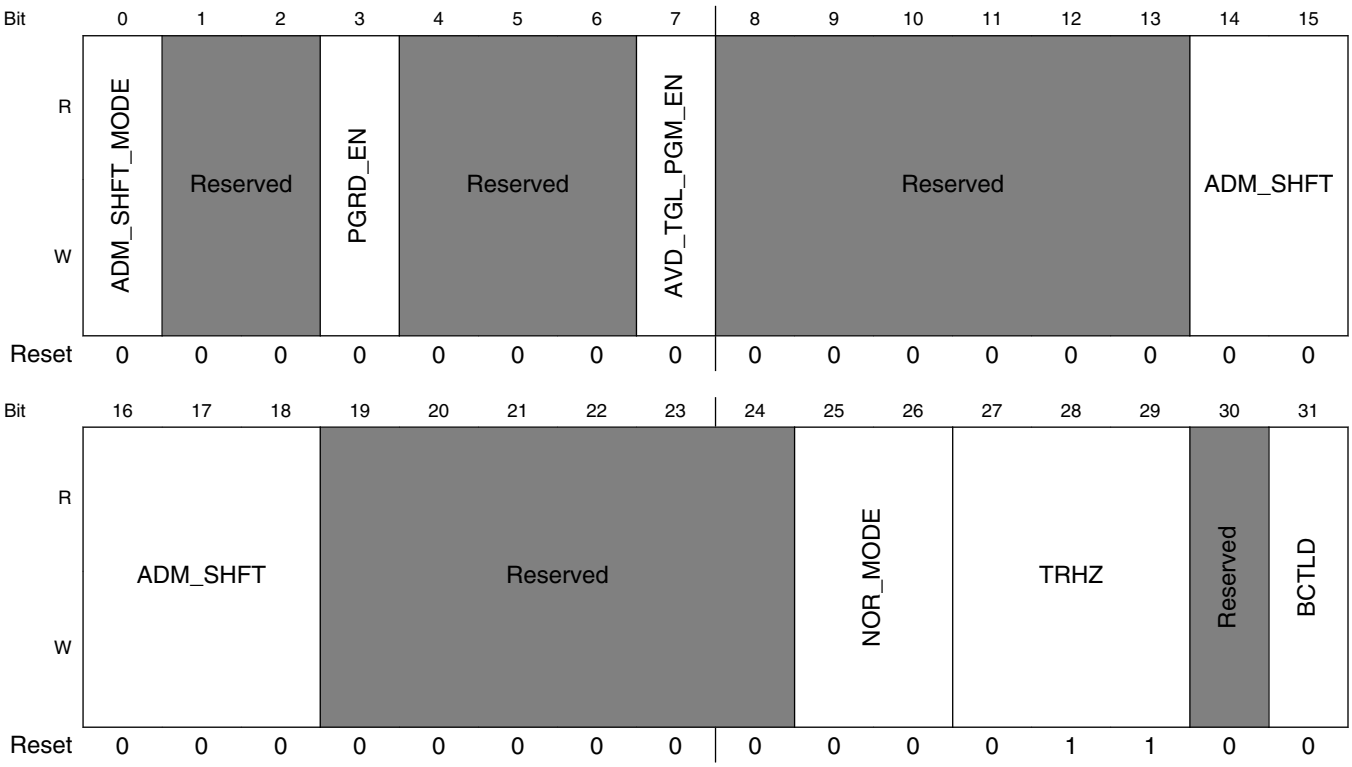
The following shows the CSOR*n* fields when CSPR*n*[MSEL] selects the NOR flash mode.

NOTE

- CSOR0 is the only chip-select used for booting.

- ADM_SHFT_MODE/ADM_SHFT is obtained from the configuration word when boot_load/rcw_load occurs.
- The reset value of TRHZ is programmable by passing the parameter.

Address: 12_4000h base + 130h offset + (12d × i), where i=0d to 7d



IFC_CSOR*n*_NOR field descriptions

Field	Description
0 ADM_SHFT_MODE	Address shift mode **NOTE:** Details of shifting is given in Mode 0 pin muxing (CSOR*n*[ADM_SHFT_MODE] = 0) . 0 Address msbs will be assigned to AD bus and ADDR bus carries the lsb 1 AD bus will carry lsbs and ADDR bus carries the msb
1–2 -	This field is reserved.
3 PGRD_EN	Page read enable from NOR device **NOTE:** This feature is not supported when ADM_SHFT_MODE is 1, as in this mode, the device captures the lsbs from the AD bus. Thus, in order for the device to distinguish a data phase from an address phase, it would require that AVD/ALE be asserted for every address phase of the burst operation. 0 A multi-beat read transaction received from system bus will be split into per-beat accesses (based on NOR port size). 1 A multi-beat read transaction received from system bus will be performed as a single-page read operation (burst type) on NOR flash.

Table continues on the next page...

IFC_CSOR*n*_NOR field descriptions (continued)

Field	Description
4–6 -	This field is reserved.
7 AVD_TGL_ PGM_EN	AVD toggle enable during burst program 0 Assert ALE/AVD only for the first address phase (performed on the flash interface) of a burst write operation received from the system interface 1 Assert ALE/AVD during every subsequent address phase (performed on the flash interface) including the first phase of a burst write operation received from the system interface
8–13 -	This field is reserved.
14–18 ADM_SHFT	Address data multiplexing shift. It controls the way internal 32 bit address is placed on the external bus, during NOR/GPCM address phase. Address shifting will be done by 1 bit. Patterns not shown are reserved. 00000 No shift 00001 shift ifc_addr by 1 00010 shift ifc_addr by 2 00011 shift ifc_addr by 3 00100 shift ifc_addr by 4 00101 shift ifc_addr by 5 00110 shift ifc_addr by 6 00111 shift ifc_addr by 7 01000 shift ifc_addr by 8 01001 shift ifc_addr by 9 01010 shift ifc_addr by 10 01011 shift ifc_addr by 11 01100 shift ifc_addr by 12 01101 shift ifc_addr by 13 01110 shift ifc_addr by 14 01111 shift ifc_addr by 15 10000 shift ifc_addr by 16 10001 shift ifc_addr by 17 10010 shift ifc_addr by 18 10011 shift ifc_addr by 19 10100 shift ifc_addr by 20
19–24 -	This field is reserved.
25–26 NOR_MODE	Type of the NOR device hooked at CS*n*. 00 Simple asynchronous NOR (ALE is asserted before CE_B) 01 Internal latch based AVD NOR device (ALE is asserted after CE_B) Others Reserved
27–29 TRHZ	Time for read enable high to output high impedance (Z). Number of clocks required for memory to go in high Z after read enable deassertion. If IFC is accessing NOR flash for read operation, then after last byte read NOR FSM must wait for these many clock cycles so that there is no contention on external buffer. Others Reserved

Table continues on the next page...

IFC_CSOR*n*_NOR field descriptions (continued)

Field	Description
	000 Wait for 20 IP Clocks 001 Wait for 40 IP Clocks 010 Wait for 60 IP Clocks 011 Wait for 80 IP Clocks 100 Wait for 100 IP Clocks
30 -	This field is reserved.
31 BCTLD	Buffer control disable: Disables assertion of BCTL during access to the current memory bank. 0 BCTL is asserted upon access to the current memory bank 1 BCTL is not asserted upon access to the current memory bank

13.3.7 Chip-Select Option Register - GPCM (IFC_CSOR*n*_GPCM)

The CSCOR*n* can work in following two modes:

- Normal GPCM
- Generic ASIC

The following shows the CSOR*n* fields when CSPR*n*[MSEL] selects the GPCM mode.

Address: 12_4000h base + 130h offset + (12d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	GPMODE	PAR	PAR_EN	Reserved	GPTO				Reserved			ABRT_RSP_EN	RGETA	WGETA	ADM_SHFT	
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	ADM_SHFT			Reserved	BURST_LEN			GAPERRD	Reserved			TRHZ			Reserved	BCTLD
Reset	0	0	0	0	0	0	0	0	0	0	0	n*	n*	n*	0	0

* Notes:

- TRHZ field: TRHZ – Reset value is programmable by passing the parameter.

IFC_CSOR*n*_GPCM field descriptions

Field	Description
0 GPMODE	GPCM Mode of operation: 0 Normal GPCM operation 1 Generic ASIC mode operation
1 PAR	Parity mode. 0 Odd Parity 1 Even Parity
2 PAR_EN	Parity checking enable/disable: 0 Parity checking disabled over the received data 1 Parity checking enabled over the received data
3 -	This field is reserved.
4–7 GPTO	GPCM Timeout Count: For normal GPCM, the timeout occur only when read data/write data transaction is external transaction acknowledgement based and IFCTA signal has not come. In Generic ASIC mode timeout occurs when RDY_L has not come for the number of IP clks cycles defined by this register field. 0000 256 cycles of IFC module input clocks 0001 512 cycles of IFC module input clocks 0010 1024 cycles of IFC module input clocks 0011 2048 cycles of IFC module input clocks 0100 4096 cycles of IFC module input clocks 0101 8192 cycles of IFC module input clocks 0110 16,384 cycles of IFC module input clocks 0111 32,768 cycles of IFC module input clocks 1000 65,536 cycles of IFC module input clocks 1001 131,072 cycles of IFC module input clocks 1010 262,144 cycles of IFC module input clocks 1011 524,288 cycles of IFC module input clocks 1100 1,048,576 cycles of IFC module input clocks 1101 2,097,152 cycles of IFC module input clocks 1110 4,194,304 cycles of IFC module input clocks 1111 8,388,608 cycles of IFC module input clocks
8–10 -	This field is reserved.
11 ABRT_RSP_EN	Abort Error Response Enable. A error response will be sent for AXI read transaction if transaction is aborted by IFCTA_B when REGTA is programmed in abort mode. This bit is valid only for normal GPCM mode. 0 No error response will be sent on AXI. 1 Error response will be sent for abort to AXI.
12 RGETA	GPCM external access termination mode for read access. **NOTE:** This bit is valid only for normal GPCM Mode.

Table continues on the next page...

IFC_CSOR*n*_GPCM field descriptions (continued)

Field	Description
	0 Abort mode. IFCTA_B signal acts as abort signal. GPCM read access is terminated internally by the controller at the expiry of TRAD counter unless aborted externally (If IFCTA_B asserts before the expiry of TRAD counter access will be terminated). Error will be reported in GPCM_EVTER_STAT[ABER] register. 1 Acknowledgement mode. IFCTA_B acts as acknowledgement/data qualifier signal. GPCM read access is acknowledged by external pin IFCTA_B and only it can complete the read access. If it is not asserted within CSOR[GPTO] time, GPCM_EVTER_STAT[TOER] will be set.
13 WGETA	GPCM external access termination mode for write access: **NOTE:** This bit is valid only for normal GPCM Mode. 0 Abort mode. GPCM write access is terminated if IFCTA_B comes (asserted) before expiry of TWP counter in case of single beat transaction. If IFCTA_B is asserted during burst write the current transaction is aborted and only after deassertion of IFCTA_B it will be resumed. Details are given in Normal GPCM program operation, Figure 13-269. Note that in abort mode no error is reported for write transaction. 1 Acknowledgement mode: GPCM write access is acknowledged/qualified by external pin IFCTA_B assertion. If it is not asserted within CSOR[GPTO] time, GPCM_EVTER_STAT[TOER] will be set
14–18 ADM_SHFT	Address data multiplexing shift: Left shift the flash address by this value and assign it to address data multiplexed bus (AD[0:15]) ifc_data. By this method, the address msbs will be assigned to address data muxed bus that can be latched by asserting the AVD/ALE signal. **NOTE:** This shifting is only valid in normal GPCM mode. Others Reserved 00000 No shift 00001 Left shift addr[0:31] by 1 and assign it to AD[0:15] 00010 Left shift ifc_addr by 2 00011 Left shift ifc_addr by 3 00100 Left shift ifc_addr by 4 00101 Left shift ifc_addr by 5 00110 Left shift ifc_addr by 6 00111 Left shift ifc_addr by 7 01000 Left shift ifc_addr by 8 01001 Left shift ifc_addr by 9 01010 Left shift ifc_addr by 10 01011 Left shift ifc_addr by 11 01100 Left shift ifc_addr by 12 01101 Left shift ifc_addr by 13 01110 Left shift ifc_addr by 14 01111 Left shift ifc_addr by 15 10000 Left shift ifc_addr by 16 10001 Left shift ifc_addr by 17 10010 Left shift ifc_addr by 18 10011 Left shift ifc_addr by 19 10100 Left shift ifc_addr by 20

Table continues on the next page...

IFC_CSOR*n*_GPCM field descriptions (continued)

Field	Description
19 -	This field is reserved.
20–22 BURST_LEN	GPCM burst length. It defines the maximum number of beats that will be sent/received in one burst cycle. This is programmed in terms of port-size transfer. For example, if the port size is 2 bytes and burst_length is 2, then the total of 8 bytes will be transferred in one burst cycle (that is, 4 beats of data transfers and each data transfer of 2 bytes). This is valid only in normal GPCM mode. 0 Non-burst mode 1 2 2 4 3 8 4 16 5 32 6 64 7 128
23–24 GAPERRD	Generic ASIC parity error indication delay. Represents the delay (in terms of number of ifc_clk) between the indication of parity error for address and write data with respect to start of frame (SOF_L). 0 1 ifc_clk delayed 1 2 ifc_clk delayed 2 3 ifc_clk delayed 3 4 ifc_clk delayed
25–26 -	This field is reserved.
27–29 TRHZ	Time for read enable high to output high impedance (Z)- Number of clocks required for memory to go in high Z after read enable deassertion. If IFC is accessing for read operation, then after last byte read wait for these many clock cycles so that there is no contention on external bus. Others Reserved 000 Wait for 20 IFC module input clocks 001 Wait for 40 IFC module input clocks 010 Wait for 60 IFC module input clocks 011 Wait for 80 IFC module input clocks 100 Wait for 100 IFC module input clocks
30 -	This field is reserved.
31 BCTLD	Buffer control disable: Disables assertion of BCTL during access to the current memory bank. 0 BCTL is asserted upon access to the current memory bank. 1 BCTL is not asserted upon access to the current memory bank.

13.3.8 Extended Chip Select Option Register - NAND Flash Mode (IFC_CSOR*n*_EXT)

The figure below shows the CSOR*n*_EXT fields when CSPR*n*[MSEL] selects the NAND flash mode.

NOTE

CSOR*n*_EXT register is reserved if CSPR*n*[MSEL] is in NOR/ GPCM mode.

Address: 12_4000h base + 134h offset + (12d × 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	Reserved																					SPARE_BYTES_CSn										
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IFC_CSOR*n*_EXT field descriptions

Field	Description
0–20 -	This field is reserved.
21–31 SPARE_BYTES_CSn	No. of bytes in spare region. This filed is applicable only when corresponding CSOR*n*[SPRZ] is 3'h110 Others Reserved. 0x000 0 Spare region bytes 0x001 1 Spare region byte 0x400 1024 spare region bytes

13.3.9 Flash Timing Register 0 for Chip-Select *n* - NAND Flash Mode (IFC_FTIM0_CS*n*_NAND)

The flash timing registers define the flash interface timings. These register fields are defined differently depending on the machine type (NOR/NAND/GPCM) selected for that bank by CSPR*n*[MSEL] field. Timing registers are provided separately for each chip-select. Timing parameter are in terms of IFC module input clock cycles. More details about the register field is given in section Programming model for flash interface timing .

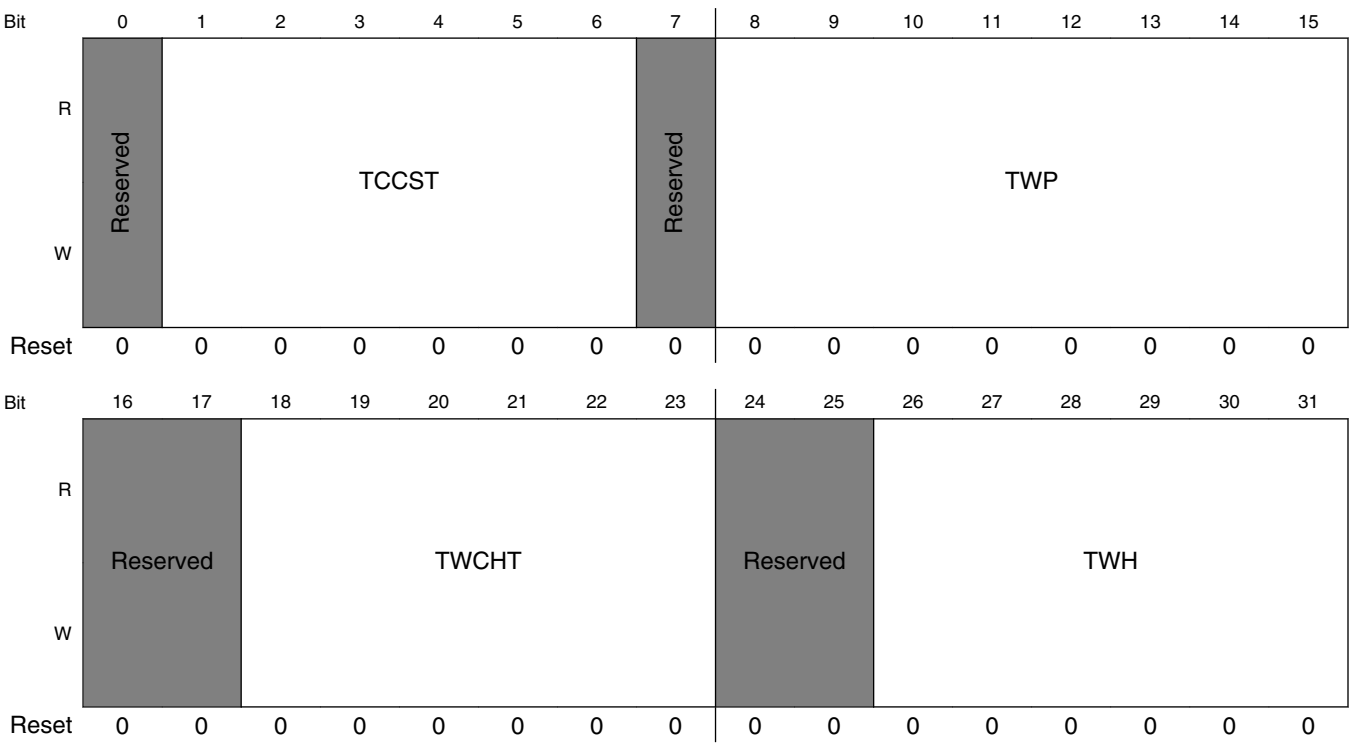
NOTE

- Timing values of FTIM0, FTIM1, FTIM2, and FTIM3 registers for chip-select 0 in NAND/NOR flash mode will be loaded at boot_load/rcw_load by the parameters passed.
- The TRP value loaded duringRCW/ Boot load is taken as (TRAD +2) input clocks.

- For normal GPCM mode (write transaction) all the three timing parameters (that is, TEAHC, TACSE, and TCS) should not be programmed zero together.
- For normal GPCM mode (read transaction) all the three timing parameters (that is, TEAHC, TACSE, and TACO) should not be programmed zero together.

The following figure shows the FTIM0_CS*n* fields when CSPR*n*[MSEL] selects the NAND flash mode.

Address: 12_4000h base + 1C0h offset + (48d × i), where i=0d to 7d



IFC_FTIM0_CS*n*_NAND field descriptions

Field	Description
0 -	This field is reserved.
1–6 TCCST	Chip enable (CE) to command latch enable (CLE) setup time 000000 Reserved 000001 1 IFC module input clock 000010 2 IFC module input clocks ... 111111 63 IFC module input clocks
7 -	This field is reserved.

Table continues on the next page...

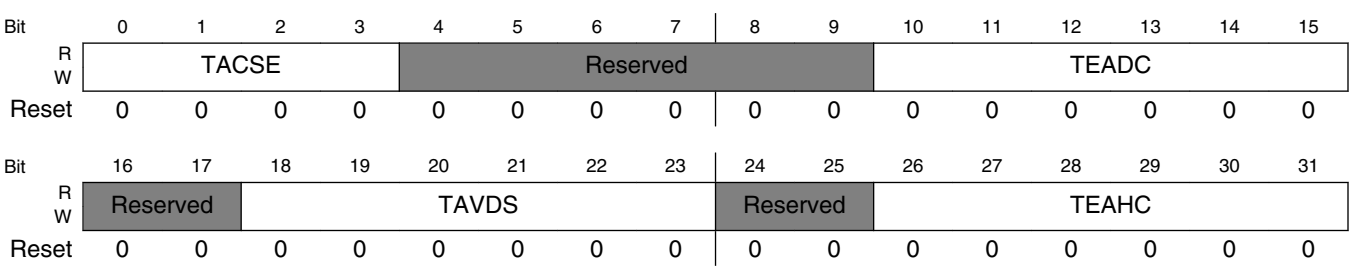
IFC_FTIM0_CS*n*_NAND field descriptions (continued)

Field	Description
8–15 TWP	Write enable (WE) pulse width 00000000 Reserved 00000001 1 IFC module input clock 00000010 2 IFC module input clocks ... 11111111 255 IFC module input clocks
16–17 -	This field is reserved.
18–23 TWCHT	WE to command hold time 000000 Reserved 000001 1 IFC module input clock 000010 2 IFC module input clocks ... 111111 63 IFC module input clocks
24–25 -	This field is reserved.
26–31 TWH	WE high hold time 000000 Reserved 000001 1 IFC module input clock 000010 2 IFC module input clocks ... 111111 63 IFC module input clocks

13.3.10 Flash Timing Register 0 for CS*n* - NOR Flash Mode (IFC_FTIM0_CS*n*_NOR)

The following shows the FTIM0_CS*n* fields when CSPR*n*[MSEL] selects the NOR flash mode.

Address: 12_4000h base + 1C0h offset + (48d × i), where i=0d to 7d



IFC_FTIM0_CS*n*_NOR field descriptions

Field	Description
0–3 TACSE	Address phase (after external latch enable deassertion) end to chip enable assertion time 0000 Reserved 0001 1 IFC module input clock 0010 2 IFC module input clocks ... 1111 15 IFC module input clocks
4–9 -	This field is reserved.
10–15 TEADC	External latch address delay cycles: External address valid (AVD) assertion time (pulse width of external latch enable signal). 000000 Reserved 000001 1 IFC module input clock 000010 2 IFC module input clocks ... 111111 63 IFC module input clocks
16–17 -	This field is reserved.
18–23 TAVDS	Delay between CS assertion to AVD/ALE assertion in AVD devices (address latch internal to the device) 000000 Reserved 000001 1 IFC module input clock 000010 2 IFC module input clocks ... 111111 63 IFC module input clocks
24–25 -	This field is reserved.
26–31 TEAHC	External latch address hold cycles: Address hold cycle relative to external latch enable signal 000000 Reserved 000001 1 IFC module input clock 000010 2 IFC module input clocks ... 111111 63 IFC module input clocks

13.3.11 Flash Timing Register 0 for CS*n* - Normal GPCM Mode (IFC_FTIM0_CS*n*_GPCM)

The following shows the FTIM0_CS*n* fields when CSPR*n*[MSEL] selects the GPCM mode.

Address: 12_4000h base + 1C0h offset + (48d × 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	TACSE				Reserved						TEADC						Reserved										TEAHC					
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

IFC_FTIM0_CS*n*_GPCM field descriptions

Field	Description
0–3 TACSE	Address phase (After address hold cycle) end to chip enable assertion time 0000 Reserved 0001 1 IFC module input clock 0010 2 IFC module input clocks ... 1111 15 IFC module input clocks
4–9 -	This field is reserved.
10–15 TEADC	External latch address delay cycles. External address latch enable (ALE) assertion time (pulse width of external latch enable signal) 000000 Reserved 000001 1 IFC module input clock 000010 2 IFC module input clocks ... 111111 63 IFC module input clocks
16–25 -	This field is reserved.
26–31 TEAHC	External latch address hold cycles: Address hold cycle relative to external latch enable signal 000000 Reserved 000001 1 IFC module input clock 000010 2 IFC module input clocks ... 111111 63 IFC module input clocks

13.3.12 Flash Timing Register 1 for Chip-Select n - NAND Flash Mode (IFC_FTIM1_CS*n*_NAND)

The following shows the FTIM1_CS*n* fields when CSPR*n*[MSEL] selects the NAND flash mode.

Address: 12_4000h base + 1C4h offset + (48d × i), where i=0d to 7d

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

IFC_FTIM1_CS*n*_NAND field descriptions

Field	Description
0–7 TADLE	Effective address to data loading time 00000000 Reserved 00000001 1 IFC module input clock 00000010 2 IFC module input clocks ... 11111111 255 IFC module input clocks
8–15 TWBE	WE high to ready busy (RB_B) low time 00000000 Reserved 00000001 1 IFC module input clock 00000010 2 IFC module input clocks ... 11111111 255 IFC module input clocks
16–17 -	This field is reserved.
18–23 TRR	Ready busy high to read enable (RE) low time 000000 Reserved 000001 1 IFC module input clock 000010 2 IFC module input clocks ... 111111 63 IFC module input clocks
24–31 TRP	RE pulse width 00000000 Reserved 00000001 1 IFC module input clock

Table continues on the next page...

IFC_FTIM1_CS*n*_NAND field descriptions (continued)

Field	Description
	00000010 2 IFC module input clocks ... 11111111 255 IFC module input clocks

13.3.13 Flash Timing Register 1 for CS*n* - NOR Flash Mode (IFC_FTIM1_CS*n*_NOR)

The following shows the FTIM1_CS*n* fields when BR*n*[MSEL] selects the NOR flash mode.

Address: 12_4000h base + 1C4h offset + (48d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	TACO								-							
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	-		TRAD_NOR						Reserved		TSEQRAD_NOR					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IFC_FTIM1_CS*n*_NOR field descriptions

Field	Description
0–7 TACO	CS assertion to output enable (OE) assertion setup time 00000000 Reserved 00000001 1 IFC module input clock 00000010 2 IFC module input clocks ... 11111111 255 IFC module input clocks
8–17 -	
18–23 TRAD_NOR	NOR flash read access delay: It represents the read enable to data access time plus total round trip board delay from the external NOR flash memory during read operation. Its value can be calculated as *[TOEmax + 2*Board Delay + Device Setup Time]*. 000000 Reserved 000001 1 IFC module input clock 000010 2 IFC module input clock 111111 63 IFC module input clocks
24–25 -	This field is reserved.

Table continues on the next page...

IFC_FTIM1_CS*n*_NOR field descriptions (continued)

Field	Description
26–31 TSEQRAD_NOR	NOR flash sequential read access delay. It represents the address to data access time plus total round trip delay from the external NOR flash memory during sequential read operation. Its value can be calculated as *[NOR Page Address Delay (max) for sequential read + 2*Board Delay + Device Setup Time]*. 000000 Reserved 000001 1 IFC module input clock ... 111111 63 IFC module input clocks

13.3.14 Flash Timing Register 1 for CS*n* - Normal GPCM Mode (IFC_FTIM1_CS*n*_GPCM)

The following shows the FTIM1_CS*n* fields when BR*n*[MSEL] and CSOP*n*[GPMODE] selects the normal GPCM mode.

Address: 12_4000h base + 1C4h offset + (48d × 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	TACO								Reserved								TRAD								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

IFC_FTIM1_CS*n*_GPCM field descriptions

Field	Description
0–7 TACO	CS assertion to output enable (OE) assertion setup time 00000000 Reserved 00000001 1 IFC module input clock 00000010 2 IFC module input clocks ... 11111111 255 IFC module input clock
8–17 -	This field is reserved.
18–23 TRAD	GPCM read access delay: It represents the output enable assertion time. See Normal GPCM read operation . 000000 Reserved 000001 1 IFC module input clock 000010 2 IFC module input clocks ... 111111 63 IFC module input clocks
24–31 -	This field is reserved.

13.3.15 Flash Timing Register 2 for Chip-Select n - NAND Flash Mode (IFC_FTIM2_CS*n*_NAND)

The following shows the FTIM2_CS*n* fields when CSPR*n*[MSEL] selects the NAND flash mode.

Address: 12_4000h base + 1C8h offset + (48d × 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	Reserved			TRAD								Reserved					TREH					Reserved			TWHRE							
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

IFC_FTIM2_CS*n*_NAND field descriptions

Field	Description
0–2 -	This field is reserved.
3–10 TRAD	Flash read access delay: It represents the data sampling time of read data. It should be programmed such that sampling is done at the center of the received data eye. For calculation of data eye width and appropriate data sampling time refer to NAND asynchronous mode calculating read data window width . 00000000 Reserved 00000001 1 IFC module input clock 00000010 2 IFC module input clocks ... 11111111 255 IFC module input clocks
11–14 -	This field is reserved.
15–20 TREH	RE_B High Time 000000 Reserved 000001 1 IFC module input clock 000010 2 IFC module input clocks ... 111111 63 IFC module input clocks
21–23 -	This field is reserved.
24–31 TWHRE	WE_B high to RE_B low effective 00000000 Reserved 00000001 1 IFC module input clock 00000010 2 IFC module input clocks ... 11111111 255 IFC module input clocks

13.3.16 Flash Timing Register 2 for CS*n* - NOR Flash Mode (IFC_FTIM2_CS*n*_NOR)

The following shows the FTIM2_CS*n* fields when BR*n*[MSEL] selects the NOR flash mode.

Address: 12_4000h base + 1C8h offset + (48d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Reserved				TCS				Reserved		TCH				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	TWPH						Reserved		TWP							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IFC_FTIM2_CS*n*_NOR field descriptions

Field	Description
0–3 -	This field is reserved.
4–7 TCS	Chip-select assertion to WE assertion setup time 0000 Reserved 0001 1 IFC module input clock 0010 2 IFC module input clocks ... 1111 15 IFC module input clocks
8–9 -	This field is reserved.
10–13 TCH	Chip-select hold time with respect to WE deassertion 0000 Reserved 0001 1 IFC module input clock 0010 2 IFC module input clocks ... 1111 15 IFC module input clocks
14–15 -	This field is reserved.
16–21 TWPH	Write enable pulse high time 000000 Reserved 000001 1 IFC module input clock 000010 2 IFC module input clocks ... 111111 63 IFC module input clocks

Table continues on the next page...

IFC_FTIM2_CS*n*_NOR field descriptions (continued)

Field	Description
22–23 -	This field is reserved.
24–31 TWP	Write enable pulse width 00000000 Reserved 00000001 1 IFC module input clock 00000010 2 IFC module input clocks ... 11111111 255 IFC module input clocks

13.3.17 Flash Timing Register 2 for CS*n* - Normal GPCM Mode (IFC_FTIM2_CS*n*_GPCM)

The following shows the FTIM2_CS*n* fields when BR*n*[MSEL] selects the normal GPCM mode.

Address: 12_4000h base + 1C8h offset + (48d × i), where i=0d to 7d

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

IFC_FTIM2_CS*n*_GPCM field descriptions

Field	Description
0–3 -	This field is reserved.
4–7 TCS	Chip-select assertion to WE assertion setup time 0000 Reserved 0001 1 IFC module input clock 0010 2 IFC module input clocks ... 1111 15 IFC module input clocks
8–9 -	This field is reserved.
10–13 TCH	Chip-select hold time with respect to WE deassertion 0000 Reserved

Table continues on the next page...

IFC_FTIM2_CS*n*_GPCM field descriptions (continued)

Field	Description
	0001 1 IFC module input clock 0010 2 IFC module input clocks ... 1111 15 IFC module input clocks
14–23 -	This field is reserved.
24–31 TWP	Write enable pulse width 00000000 Reserved 00000001 1 IFC module input clock 00000010 2 IFC module input clocks ... 11111111 255 IFC module input clock

13.3.18 Flash Timing Register 3 for Chip-Select n - NAND Flash Mode (IFC_FTIM3_CS*n*_NAND)

The following shows the FTIM3_CS*n* fields when CSPR*n*[MSEL] selects the NAND flash mode.

Address: 12_4000h base + 1CCh offset + (48d × 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	TWW								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

IFC_FTIM3_CS*n*_NAND field descriptions

Field	Description
0–7 TWW	Write protect transition to chip-select assertion Refer Write protect for more details. 00000000 Reserved 00000001 1 IFC module input clock 00000010 2 IFC module input clocks ... 11111111 255 IFC module input clocks
8–31 -	This field is reserved.

13.3.19 Flash Timing Register 3 for CS*n* - NOR Flash Mode (IFC_FTIM3_CS_NOR)

The following shows the FTIM3_CS*n* fields when CSPR*n*[MSEL] selects the NOR flash mode.

NOTE

Offset for IFC_FTIM3_CSn_NOR are 0x0_01CC, 0x0_01FC, 0x0_022C, 0x0_025C, 0x0_028C, 0x0_02BC, 0x0_02EC, and 0x0_031C

Address: 12_4000h base + 1CCh offset = 12_41CCh

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																															
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

IFC_FTIM3_CS_NOR field descriptions

Field	Description
0–31 -	This field is reserved.

13.3.20 Flash Timing Register 3 for CS*n* - Normal GPCM Mode (IFC_FTIM3_CS*n*_GPCM)

The following shows the FTIM3_CS*n* fields when CSPR*n*[MSEL] selects the GPCM flash mode.

Address: 12_4000h base + 1CCh offset + (48d × 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	TAAD						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

IFC_FTIM3_CS*n*_GPCM field descriptions

Field	Description
0–5 TAAD	GPCM address access delay. This timing parameter is required for read cycles when GPCM is working in burst mode, that is, burst length is programmed to a non-zero value. The first data beat of a burst is sampled after TRAD time of OE assertion; subsequent beat data are sampled after TAAD time of the previous sample pulse. The same timing is used for sending the new address of the beat. The second address is sent after TRAD time of OE assertion and subsequent address beats are sent after TAAD time of the last beat. This is programmed in terms of IFC module input clock but should always be a mupltiple of ifc_clk. This is valid only for normal GPCM mode.

Table continues on the next page...

IFC_FTIM3_CS*n*_GPCM field descriptions (continued)

Field	Description
	6'h0 Reserved 6'h1 1 IFC module input clock 6'h2 2 IFC module input clocks ... 6'h3F 63 IFC module input clocks
6–31 -	This field is reserved.

13.3.21 Ready busy status for each chip-select (IFC_RB_STAT)

Address: 12_4000h base + 400h offset = 12_4400h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RB0	RB1	RB2	RB3	RB4	RB5	RB6	RB7	Reserved							
W																
Reset	n	n	n	n	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															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IFC_RB_STAT field descriptions

Field	Description
0 RB0	Ready Busy input from CS0 This register field represents the status of RB_B signal of CS0 sampled by IFC at every clock. 0 At least one of the device connected at CS0, CS2-CS7 is busy 1 All devices connected to CS0, CS2-CS7 are ready
1 RB1	Ready Busy input from CS1 This register field represents the status of RB_B signal of CS1 sampled by IFC at every clock. 0 Device connected at CS1 is driving RB_B signal as Busy 1 Device connected at CS1 is driving RB_B signal as Ready.
2 RB2	Ready Busy input from CS2 This register field represents the status of RB_B signal of CS2 sampled by IFC at every clock. 0 At least one of the device connected at CS0, CS2-CS7 is busy 1 All devices connected to CS0, CS2-CS7 are ready
3 RB3	Ready Busy input from CS3 This register field represents the status of RB_B signal of CS3 sampled by IFC at every clock.

Table continues on the next page...

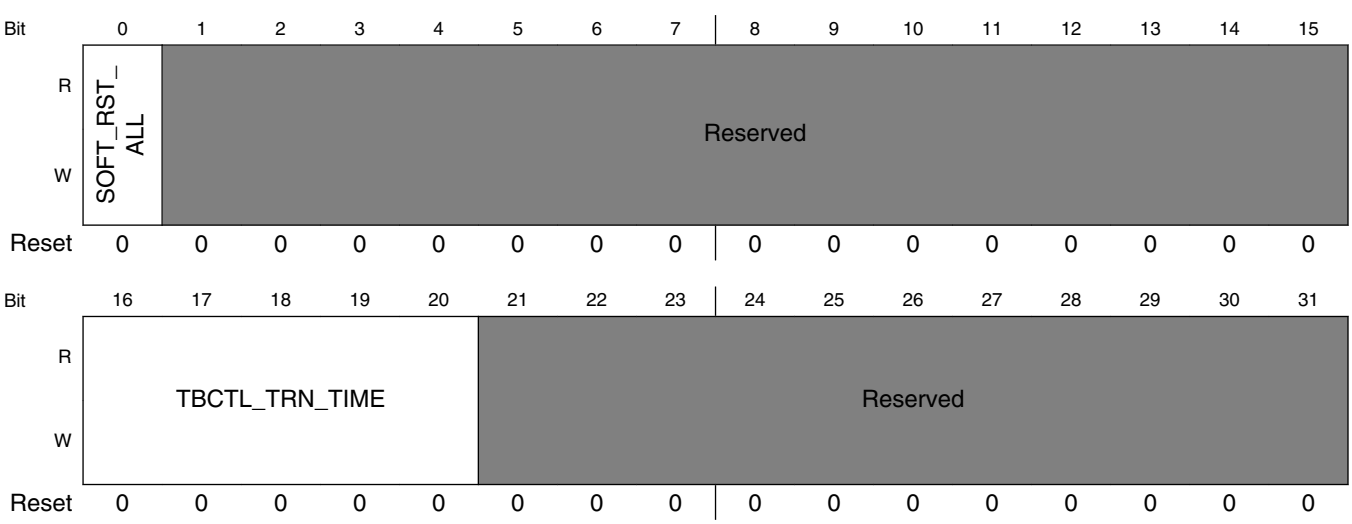
IFC_RB_STAT field descriptions (continued)

Field	Description
	0 At least one of the device connected at CS0, CS2-CS7 is busy 1 All devices connected to CS0, CS2-CS7 are ready
4 RB4	Ready Busy input from CS4 This register field represents the status of RB_B signal of CS4 sampled by IFC at every clock. 0 At least one of the device connected at CS0, CS2-CS7 is busy 1 All devices connected to CS0, CS2-CS7 are ready
5 RB5	Ready Busy input from CS5 This register field represents the status of RB_B signal of CS5 sampled by IFC at every clock. 0 At least one of the device connected at CS0, CS2-CS7 is busy 1 All devices connected to CS0, CS2-CS7 are ready
6 RB6	Ready Busy input from CS6 This register field represents the status of RB_B signal of CS6 sampled by IFC at every clock. 0 At least one of the device connected at CS0, CS2-CS7 is busy 1 All devices connected to CS0, CS2-CS7 are ready
7 RB7	Ready Busy input from CS7 This register field represents the status of RB_B signal of CS7 sampled by IFC at every clock. 0 At least one of the device connected at CS0, CS2-CS7 is busy 1 All devices connected to CS0, CS2-CS7 are ready
8–31 -	This field is reserved.

13.3.22 General control register (IFC_GCR)

The reset value of this field is passed through parameter.

Address: 12_4000h base + 40Ch offset = 12_440Ch



IFC_GCR field descriptions

Field	Description
0 SOFT_RST_ALL	Software Reset All: It is used to reset the whole IFC hardware (NAND, NOR, and GPCM). With this reset only the event/error status registers of IFC will be cleared. Programming registers will retain their value. The software reset mechanism is implemented such that IFC will send the pending transactions on system bus with error response. Once the whole IFC is in IDLE state, the software can clear this bit. IFC hardware stops software from clearing this bit till all the processing has been done and IFC hardware is in idle state. Software has to read this bit after performing write to this register bit. If the read value is not 0, it means hardware is still working. Write 0 will only be executed once it is in idle state after finishing all its operation on system side. 0 No Software reset 1 Assert Software reset
1–15 -	This field is reserved.
16–20 TBCTL_TRN_TIME	It represents the turnaround time of external buffer in terms of number of IFC module input clock cycles. BCTL should be kept asserted for these many clock cycles before issuing a write transaction after a read on flash interface.
21–31 -	This field is reserved.

13.3.23 Common event and error status register (IFC_CM_EVTER_STAT)

Common event and error status register (CM_EVTER_STAT) indicates the cause of an error or event that cannot be classified in NAND, NOR, and GPCM.

Address: 12_4000h base + 418h offset = 12_4418h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CSER	Reserved														
W	w1c															
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															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IFC_CM_EVTER_STAT field descriptions

Field	Description
0 CSER	Chip-Select Error

Table continues on the next page...

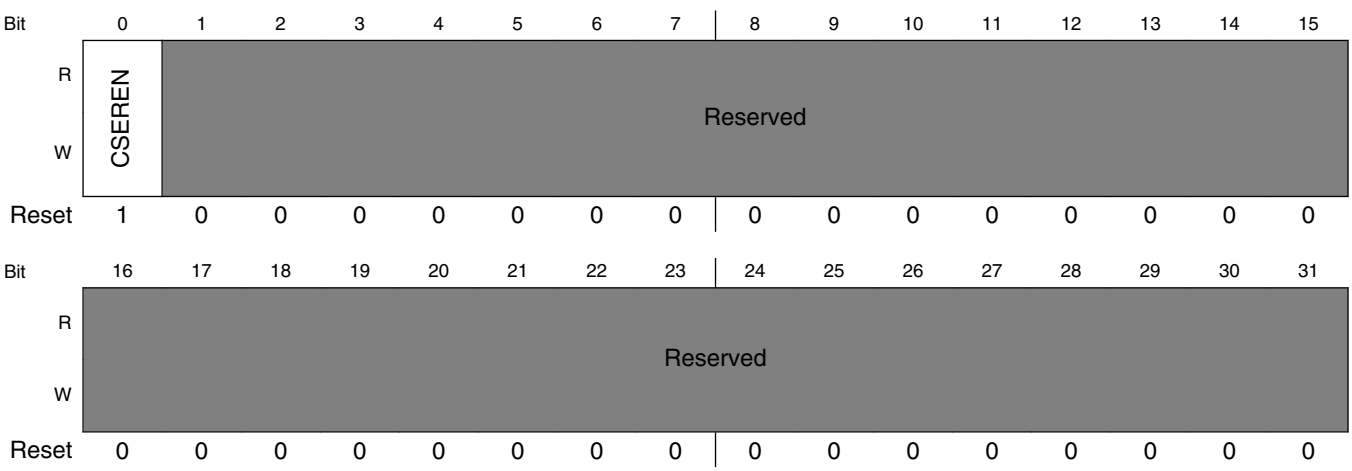
IFC_CM_EVTER_STAT field descriptions (continued)

Field	Description
	0 No chip-select error 1 A transaction was sent to IFC which is not mapped to any memory bank
1–31 -	This field is reserved.

13.3.24 Common event and error enable register (IFC_CM_EVTER_EN)

This register is used to enable/disable the logging of event and error indication in the CM_EVTER_STAT register.

Address: 12_4000h base + 424h offset = 12_4424h



IFC_CM_EVTER_EN field descriptions

Field	Description
0 CSEREN	Chip-Select Error Checking Enable 0 Chip-Select Error Checking Disabled 1 Chip-Select Error Checking Enabled
1–31 -	This field is reserved.

13.3.25 Common event and error interrupt enable register (IFC_CM_EVTER_INTR_EN)

Common event and error interrupt enable register (CM_EVTER_INTR_EN) is used to enable/disable the generation of interrupt signal corresponding to common error and event reporting. Software must clear pending events and errors in CM_EVTER_STAT register before enabling the interrupts.

Address: 12_4000h base + 430h offset = 12_4430h

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

IFC_CM_EVTER_INTR_EN field descriptions

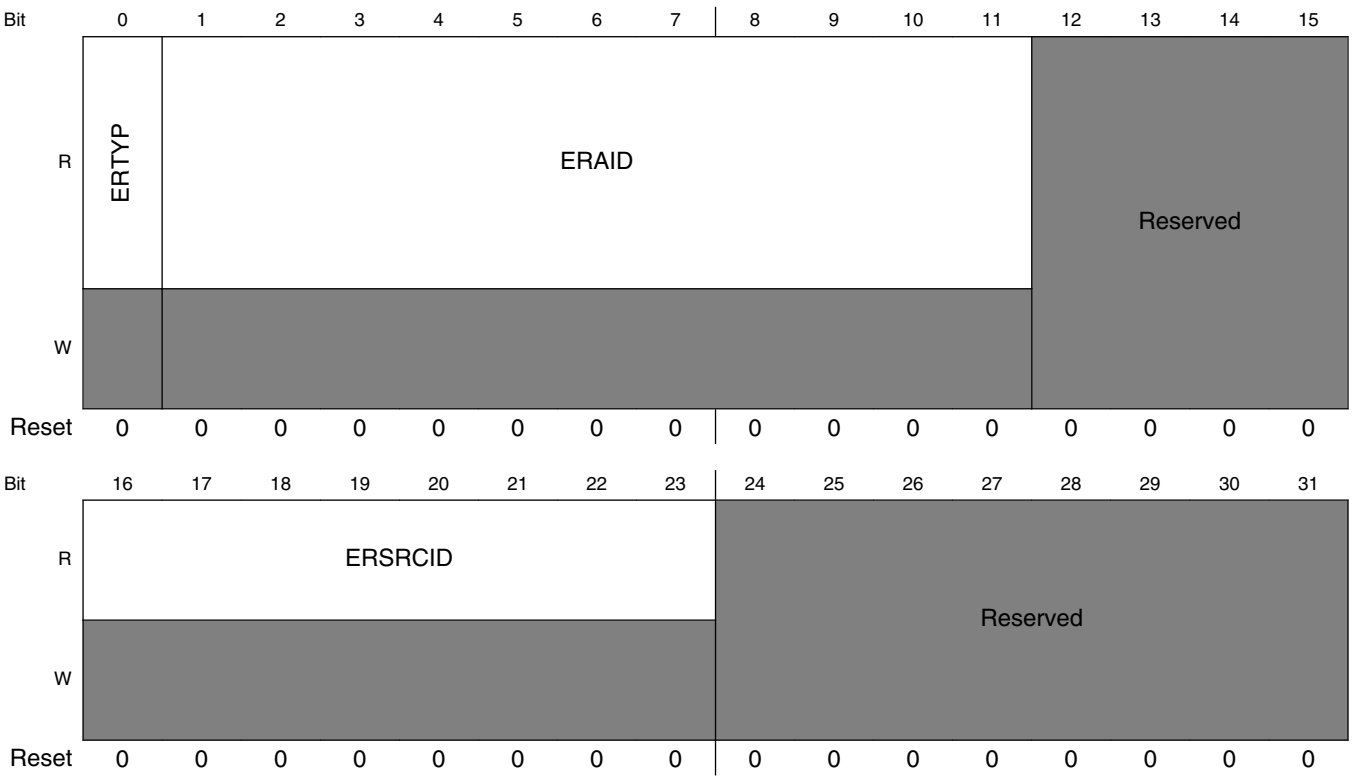
Field	Description
0 CSERIREN	Chip-Select Error Interrupt Enable 0 Chip-Select Error Interrupt Disabled 1 Chip-Select Error Interrupt Enabled
1–31 -	This field is reserved.

13.3.26 Common transfer error attributes register 0 (IFC_CM_ERATTR0)

These registers store the attribute corresponding to the first transaction on which common error occured.

Common transfer error attribute register 0 (CM_ERATTR0) is used to register the transaction attributes corresponding to first common error occured.

Address: 12_4000h base + 43Ch offset = 12_443Ch



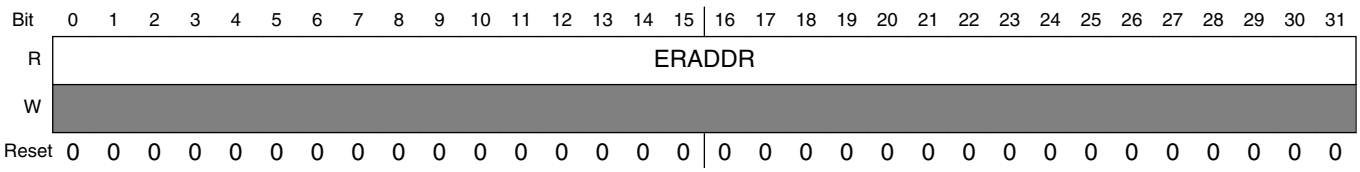
IFC_CM_ERATTR0 field descriptions

Field	Description
0 ERTYP	Transaction type of the error 0 Write Transaction 1 Read Transaction
1–11 ERAID	AXI ID of the error transaction
12–15 -	This field is reserved.
16–23 ERSRCID	Source ID of the error transaction, always zero.
24–31 -	This field is reserved.

13.3.27 Common transfer error attributes register 1 (IFC_CM_ERATTR1)

Common transfer error attribute register1 (CM_ERATTR1) is used to register the transaction address corresponding to first error occured.

Address: 12_4000h base + 440h offset = 12_4440h

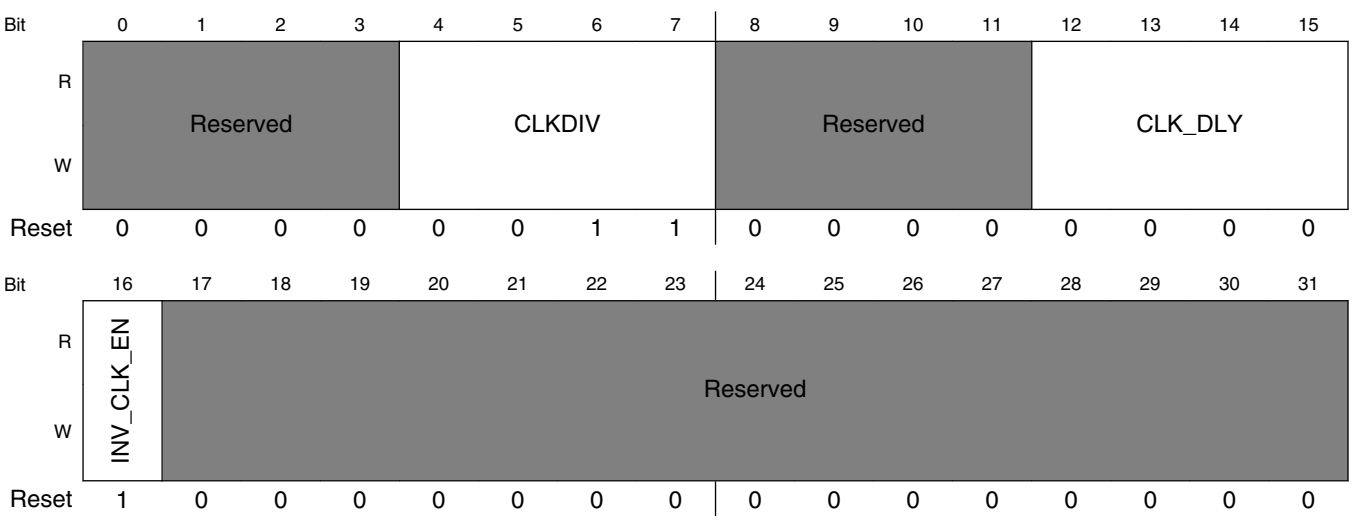


IFC_CM_ERATTR1 field descriptions

Field	Description
0–31 ERADDR	32 bits of transaction address corresponding to error transaction

13.3.28 Clock control register (IFC_CCR)

Address: 12_4000h base + 44Ch offset = 12_444Ch



IFC_CCR field descriptions

Field	Description
0–3 -	This field is reserved.

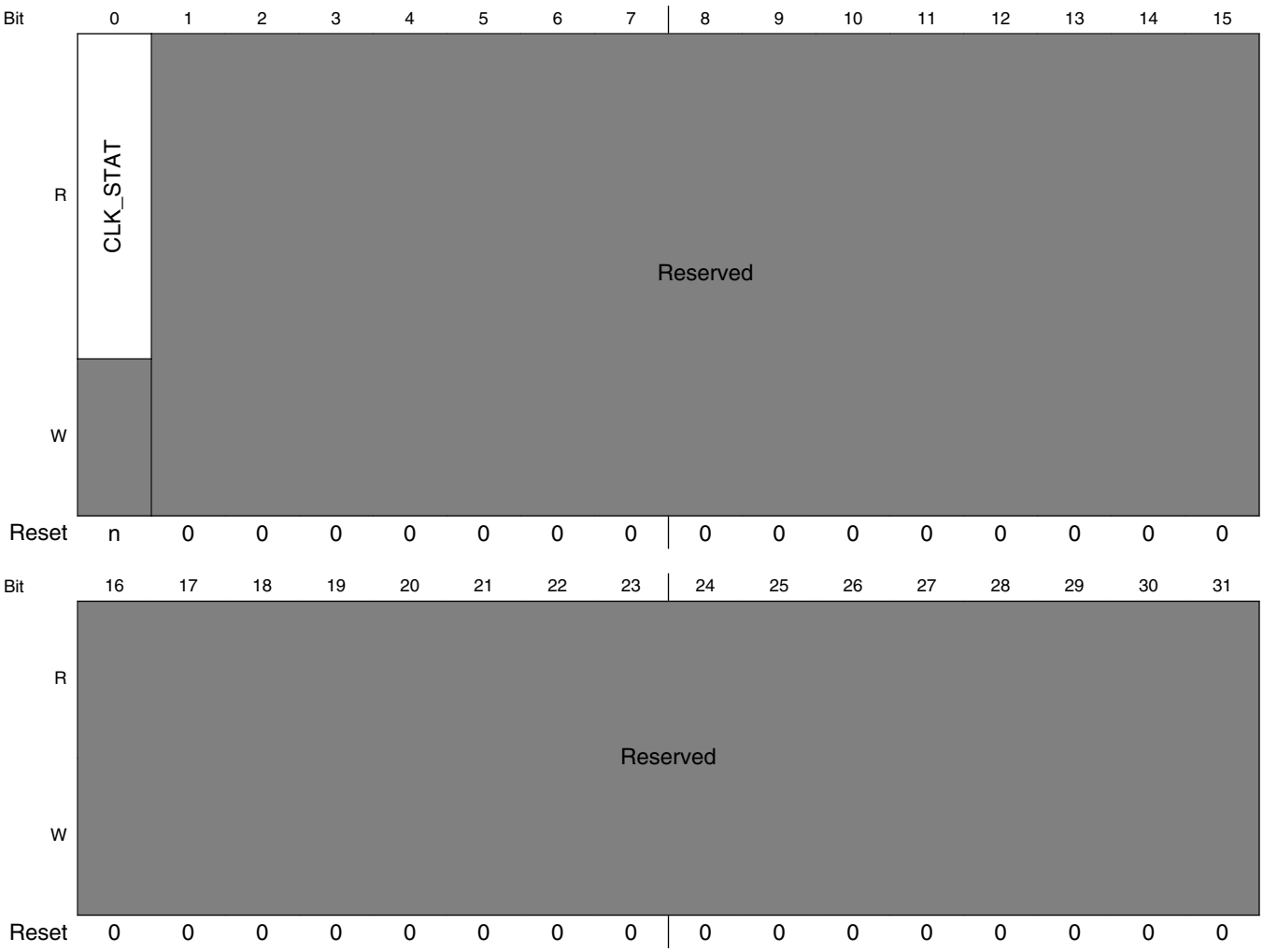
Table continues on the next page...

IFC_CCR field descriptions (continued)

Field	Description
4–7 CLKDIV	Clock division ratio: This field is used to generate the divided clock from the IFC module input clock. The divided clock is sent out after inverting. 0000 Reserved 0001 Divide by 2 0010 0011 Divide by 4 0100 Reserved 0101 0110 Reserved 0111 Divide by 8 1000 Reserved 1001 Reserved 1010 Reserved 1011 1100 Reserved 1101 Reserved 1110 Reserved 1111
8–11 -	This field is reserved.
12–15 CLK_DLY	IFC Clock Delay This field specifies the number of IFC module input clocks by which external clock is delayed. 0000 No delay 0001 1 IFC module input clocks delay introduced 0010 2 IFC module input clocks delay introduced ... FFFF 15 IFC module input clocks delay introduced
16 INV_CLK_EN	IFC Clock Inversion This field specifies whether IFC clock is inverted before sending out. 0 IFC clock is not inverted 1 IFC clock is inverted
17–31 -	This field is reserved.

13.3.29 Clock status register (IFC_CSR)

Address: 12_4000h base + 450h offset = 12_4450h



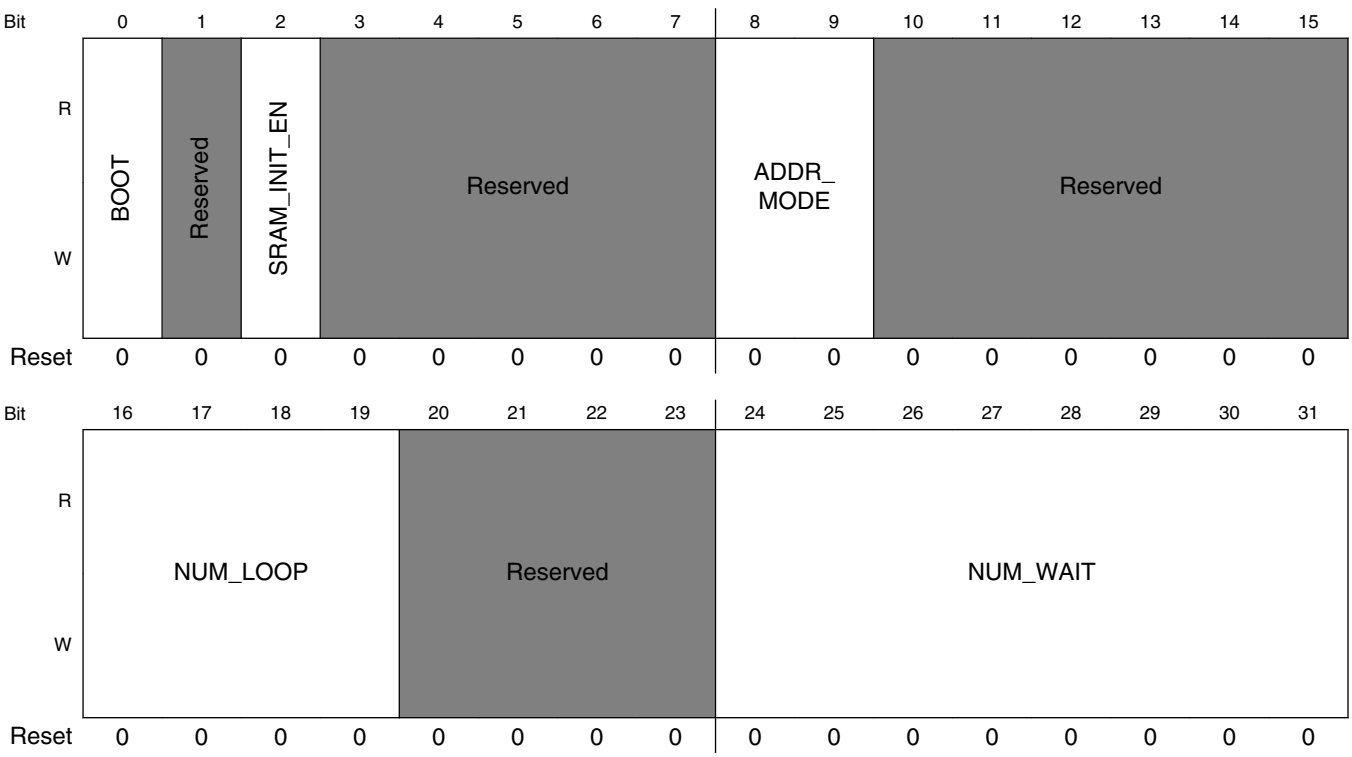
IFC_CSR field descriptions

Field	Description
0 CLK_STAT	Clock Status. This bit is cleared by hardware in case any of the field of CCR is changed. It is again set when external clock is stable. New GPCM operation can be initiated only after clock is stable for new CLK_DIV ratio. 0 Clock is unstable 1 Clock is stable
1–31 -	This field is reserved.

13.3.30 NAND configuration register (IFC_NCFGR)

A separate 1 KByte memory-mapped region is assigned to NAND-specific registers.

Address: 12_4000h base + 1000h offset = 12_5000h



IFC_NCFGR field descriptions

Field	Description
0 BOOT	NAND flash auto-boot load mode: During system boot from NAND flash, this bit remains set to alter the use of the FCM buffer RAM. Software should clear BOOT once FCM is to be restored for normal operation. Setting BOOT without auto-boot in progress only alters the mapping of the buffer RAM. 0 NAND FCM is operating in normal functional mode, with a 16-Kbyte FCM buffer RAM. In this case normal SRAM buffer mapping applies. 1 Flash is accessed in boot mode. SRAM buffer mapping gets changed and whole 8 Kbyte NAND flash main area in the buffer looks as contiguous linear addressable memory region. This bit is set by hardware when por_cfg_rcw_load or por_cfg_boot_load comes with NAND as bootable device.
1 -	This field is reserved.
2 SRAM_INIT_EN	SRAM Initialization Enable This bit is provided to trigger the auto initialization of complete SRAM with FF data. Software must set this bit before initiating first program operation on NAND post reset. IFC clears this bit after finishing this operation.

Table continues on the next page...

IFC_NCFGR field descriptions (continued)

Field	Description
	After setting this bit to 1'b1, software must poll this bit. Reading back value 1'b0 confirms that IFC hardware has filled the SRAM and cleared this bit. 0 No auto SRAM Initialization. 1 Triggers the IFC sram initialization logic which fills all sram buffer locations with 'hFF data
3–7 -	This field is reserved.
8–9 ADDR_MODE	Addressing Mode: Applicable for the current active CS. Applicable only with opcodes CA0 and RA0. Others Reserved 00 ROW0/COL0, ROW0+1/COL0, ROW0+2/COL0 and so on, in each iteration defined by NUM_LOOP field (Incremental by Page Size) 01 ROW0/COL0 addresses in first iteration, ROW1/COL1 address in second iteration, ROW2/COL2 address in third iteration, ROW3/COL3 address in fourth iteration
10–15 -	This field is reserved.
16–19 NUM_LOOP	Number of loop iterations of FIR sequences for multipage operations FIR sequence will execute lopping by NUM_LOOP's time. **NOTE:** 1 NUM_LOOP value should be programmed carefully as it is restricted by SRAM buffer size. We support 16-page operation for small page, 4-page operation for large page of size 2 KB and 2-page operations for 4 KB page size. Hence NUM_LOOP value of 0000-1111 is only valid for small page; for 2 KB page size NUM_LOOP 0000-0011 are valid values; and for 4-KB page size NUM_LOOP value 0000 and 0001 are valid. **NOTE:** 2 For ADDR_MODE = 01, NUM_LOOP should not exceed 4. **NOTE:** 3 Ensure that separate SRAM buffers should be allocated during consecutive runs of NUM_LOOP else the data from the previous iteration will be overwritten and produce incorrect results. 0000 1 time execution of FIR sequence 0001 2 time execution of FIR sequence 0010 3 time execution of FIR sequence 0011 4 time execution of FIR sequence 0100 5 time execution of FIR sequence 0101 6 time execution of FIR sequence 0110 7 time execution of FIR sequence 0111 8 time execution of FIR sequence 1000 9 time execution of FIR sequence 1001 10 time execution of FIR sequence 1010 11 time execution of FIR sequence 1011 12 time execution of FIR sequence 1100 13 time execution of FIR sequence 1101 14 time execution of FIR sequence 1110 15 time execution of FIR sequence 1111 16 time execution of FIR sequence
20–23 -	This field is reserved.

Table continues on the next page...

IFC_NCFGR field descriptions (continued)

Field	Description
24–31 NUM_WAIT	Number of wait cycles It represents the count value for which FCM will wait when used with opcode NWAIT in FIR. The value specified in this register represents the number of IFC module input clock cycles. Minimum value to be programmed is 2.

13.3.31 NAND Flash Command Register 0 (IFC_NAND_FCR0)

The NAND flash command registers hold up to 8 NAND flash EEPROM command bytes that may be referenced by opcodes in NAND_FIR during FCM operation. The values of the commands should follow the manufacturer's datasheet for the relevant NAND flash.

Address: 12_4000h base + 1014h offset = 12_5014h

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	CMD0								CMD1								CMD2								CMD3							
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

IFC_NAND_FCR0 field descriptions

Field	Description
0–7 CMD0	General purpose FCM flash command byte 0. Opcodes in FIR that issue command index 0 write CMD0 to the NAND flash command/data bus.
8–15 CMD1	General purpose FCM flash command byte 1. Opcodes in FIR that issue command index 1 write CMD1 to the NAND flash command/data bus.
16–23 CMD2	General purpose FCM flash command byte 2. Opcodes in FIR that issue command index 2 write CMD2 to the NAND flash command/data bus.
24–31 CMD3	General purpose FCM flash command byte 3. Opcodes in FIR that issue command index 3 write CMD3 to the NAND flash command/data bus

13.3.32 NAND flash command register 1 (IFC_NAND_FCR1)

Address: 12_4000h base + 1018h offset = 12_5018h

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	CMD4								CMD5								CMD6								CMD7							
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

IFC_NAND_FCR1 field descriptions

Field	Description
0–7 CMD4	General purpose FCM flash command byte 4. Opcodes in FIR that issue command index 4 write CMD4 to the NAND flash command/data bus.
8–15 CMD5	General purpose FCM flash command byte 5. Opcodes in FIR that issue command index 5write CMD5 to the NAND flash command/data bus.
16–23 CMD6	General purpose FCM flash command byte 6. Opcodes in FIR that issue command index 6 write CMD6 to the NAND flash command/data bus.
24–31 CMD7	General purpose FCM flash command byte 7. Opcodes in FIR that issue command index 7 write CMD7 to the NAND flash command/data bus

13.3.33 Flash Row Address Register n (IFC_ROW*n*)

ROW*n* registers are used for addressing the NAND flash memory. These registers are used as per the field NCFGR1[ADDR_MODE]. ROW*n* register holds the row address to be issued on NAND flash interface using address phase. CSOR*n*[RAL] field determines the number of bits to be issued to the flash from ROW register during row address phase.

Address: 12_4000h base + 103Ch offset + (16d × i), where i=0d to 3d

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	RA																															
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

IFC_ROW*n* field descriptions

Field	Description
0–31 RA	Row Address This register holds the row address to be issued on flash during row address phase

13.3.34 Flash COL Address Register n (IFC_COL*n*)

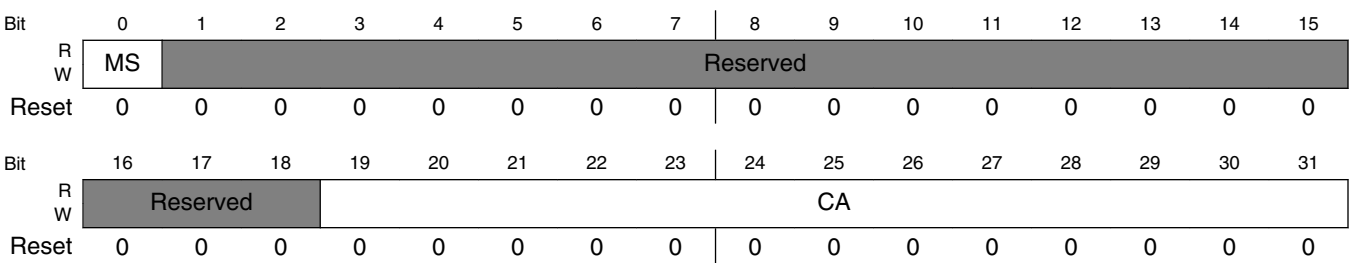
COL*n* registers are used for addressing the NAND FLASH memory. These registers are used as per the field NCFGR1[ADDR_MODE]. The following holds the column address to be issued to flash during address phase. CSOR*n*[PGS] field determines the number of bits to be issued to flash from COL register during column address phase.

For small-page size, that is, 512 Bytes, the four lsbs of ROW*n* register is used to select the IFC buffer for data transfer to/from IFC buffer to/from flash. It indexes the page in NAND flash EEPROM at the current block, and locates the corresponding transfer buffer in the FCM buffer RAM.

The four lsbs of RA index, 1 of the 16, 1-Kbyte buffers in the FCM buffer RAM as follows:

- 0000-The page is transferred to/from FCM buffer 0, address offsets 0x0000-0x03FF
- 0001-The page is transferred to/from FCM buffer 1, address offsets 0x0400-0x07FF
- 0010-The page is transferred to/from FCM buffer 2, address offsets 0x0800-0x0BFF
- 0011-The page is transferred to/from FCM buffer 3, address offsets 0x0C00-0x0FFF
- 0100-The page is transferred to/from FCM buffer 4, address offsets 0x1000-0x13FF
- 0101-The page is transferred to/from FCM buffer 5, address offsets 0x1400-0x17FF
- 0110-The page is transferred to/from FCM buffer 6, address offsets 0x1800-0x1BFF
- 0111-The page is transferred to/from FCM buffer 7, address offsets 0x1C00-0x1FFF
- 1000-The page is transferred to/from FCM buffer 8, address offsets 0x2000-0x23FF
- 1001-The page is transferred to/from FCM buffer 9, address offsets 0x2400-0x27FF
- 1010-The page is transferred to/from FCM buffer 10, address offsets 0x2800-0x2BFF
- 1011-The page is transferred to/from FCM buffer 11, address offsets 0x2C00-0x2FFF
- 1100-The page is transferred to/from FCM buffer 12, address offsets 0x3000-0x33FF
- 1101-The page is transferred to/from FCM buffer 13, address offsets 0x3400-0x37FF
- 1110-The page is transferred to/from FCM buffer 14, address offsets 0x3800-0x3BFF
- 1111-The page is transferred to/from FCM buffer 15, address offsets 0x3C00-0x3FFF

Address: 12_4000h base + 1044h offset + (16d × i), where i=0d to 3d



IFC_COL*n* field descriptions

Field	Description
0 MS	Main/spare region locator.: **NOTE:** In the case that NAND_BC[BC] = 0, MS is treated as 0. 0 Data is transferred to/from the main region of the SRAM buffer; that is, the first 512 bytes of the buffer are used. 1 Data is transferred to/from the spare region of the SRAM buffer; that is, the second 512 bytes of the buffer are used, but only an initial 16 bytes of spare region are defined.
1–18 -	This field is reserved.

Table continues on the next page...

IFC_COL*n* field descriptions (continued)

Field	Description
19–31 CA	Column Address This register holds the column address to be issued on flash during column address phase. CA indexes the first byte to transfer to/from the main or spare region of the NAND flash EEPROM and corresponding transfer buffer. In the case that NAND_BC[BC] = 0, CA is treated as 0. For MS = 0, CA can range 0x0000-0x1FF; for MS = 1, CA can range 0x000-0x00F. For a 16-bit port size, the least significant bit of the Column Address is assumed to be zero; hence, the Column Address remains a byte index at all times.

13.3.35 Flash COL Address Register for 2 KB Large-Page Device (IFC_COL*n*_2KB)

For large-page size of 2 Kbytes, two lsbs of ROW*n* register are used to select the IFC buffer for data transfer to/from IFC buffer to/from flash. It indexes the page in NAND flash EEPROM at the current block, and locates the corresponding transfer buffer in the FCM buffer RAM.

The two lsbs of RA index one of the four 4-Kbyte buffers in the FCM buffer RAM as follows:

- 00-The page is transferred to/from FCM buffer 0, address offsets 0x0000-0x0FFF
- 01-The page is transferred to/from FCM buffer 1, address offsets 0x1000-0x1FFF
- 10-The page is transferred to/from FCM buffer 2, address offsets 0x2000-0x2FFF
- 11-The page is transferred to/from FCM buffer 3, address offsets 0x3000-0x3FFF

Address: 12_4000h base + 1044h offset + (16d × i), where i=0d to 3d

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

IFC_COL*n*_2KB field descriptions

Field	Description
0 MS	Main/spare region locator. In the case that NAND_BC[BC] = 0, MS is treated as 0. 0 Data is transferred to/from the main region of the SRAM buffer; that is, the first 2048 bytes of the buffer are used. 1 Data is transferred to/from the spare region of the SRAM buffer; that is, the second 2048 bytes of the buffer are used, but only an initial 64 bytes of spare region are defined.

Table continues on the next page...

IFC_COL*n*_2KB field descriptions (continued)

Field	Description
1–18 -	This field is reserved.
19–31 CA	Column Address This register holds the column address to be issued on flash during column address phase. CA indexes the first byte to transfer to/from the main or spare region of the NAND flash EEPROM and corresponding transfer buffer. In the case that FBCR[BC] = 0, CA is treated as 0. For MS = 0, CA can range 0x000-0x7FF; for MS = 1, CA can range 0x000-0x03F. For a 16-bit port size, the least significant bit of the Column Address is assumed to be zero; hence, the Column Address remains a byte index at all times.

13.3.36 Flash COL Address Register for 4 KByte Large-Page Device (IFC_COL*n*_4KB)

For large-page size of 4 Kbytes, lsb of ROW*n* register is used to select the IFC buffer for data transfer to/from IFC buffer to/from flash. It indexes the page in NAND flash EEPROM at the current block, and locates the corresponding transfer buffer in the FCM buffer RAM.

The lsb of RA indexes one of the two 8-Kbyte buffers in the FCM buffer RAM as follows:

- 0-The page is transferred to/from FCM buffer 0, address offsets 0x0000-0x1FFF
- 1-The page is transferred to/from FCM buffer 1, address offsets 0x2000-0x3FFF

Address: 12_4000h base + 1044h offset + (16d × i), where i=0d to 3d

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

IFC_COL*n*_4KB field descriptions

Field	Description
0 MS	Main/spare region locator. In the case that NAND_BC[BC] = 0, MS is treated as 0. 0 Data is transferred to/from the main region of the SRAM buffer; that is, the first 4096 bytes of the buffer are used. 1 Data is transferred to/from the spare region of the SRAM buffer; that is, the second 4096 bytes of the buffer are used, but only an initial 64 bytes of spare region are defined.

Table continues on the next page...

IFC_COL*n*_4KB field descriptions (continued)

Field	Description
1–18 -	This field is reserved.
19–31 CA	Column Address This register holds the column address to be issued on flash during column address phase. CA indexes the first byte to transfer to/from the main or spare region of the NAND flash EEPROM and corresponding transfer buffer. In the case that FBCR[BC] = 0, COL is treated as 0. For MS = 0, CA can range 0x000-0xFFF; for MS = 1, CA can range 0x000- CSOR*n*[SPRZ]. For a 16-bit port size, the least significant bit of the Column Address is assumed to be zero; hence, the Column Address remains a byte index at all times.

13.3.37 Flash Byte Count Register for NAND Flash (IFC_NAND_BC)

This register defines the NAND flash data block transfer size.

Address: 12_4000h base + 1108h offset = 12_5108h

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																BC															
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

IFC_NAND_BC field descriptions

Field	Description
0–17 -	This field is reserved.
18–31 BC	Byte count determines how many bytes are transferred by the flash controller during data read (RBCD) or data write (WBCD) opcodes. In case of partial page operations, the value programmed in this field must be aligned to the port size of the device. Thus, if the bank port size is 16 bits, the lsb bit of BC (that is, bit 31) is ignored. The first byte accessed in the NAND flash is located by the COLn register, and successive bytes are transferred until BC bytes have been counted If BC = 0, an entire flash page and its spare region will be transferred by FCM, in which case COL*n*[MS] and COL*n*[CA] are treated as zero regardless of their values. **NOTE:** BC = 0 is the only setting that permits flash controller to generate and check ECC.

13.3.38 NAND flash instruction register 0 (IFC_NAND_FIR0)

Following flash instruction registers hold the instructions. When any operation on selected bank is triggered and FIR is programmed by the user, flash controller executes 6-bit opcode at a time till all the opcodes are being fetched from all the FIR registers. There are three such registers provided for the application. A total of 15 instructions can be programmed for FLASH operations.

NAND flash instruction register0 holds first five opcodes.

Table 13-210. Instruction Opcodes

Opcode encoding (6 bits)	Opcode name	Description
0x00	NOOP	No-operation and end of operation sequence
0x01	CA0	Issue current column address (CA) as set in COL0
0x02	CA1	Issue current column address as set in COL1
0x03	CA2	Issue current column address as set in COL2
0x04	CA3	Issue current column address as set in COL3
0x05	RA0	Issue current row address as set in ROW0
0x06	RA1	Issue current row address as set in ROW1
0x07	RA2	Issue current row address as set in ROW2
0x08	RA3	Issue current row address as set in ROW3
0x09	CMD0	Issue command from NAND_FCR0[CMD0]
0x0A	CMD1	Issue command from NAND_FCR0[CMD1]
0x0B	CMD2	Issue command from NAND_FCR0[CMD2]
0x0C	CMD3	Issue command from NAND_FCR0[CMD3]
0x0D	CMD4	Issue command from NAND_FCR1[CMD4]
0x0E	CMD5	Issue command from NAND_FCR1[CMD5]
0x0F	CMD6	Issue command from NAND_FCR1[CMD6]
0x10	CMD7	Issue command from NAND_FCR1[CMD7]
0x11	CW0	Wait for TWBE time, poll R/B to return high or time-out (depends upon IFC_NANDCR[FTOCNT]), then issue command from NAND_FCR0[CMD0]
0x12	CW1	Wait for TWBE time, poll R/B to return high or time-out (depends upon IFC_NANDCR[FTOCNT]), then issue command from NAND_FCR0[CMD1]
0x13	CW2	Wait for TWBE time, poll R/B to return high or time-out (depends upon IFC_NANDCR[FTOCNT]), then issue command from NAND_FCR0[CMD2]
0x14	CW3	Wait for TWBE time, poll R/B to return high or time-out (depends upon IFC_NANDCR[FTOCNT]), then issue command from NAND_FCR0[CMD3]

Table continues on the next page...

Table 13-210. Instruction Opcodes (continued)

Opcode encoding (6 bits)	Opcode name	Description
0x15	CW4	Wait for TWBE time, poll R/B to return high or time-out (depends upon IFC_NANDCR[FTOCNT]), then issue command from NAND_FCR1[CMD4]
0x16	CW5	Wait for TWBE time, poll R/B to return high or time-out (depends upon IFC_NANDCR[FTOCNT]), then issue command from NAND_FCR1[CMD5]
0x17	CW6	Wait for TWBE time, poll R/B to return high or time-out (depends upon IFC_NANDCR[FTOCNT]), then issue command from NAND_FCR1[CMD6]
0x18	CW7	Wait for TWBE time, poll R/B to return high or time-out (depends upon IFC_NANDCR[FTOCNT]), then issue command from NAND_FCR1[CMD7]
0x19	WBCD	Write BC bytes of data from current FCM buffer to flash device
0x1A	RBCD	Wait for R/B_B to return high or time-out (depends upon IFC_NANDCR[FTOCNT]), then Read BC bytes of data from flash device into current FCM RAM buffer
0x1B	BTRD	Same as RBCD except in this case deassertion of read enable will happen once the read data has been sampled (same as boot read)
0x1C	RDSTAT	Wait for TWHRE time, then Read one byte/two bytes (8b/16b port) of data from flash device into RS0 and RS1 field of FSR
0x1D	NWAIT	Wait for NCFGR[NUM_WAIT] Clock cycles
0x1E	WFR	Wait for TWBE time, poll Ready
0x1F	SBRD	Wait for R/B to return high or time-out (depends upon IFC_NANDCR[FTOCNT]), then Read 8/16 bits of data from NAND flash memory into the NAND_MDR register. COLn register will determine the location from where the data will be fetched in a given page. Deassertion of read enable will happen once the read data has been sampled. With this opcode NAND_BC register field value will be ignored. Although the value is ignored, it is mandatory that the NAND_BC register hold a non-zero value else the Column Address would be treated as 0 (refer Flash COL Address Register n (IFC_COL*n*))
0x20	UA	Issue current row address as set in ROW3 The SRAM Buffer used for data transfers is always Buffer 0
0x21	RB	Wait for TWHRE time, then Read BC bytes of data from flash device into current FCM RAM buffer. Deassertion of read enable will happen once the read data has been sampled
Others	-	Reserved

Address: 12_4000h base + 1110h offset = 12_5110h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	OP0						OP1						OP2			
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	OP2				OP3				OP4						Reserved	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IFC_NAND_FIR0 field descriptions

Field	Description
0–5 OP0	Opcode0
6–11 OP1	Opcode1
12–17 OP2	Opcode2
18–23 OP3	Opcode3
24–29 OP4	Opcode4
30–31 -	This field is reserved.

13.3.39 NAND flash instruction register 1 (IFC_NAND_FIR1)

NAND flash instruction register1 holds other five opcodes.

Address: 12_4000h base + 1114h offset = 12_5114h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	OP5						OP6						OP7			
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	OP7				OP8				OP9						Reserved	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IFC_NAND_FIR1 field descriptions

Field	Description
0–5 OP5	Opcode5
6–11 OP6	Opcode6
12–17 OP7	Opcode7
18–23 OP8	Opcode8

Table continues on the next page...

IFC_NAND_FIR1 field descriptions (continued)

Field	Description
24–29 OP9	Opcode9
30–31 -	This field is reserved.

13.3.40 NAND flash instruction register 2 (IFC_NAND_FIR2)

NAND flash instruction register2 holds last five opcodes.

Address: 12_4000h base + 1118h offset = 12_5118h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	OP10								OP11				OP12			
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	OP12		OP13						OP14						Reserved	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

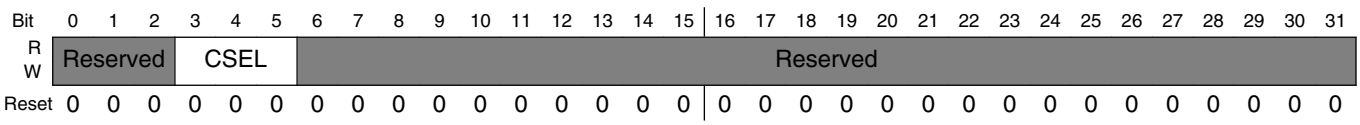
IFC_NAND_FIR2 field descriptions

Field	Description
0–5 OP10	Opcode10
6–11 OP11	Opcode11
12–17 OP12	Opcode12
18–23 OP13	Opcode13
24–29 OP14	Opcode14
30–31 -	This field is reserved.

13.3.41 NAND chip-select register (IFC_NAND_CSEL)

This register tells the NAND FCM about the selected chip-select on which program or read operation has to be performed. As NAND flash is accessed through an SRAM buffer (memory-mapped), software tells the NAND FCM which chip-select is desired using this register.

Address: 12_4000h base + 115Ch offset = 12_515Ch



IFC_NAND_CSEL field descriptions

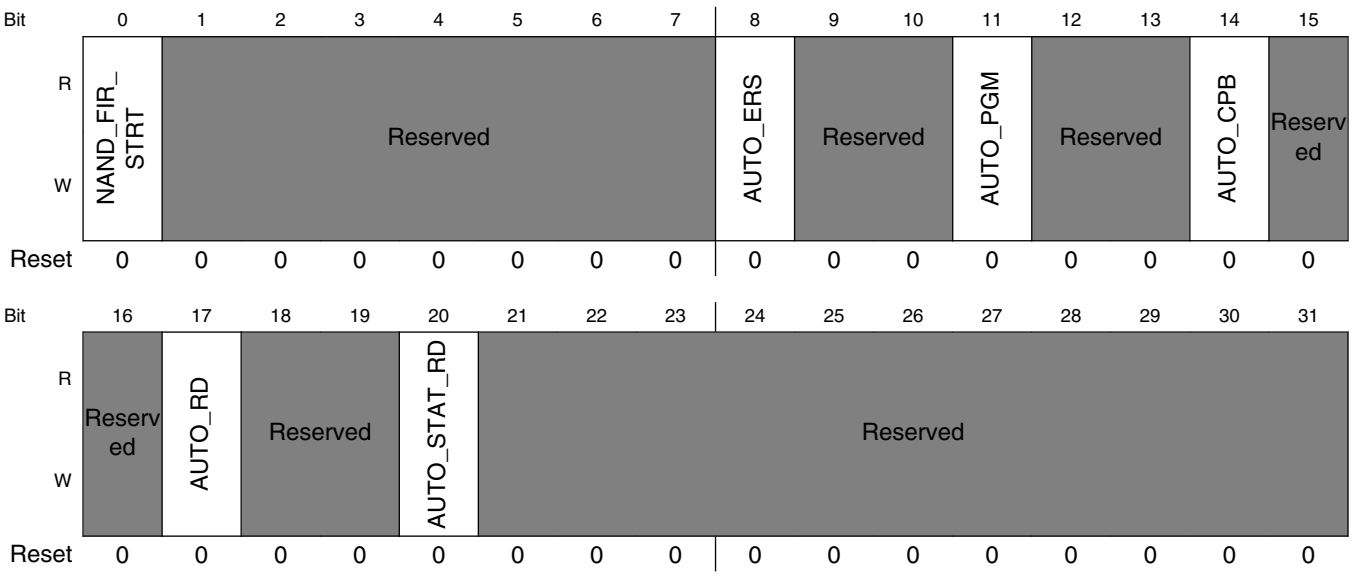
Field	Description
0–2 -	This field is reserved.
3–5 CSEL	Chip-Select for NAND flash operation 000 Chip-select0 001 Chip-select1 010 Chip-select2 011 Chip-select3 . . . 111 Chip select7
6–31 -	This field is reserved.

13.3.42 NAND operation sequence start (IFC_NANDSEQ_STRT)

This register is used to trigger the operation on NAND flash.

The following shows the NANDSEQ_STRT register. Only one operation can be triggered at a time. Software has to trigger the operation and the chip will clear this after completing the operation.

Address: 12_4000h base + 1164h offset = 12_5164h



IFC_NANDSEQ_STRT field descriptions

Field	Description
0 NAND_FIR_STRT	NAND flash operation start Writing 1 to this register bit triggers normal operation sequence programmed in NAND FIR registers on NAND flash.
1–7 -	This field is reserved.
8 AUTO_ERS	Automatic erase Writing 1 to this register bit triggers automatic erase operation **NOTE:** The FIR sequence executed is {CW0, RA0, CMD1, and NOOP}
9–10 -	This field is reserved.
11 AUTO_PGM	Automatic program Writing 1 to this register bit triggers automatic program operation **NOTE:** The FIR sequence executed is {CW0, CA0, RA0, WBCD, CMD1, and NOOP}
12–13 -	This field is reserved.
14 AUTO_CPB	Automatic copyback Writing 1 to this register bit triggers automatic copy back operation **NOTE:** The FIR sequence executed is {CW0, CA0, RA0, CMD1, CW2, CA1, RA1, CMD3, CW4, RDSTAT, and NOOP}
15–16 -	This field is reserved.
17 AUTO_RD	Automatic read operation Writing 1 to this register bit triggers automatic read operation **NOTE:** The FIR sequence executed is:

Table continues on the next page...

IFC_NANDSEQ_STRT field descriptions (continued)

Field	Description
	For small-page device: {CW0, CA0, RA0, RBCD, NOOP} For large-page device: {CW0, CA0, RA0, CMD1, RBCD, NOOP}
18–19 -	This field is reserved.
20 AUTO_STAT_RD	Automatic status read Writing 1 to this register bit triggers automatic read status **NOTE:** The FIR sequence executed is {CW0, RDSTAT, NOOP}
21–31 -	This field is reserved.

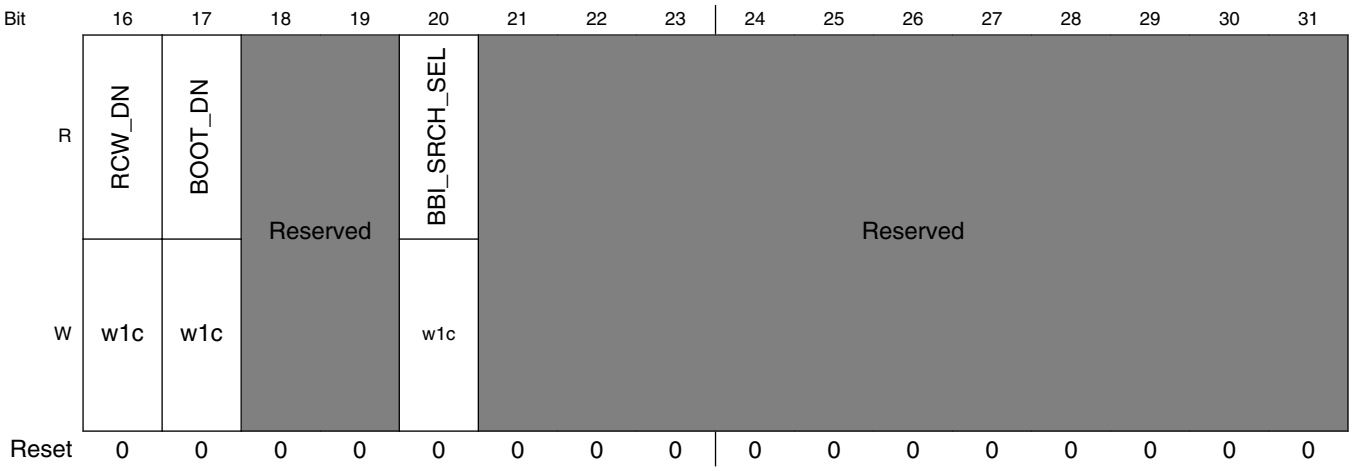
13.3.43 NAND event and error status register (IFC_NAND_EVTER_STAT)

NAND event and error status register (NAND_EVTER_STAT) indicates the cause of an error or event corresponding to NAND flash.

The following shows the register fields. It is write-1-to-clear register.

Address: 12_4000h base + 116Ch offset = 12_516Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	OPC	Reserved			FTOER	WPER	ECCER	Reserved								
W	w1c				w1c	w1c	w1c									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



IFC_NAND_EVTER_STAT field descriptions

Field	Description
0 OPC	NAND flash operation complete event. OPC indicates that all instructions in FIR has been executed. **NOTE:** 1. The software should not poll this bit during auto boot. Instead RCW_DN/ BOOT_DN should be used. However, software must clear this bit post auto boot. **NOTE:** 2. This bit is set even if a flash timeout, ECC or write protect error occurs. For example, if a flash timeout error is detected for a read operation intiated on the NAND flash memory device, IFC will dump BC bytes worth of garbage data from the NAND flash interface into the internal SRAM and set this bit. Similarly, in case of NAND write protect error IFC asserts chip select and sends the program data to NAND device, it is device who ignores the data as write protect signal is asserted. As the transaction goes into NAND, OPC will be set at the end of FIR execution. OPC bit indicates that all the instructions in the FIRs have been executed. Also, it implies that OPC should be polled first and then FSR for completion. 0 NAND flash operation is not complete 1 NAND flash operation completed
1–3 -	This field is reserved.
4 FTOER	Flash timeout error 0 No flash interface timeout 1 Flash interface timeout occured
5 WPER	Write protect error **NOTE:** Error is asserted when the write operation starts executing on the flash interface. 0 No write protect error 1 A write is attempted to the memory bank which is write protected
6 ECCER	ECC error **NOTE:** With the clearing of this error status bit, the ECCSTAT0/1/2/3 register contents will also be cleared. 0 No uncorrectable ECC Error occured on page read operation 1 Uncorrectable ECC error occured in page read operation

Table continues on the next page...

IFC_NAND_EVTER_STAT field descriptions (continued)

Field	Description
7–15 -	This field is reserved.
16 RCW_DN	This bit is set when one flash page has been read from the flash device and written into the SRAM buffer during RCW load. It is write-1-to-clear bit and can be cleared by software after RCW loading. This event does not have any corresponding event enable and interrupt enable bit.
17 BOOT_DN	This event is set when 8 KB flash data has been read from the flash device and written into the SRAM buffer during BOOT load. It is write-1-to-clear bit and can be cleared by software after BOOT loading. This events will not have any corresponding event enable and interrupt enable bit.
18–19 -	This field is reserved.
20 BBI_SRCH_SEL	Bad Block Indicator search select: This status register bit represents the location of bad block indication in each block of NAND flash device connected at chip-select0. This field will be updated with input port value por_cfg_bbi_srch_sel when por_cfg_boot_load /por_cfg_rcw_load pulse comes. 0 Read bad block indicator (BBI) corresponding to page0 and page1 of each block of the NAND flash device connected at chipselect 0, to identify the first good block during auto-boot 1 Read bad block indicator (BBI) corresponding to page0 and the last page of each block of the NAND flash device connected at chipselect 0, to identify the first good block during auto-boot
21–31 -	This field is reserved.

13.3.44 NAND page read completion event status register (IFC_PGRDCMPL_EVT_STAT)

NAND flash page read completion event register indicates about the status of the pages read from NAND flash and stored in SRAM buffer. 16 bits of this register represent 16 sectors each of 512 bytes (used in ECC decoding). There can be 1, 4, and 8 sectors in each page of size 512 Bytes, 2 Kbyte, 4 Kbyte , and 8Kbyte . Event interrupt will be generated if one complete page has been written in SRAM buffer (when ecc_dec_en = 0) or when one complete page has been fixed in the SRAM after decoding (ecc_dec_en = 1). Software can read the corresponding page whose data has been completely written in SRAM (ECC fixed) and clear the corresponding bits in this register.

24/40-bit ECC is computed on 1 Kbyte sector and applicable for 4 Kbyte and 8 Kbyte page sizes. However the definition of this register remains same except the fact that 512 is the byte size and not the sector size.

NOTE

The register gets cleared when software clears the OPC bit in NAND_EVTER_STAT register.

The following shows the register fields. It is write-1-to-clear register.

Address: 12_4000h base + 1174h offset = 12_5174h

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	SEC_DONE																Reserved															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IFC_PGRDCMPL_EVT_STAT field descriptions

Field	Description
0–15 SEC_DONE	For small page: Each of the 16 bits of this field represent one small page read data completion status Bit 0 set-Page 0 done, that is, flash read data is completely written and fixed in SRAM buffer Bit 1 set-Page 1 done Bit 2 set-Page 2 done Bit 3 set-Page 3 done ... Bit15 Set-Page 15 done For 2KB Page: SRAM buffer can accommodate 4 page (2KB) each containing four 512 byte sectors SEC_DONE[0:3] = 4'b1111: Page 0 done SEC_DONE[4:7] = 4'b1111: Page 1 done SEC_DONE[8:11] = 4'b1111: Page 2 done SEC_DONE[12:15] = 4'b1111: Page 3 done For 4KB Page: SRAM buffer can accommodate 2 page (4KB) each containing eight 512 byte sectors SEC_DONE[0:7] = 8'b1111_1111: Page 0 done SEC_DONE[8:15] = 8'b1111_1111: Page 1 done **NOTE:** Event interrupt will be generated when one complete page read is done. Software has to read the corresponding page from SRAM buffer and clear the bits from this register. For example if page size is 2 KB and page 0 is being written in SRAM buffer, then SEC_DONE[0:15] will be 16'HF000, now software has to write the data 16'HF000 in this register to clear it.
16–31 -	This field is reserved.

13.3.45 NAND event and error enable register (IFC_NAND_EVTER_EN)

Event and Error Enable Register (NAND_EVTER_EN) is used to enable/disable the logging of event and error indication in the NAND_EVTER_STAT and PGRDCMPL_EVT_STAT register.

Address: 12_4000h base + 1180h offset = 12_5180h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	OPCEN	Reserved	PGRDCMPLEN	Reserved	FTOEREN	WPEREN	ECCEREN	Reserved								
Reset	1	0	1	0	1	1	1	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

IFC_NAND_EVTER_EN field descriptions

Field	Description
0 OPCEN	NAND flash operation complete event enable 0 NAND flash operation complete event is disabled 1 NAND flash operation completed event is enabled
1 -	This field is reserved.
2 PGRDCMPLEN	NAND flash page read completion event enable 0 Per page read event disable 1 Event will be generated on per page flash read operation completion. Status about the page whose data is available in sram buffer is given by PGRDCMPL_EVT_STAT register.

Table continues on the next page...

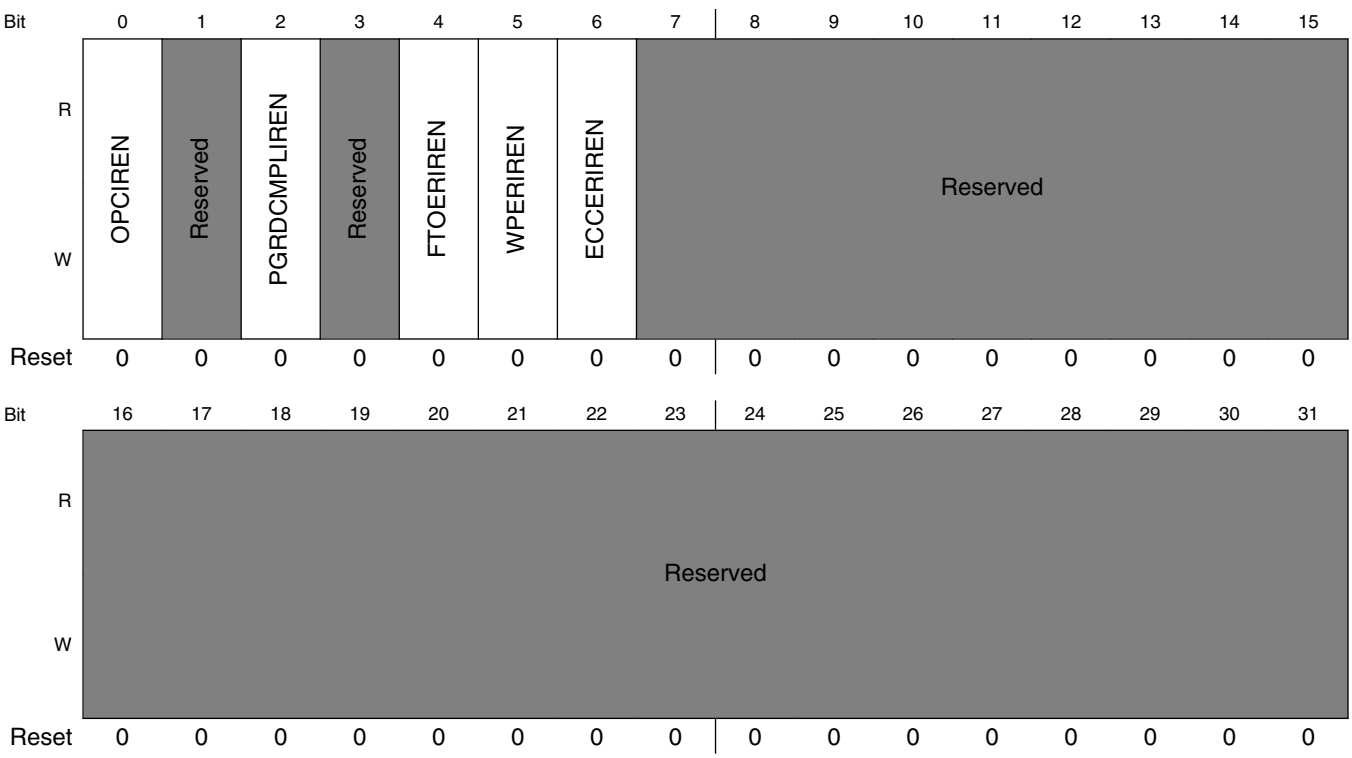
IFC_NAND_EVTER_EN field descriptions (continued)

Field	Description
3 -	This field is reserved.
4 FTOEREN	Flash time out error enable 0 Flash timeout error is disabled 1 Flash timeout error is enabled
5 WPEREN	Write protect error checking enable. 0 No write protect error checking 1 Write protect error checking is enabled
6 ECCEREN	ECC error logging enable. 0 ECCER event is not logged in NAND_EVTER_STAT register 1 ECCER event is logged in NAND_EVTER_STAT register
7–31 -	This field is reserved.

13.3.46 NAND event and error interrupt enable register (IFC_NAND_EVTER_INTR_EN)

Event and Error Interrupt Enable Register (NAND_EVTER_INTR_EN) is used to enable/disable the generation of interrupt signal corresponding to error and event reporting. Software must clear pending events and errors in NAND_EVTER_STAT and PGRDCMPL_EVT_STAT register before enabling the interrupts.

Address: 12_4000h base + 118Ch offset = 12_518Ch



IFC_NAND_EVTER_INTR_EN field descriptions

Field	Description
0 OPCIREN	NAND flash operation complete event interrupt enable 0 NAND flash operation Complete Event Interrupt is disabled 1 NAND flash operation completed Event Interrupt is Enabled
1 -	This field is reserved.
2 PGRDCMPLIREN	Page read completion event interrupt enable 0 NAND flash page read event interrupt is disabled (per page basis) 1 NAND flash page read event interrupt is enabled

Table continues on the next page...

IFC_NAND_EVTER_INTR_EN field descriptions (continued)

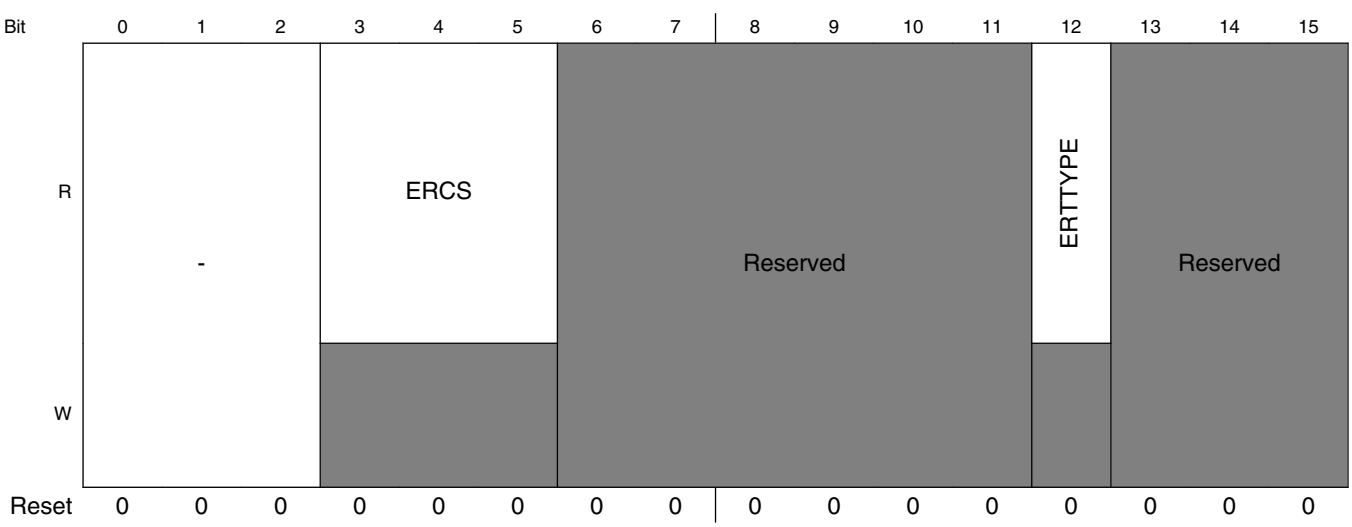
Field	Description
3 -	This field is reserved.
4 FTOERIREN	Flash timeout error interrupt enable 0 Flash timeout error interrupt is disabled 1 Flash timeout error interrupt is enabled
5 WPERIREN	Write protect error interrupt enable 0 Write protect error interrupt disable 1 Write protect error interrupt is enabled
6 ECCERIREN	ECC error interrupt enable 0 ECC Error Interrupt is disabled 1 ECC Error Interrupt is enabled
7–31 -	This field is reserved.

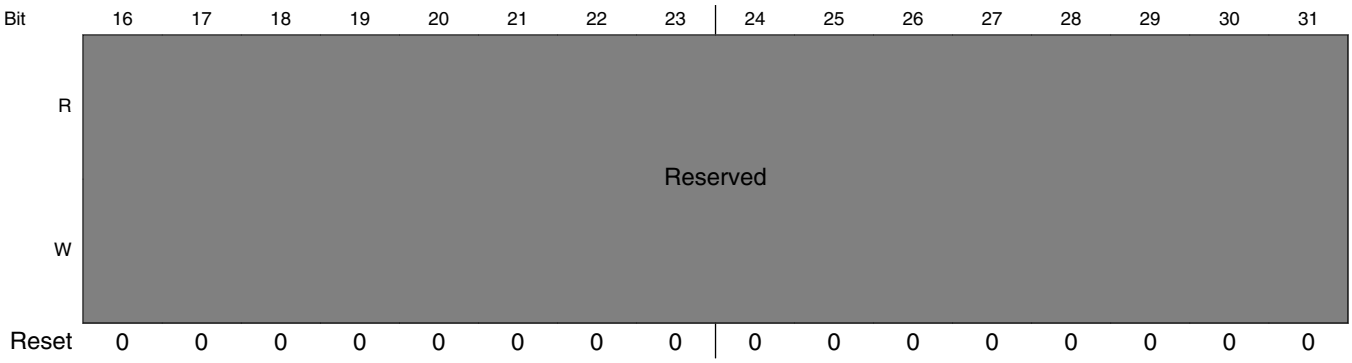
13.3.47 NAND transfer error attributes register 0 (IFC_NAND_ERATTR0)

Transfer error attribute register 0 (NAND_ERATTR0) is used to register the transaction attributes corresponding to first error occured.

These registers store the attribute corresponding to the first transaction on which error occured.

Address: 12_4000h base + 1198h offset = 12_5198h





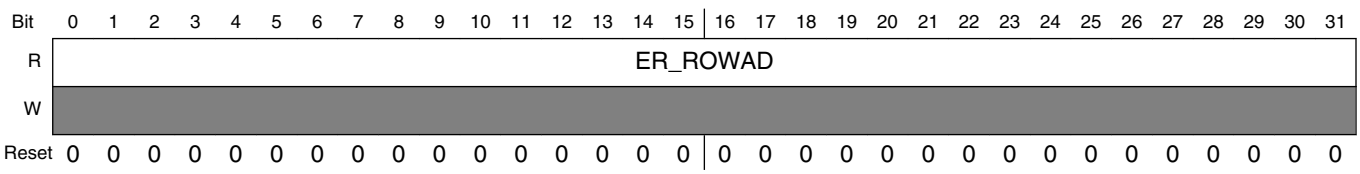
IFC_NAND_ERATTR0 field descriptions

Field	Description
0–2 -	
3–5 ERCS	Chip-Select Corresponding to NAND Error 000 Bank0 001 Bank1 010 Bank2 . . . 111 Bank7
6–11 -	This field is reserved.
12 ERTTYPE	Transaction type of NAND error 0 Write 1 Read
13–31 -	This field is reserved.

13.3.48 NAND transfer error attributes register 1 (IFC_NAND_ERATTR1)

Transfer error attribute register (NAND_ERATTR1) is used to register the transaction address corresponding to first error occured.

Address: 12_4000h base + 119Ch offset = 12_519Ch



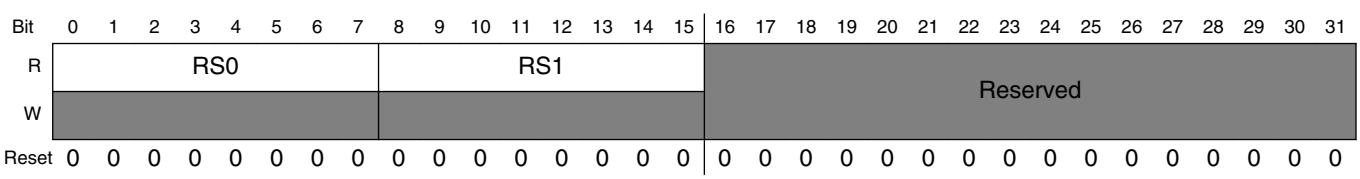
IFC_NAND_ERATTR1 field descriptions

Field	Description
0–31 ER_ROWAD	Row address corresponding to error transaction

13.3.49 NAND flash status register (IFC_NAND_FSR)

Flash status register (NAND_FSR) contains read status data from NAND flash.

Address: 12_4000h base + 11E0h offset = 12_51E0h



IFC_NAND_FSR field descriptions

Field	Description
0–7 RS0	First byte of data read from read status operation
8–15 RS1	Second byte of data read from read status operation
16–31 -	This field is reserved.

13.3.50 ECC status and result of flash operation register 0 (IFC_ECCSTAT0)

ECC status registers are used to store the number of ECC errors occured on read operation. It has different meanings for 4/8 and 24/40 bit ECC Modes as the sector size is different for 4/8 and 24/40 bit modes.

ECC is computed on sector basis, and in the SRAM buffer, there can be 16 such sectors of 512 byte each, for 4 and 8 bit ECC (Galois field GF-2 ^{13}) and 8 sectors of 1 Kbyte each for 24 and 40 bit ECC Galois Field 2 ^{14}) . Hence, each field of these registers represents number of errors in each sector. Depending on the page index mapped in the SRAM buffer, corresponding field in the ECCSTAT0 will be updated.

As SRAM buffer can have 8 sectors of 1KB each, hence only first eight register fields in ECCSTAT0/1 registers are defined with width 6 to represent 24 and 40 bit ECC along with 4 and 8 bit ECC.

NOTE

1. These register will be updated whenever decoder is enable (ECC_DEC_EN =1'b1) irrespective of ECC Error Event Indication is enabled or not (ECCEREN).
2. With the clearing of NAND_EVTER_STAT[ECCER] register bit the contents of ECCSTAT0/1/2/3 registers will also be cleared. These registers will also be cleared with soft_reset.
3. For every NAND sequence start, all the ECCSTAT registers will be cleared. It is implemented in order to flush the previously logged information.

Address: 12_4000h base + 11E8h offset = 12_51E8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved		NUMER0						Reserved		NUMER1					
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		NUMER2						Reserved		NUMER3					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IFC_ECCSTAT0 field descriptions

Field	Description
0–1 -	This field is reserved.
2–7 NUMER0	Number of ECC errors on sector #0 of SRAM buffer For 4/8 bit ECC Mode (512 byte sector): 000000 No Error 000001 1 Bit Error 000010 2 Bit Error 000011 3 bit Error 000100 4 Bit Error 000101 5 Bit Error 000110 6 Bit Error 000111 7 BIt Error 001000 8 Bit Error 001111 - Uncorrectable Errors Others Reserve For 24/40 bit ECC Mode (1KB sector): 000000 No Error 000001 - 101000 1 Bit Error to 40 bit Error in this sector 111111 Uncorrectable Errors Others Reserve
8–9 -	This field is reserved.
10–15 NUMER1	Number of ECC errors on sector #1 of SRAM buffer
16–17 -	This field is reserved.
18–23 NUMER2	Number of ECC errors on sector #2 of SRAM buffer
24–25 -	This field is reserved.
26–31 NUMER3	Number of ECC errors on sector #3 of SRAM buffer

13.3.51 ECC status and result of flash operation register 1 (IFC_ECCSTAT1)

ECC Status Register (ECCSTAT1) is used to store the number of ECC errors occured on Sector 4-7 during page read operation.

Address: 12_4000h base + 11ECh offset = 12_51ECh

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved		NUMER4						Reserved		NUMER5					
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		NUMER6						Reserved		NUMER7					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IFC_ECCSTAT1 field descriptions

Field	Description
0–1 -	This field is reserved.
2–7 NUMER4	Number of ECC errors on sector #4 of SRAM buffer
8–9 -	This field is reserved.
10–15 NUMER5	Number of ECC errors on sector #5 of SRAM buffer
16–17 -	This field is reserved.
18–23 NUMER6	Number of ECC errors on sector #6 of SRAM buffer
24–25 -	This field is reserved.
26–31 NUMER7	Number of ECC errors on sector #7 of SRAM buffer

13.3.52 ECC status and result of flash operation register 2 (IFC_ECCSTAT2)

ECC Status Register (ECCSTAT1) is used to store the number of ECC errors occured on Sector 8-11 during page read operation.

Address: 12_4000h base + 11F0h offset = 12_51F0h

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				NUMER8				Reserved				NUMER9				Reserved				NUMER10				Reserved				NUMER11			
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

IFC_ECCSTAT2 field descriptions

Field	Description
0–3 -	This field is reserved.
4–7 NUMER8	Number of ECC errors on sector #8 of SRAM buffer
8–11 -	This field is reserved.
12–15 NUMER9	Number of ECC errors on sector #9 of SRAM buffer
16–19 -	This field is reserved.
20–23 NUMER10	Number of ECC errors on sector #10 of SRAM buffer
24–27 -	This field is reserved.
28–31 NUMER11	Number of ECC errors on sector #11 of SRAM buffer

13.3.53 ECC status and result of flash operation register 3 (IFC_ECCSTAT3)

ECC Status Register (ECCSTAT3) is used to store the number of ECC errors occured on Sector 12-15 during page read operation.

Address: 12_4000h base + 11F4h offset = 12_51F4h

Bit	0–3	4–7	8–11	12–15	16–19	20–23	24–27	28–31
R	Reserved	NUMER12	Reserved	NUMER13	Reserved	NUMER14	Reserved	NUMER15
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

IFC_ECCSTAT3 field descriptions

Field	Description
0–3 -	This field is reserved.
4–7 NUMER12	Number of ECC errors on sector #12 of SRAM buffer
8–11 -	This field is reserved.
12–15 NUMER13	Number of ECC errors on sector #13 of SRAM buffer
16–19 -	This field is reserved.
20–23 NUMER14	Number of ECC errors on sector #14 of SRAM buffer
24–27 -	This field is reserved.
28–31 NUMER15	Number of ECC errors on sector #15 of SRAM buffer

13.3.54 NAND control register (IFC_NANDCR)

Address: 12_4000h base + 1278h offset = 12_5278h

Bit	0–2	3–6	7–15	16–31
R	Reserved	FTOCNT	Reserved	
W				
Reset	0 0 0	1 1 1 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

IFC_NANDCR field descriptions

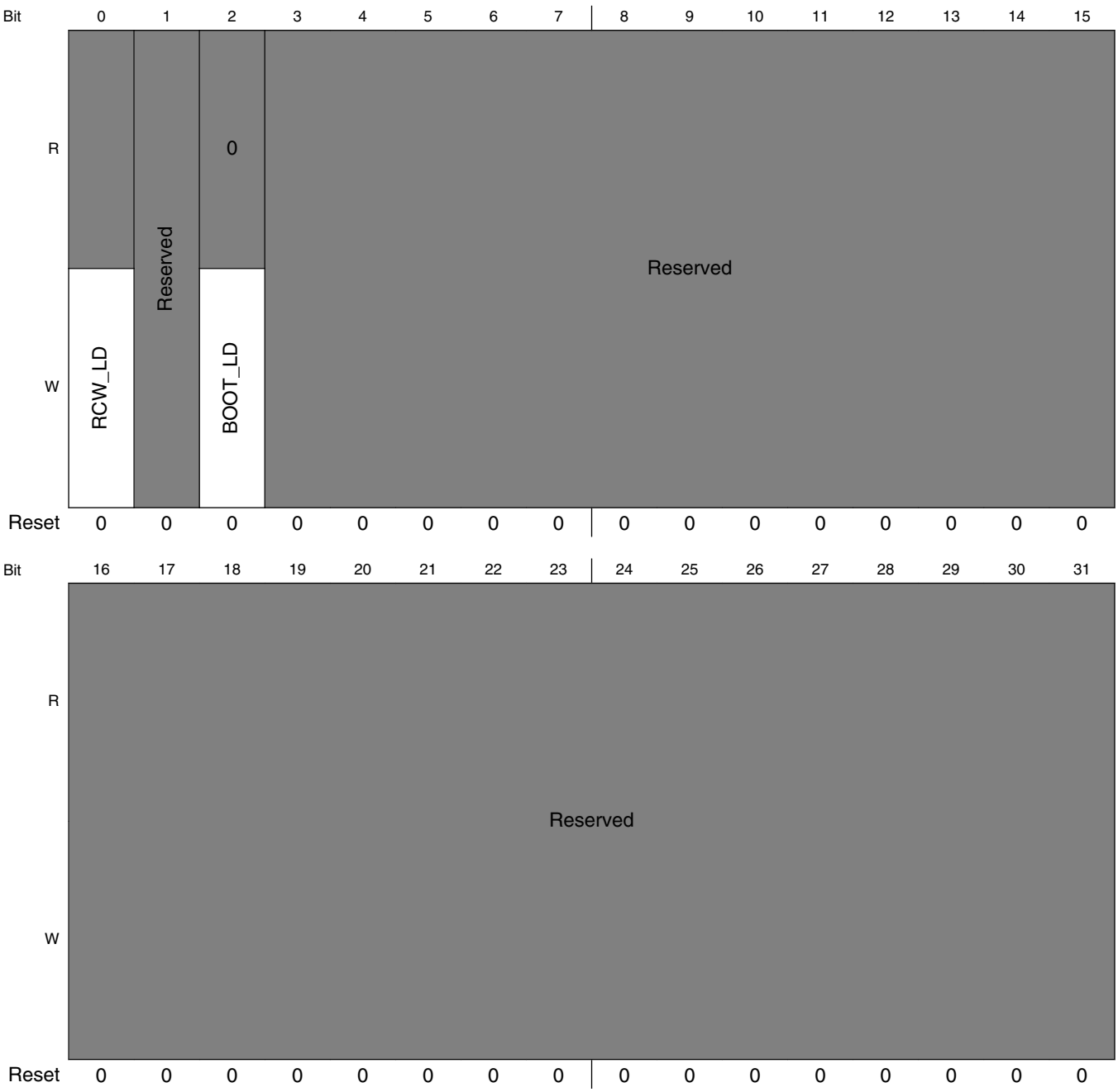
Field	Description
0–2 -	This field is reserved.
3–6 FTOCNT	Flash timeout count 0000 256 cycles of IFC module input clock 0001 512 cycles of IFC module input clock 0010 1024 cycles of IFC module input clock 0011 2048 cycles of IFC module input clock 0100 4096 cycles of IFC module input clock 0101 8192 cycles of IFC module input clock 0110 16,384 cycles of IFC module input clock 0111 32,768 cycles of IFC module input clock 1000 65,536 cycles of IFC module input clock 1001 131,072 cycles of IFC module input clock 1010 262,144 cycles of IFC module input clock 1011 524,288 cycles of IFC module input clock 1100 1,048,576 cycles of IFC module input clock 1101 2,097,152 cycles of IFC module input clock 1110 4,194,304 cycles of IFC module input clock 1111 8,388,608 cycles of IFC module input clock
7–31 -	This field is reserved.

13.3.55 NAND autoboot trigger register (IFC_NAND_AUTOBOOT_TRGR)

Following register is used to trigger the booting on the NAND flash. This register is provided if por_cfg_rcw_load and por_cfg_boot_load signals are not coming through the pins for the booting. Hence, the user can use this register to achieve the same boot functionality as described in NAND asynchronous mode boot mechanism .

It is a self-clearing register; hence, the user just has to write 1 in appropriate field to trigger the auto-boot operation and hardware will clear it.

Address: 12_4000h base + 1284h offset = 12_5284h



IFC_NAND_AUTOBOOT_TRGR field descriptions

Field	Description
0 RCW_LD	RCW Load **NOTE:** The values on the various por_cfg pins should be driven valid when this bit is set. 0 No RCW loading 1 Trigger RCW load from NAND flash (same functionality that can be achieved by sending pulse on por_cfg_rcw_load)

Table continues on the next page...

IFC_NAND_AUTOBOOT_TRGR field descriptions (continued)

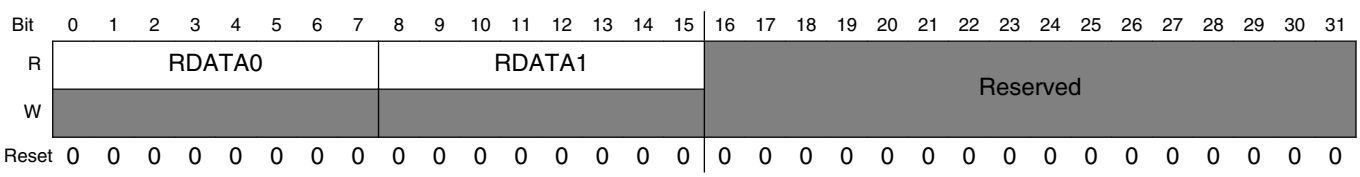
Field	Description
1 -	This field is reserved.
2 BOOT_LD	BOOT Load **NOTE:** The values on the various por_cfg pins should be driven valid when this bit is set. 0 No BOOT loading 1 Trigger autoboot loading from NAND flash (same functionality that can be achieved by sending pulse on por_cfg_boot_load)
3–31 -	This field is reserved.

13.3.56 NAND flash memory data register (IFC_NAND_MDR)

The following register is used to store one beat of data read from NAND flash memory when opcode, "SBRD" is used in the FIR register. This register is provided for reading 1 or 2 bytes of data from 8- or 16-bit NAND flash. The location from where the data gets fetched can be controlled by column register (COL*n*). When NAND FIR is programmed with opcode "SBRD", IFC always reads 1/2 bytes of data irrespective of the value of BC (no. of bytes) programmed in NAND_BC register.

Also note that the read data fetched from NAND flash is only stored in this register and SRAM buffer does not get updated as this opcode is intended for register access only.

Address: 12_4000h base + 128Ch offset = 12_528Ch



IFC_NAND_MDR field descriptions

Field	Description
0–7 RDATA0	1st read data byte when opcode SBRD is used for read
8–15 RDATA1	Second read data byte when opcode SBRD is used for read (only valid for 16 bit NAND flash)
16–31 -	This field is reserved.

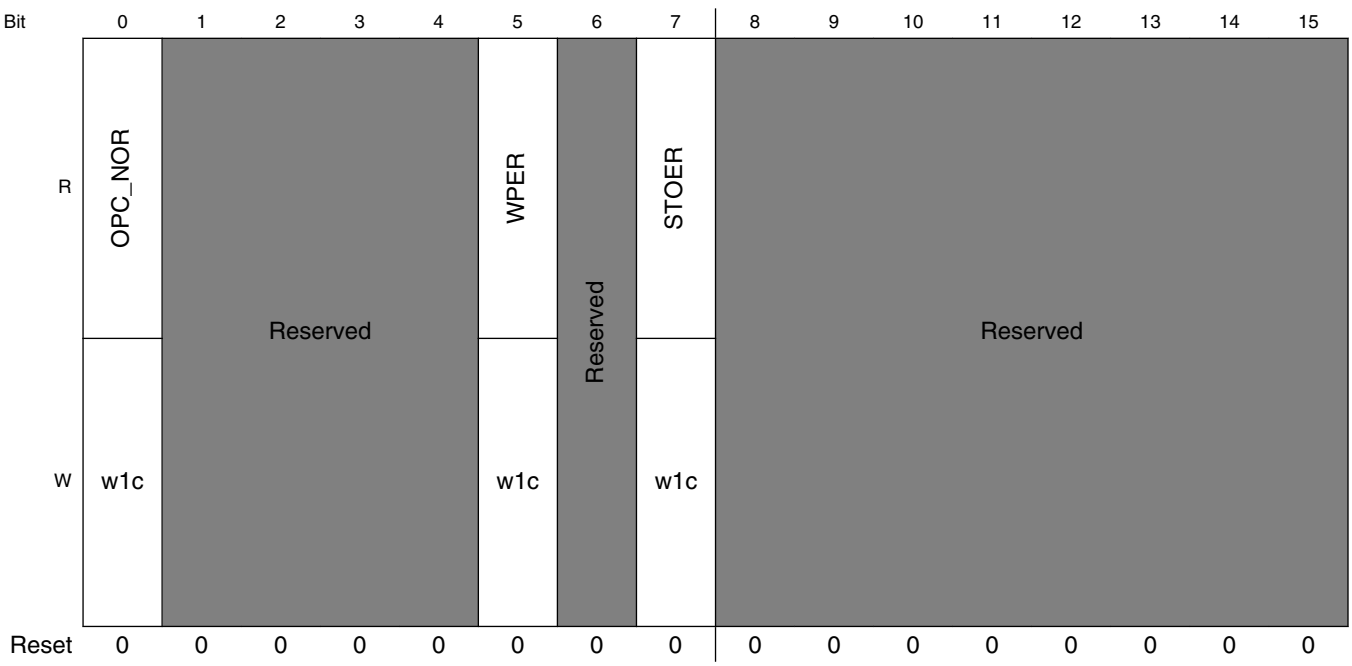
13.3.57 NOR event and error status register (IFC_NOR_EVTER_STAT)

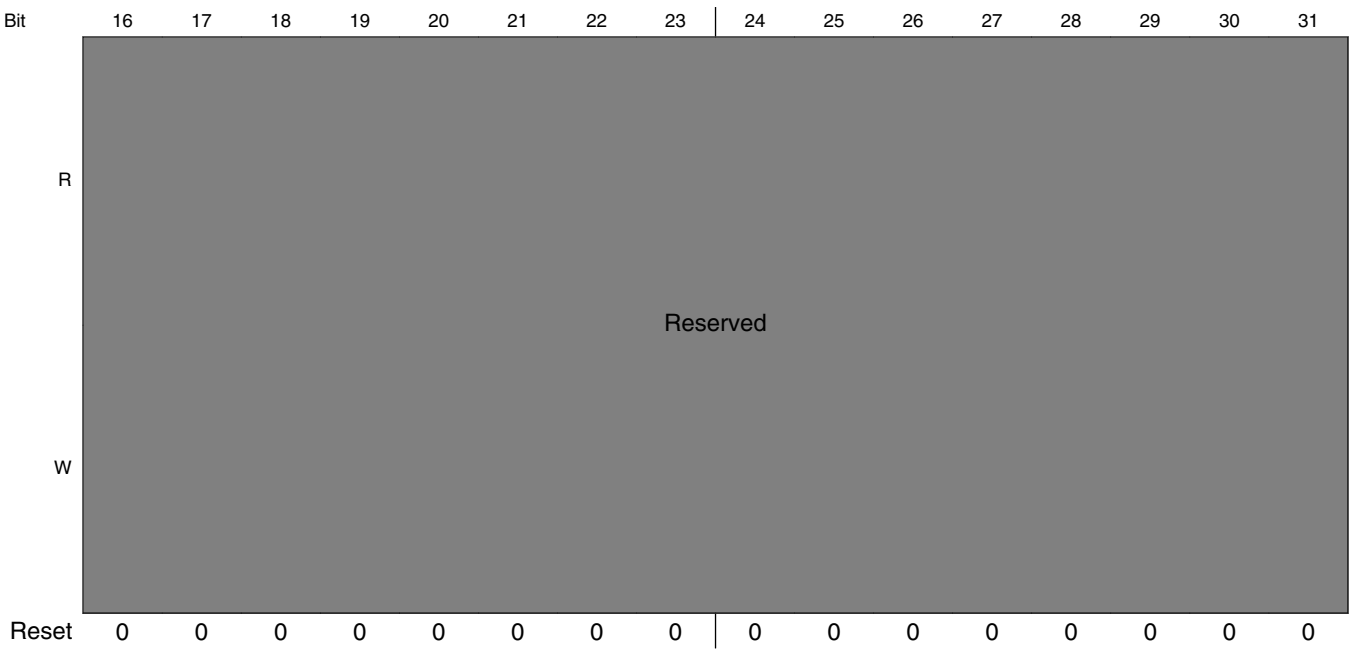
As with NAND FCM, 1 KByte memory-mapped region is allocated for the NOR-specific registers. This section describes these registers.

NOR event and error status register (NOR_EVTER_STAT) indicates the cause of an error or event corresponding to NOR flash.

The following shows the register fields. It is write-1-to-clear register.

Address: 12_4000h base + 1400h offset = 12_5400h





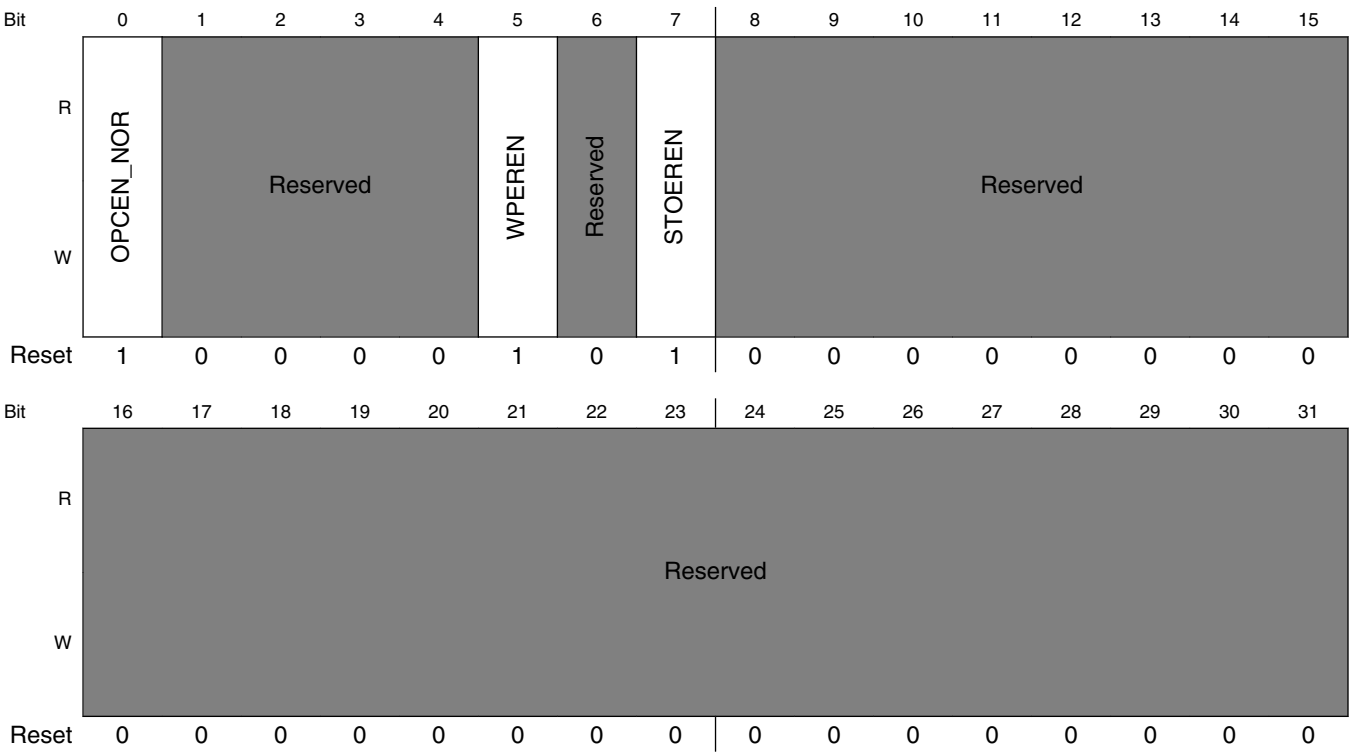
IFC_NOR_EVTER_STAT field descriptions

Field	Description
0 OPC_NOR	NOR Command Sequence Operation Complete Event Indication **NOTE:** Not valid in case of read operations which do not require a command sequence to be performed 0 NOR Command Sequence Operation not completed 1 Indicates the completion of a Command Sequence performed on the NOR flash device. A command sequence is said to have completed once all the phases (as indicated in the NORCR[NUM_PHASE]) have been sent to the NOR flash device.
1–4 -	This field is reserved.
5 WPER	Write Protect Error 0 No write protect error 1 A write is attempted to the memory bank which is write protected
6 -	This field is reserved.
7 STOER	Command sequence timeout error 0 No command sequence timeout from system side 1 Command sequence timeout from system side
8–31 -	This field is reserved.

13.3.58 NOR event and error enable register (IFC_NOR_EVTER_EN)

Event and Error Enable Register (NOR_EVTER_EN) is used to enable/disable the logging of event and error indication in the NOR_EVTER_STAT register.

Address: 12_4000h base + 140Ch offset = 12_540Ch



IFC_NOR_EVTER_EN field descriptions

Field	Description
0 OPCEN_NOR	NOR Command Sequence operation complete event enable 0 OPC_NOR Event is not enabled 1 OPC_NOR Event is enabled
1–4 -	This field is reserved.
5 WPEREN	Write Protect Error Checking Enable 0 No write protect error checking 1 Write protect error checking is enabled
6 -	This field is reserved.
7 STOEREN	Command Sequence Timeout Error Enable

Table continues on the next page...

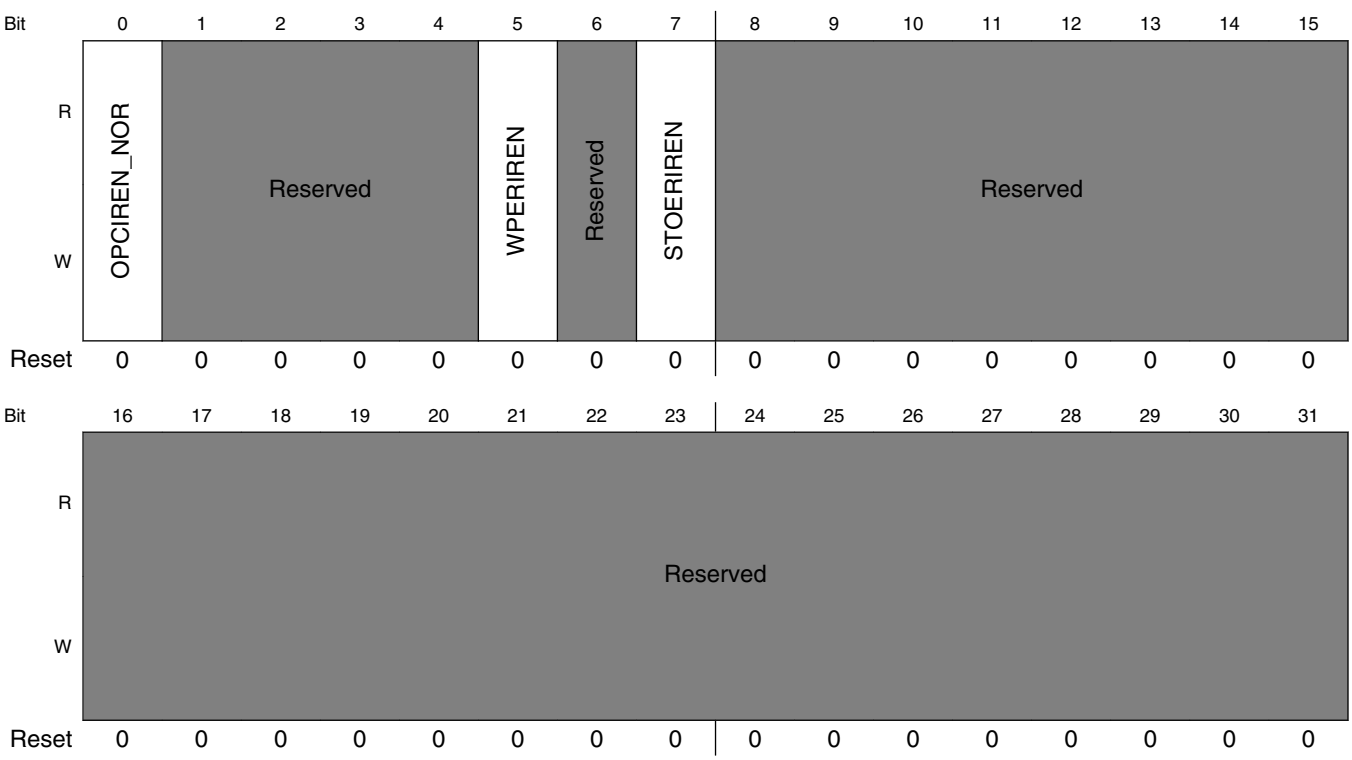
IFC_NOR_EVTER_EN field descriptions (continued)

Field	Description
	0 No command sequence timeout checking 1 Command sequence timeout error checking enable
8–31 -	This field is reserved.

13.3.59 NOR event and error interrupt enable register (IFC_NOR_EVTER_INTR_EN)

Event and Error Interrupt Enable Register (NOR_EVTER_INTR_EN) is used to enable/disable the generation of interrupt signal corresponding to error and event reporting. Software must clear pending events and errors in NOR_EVTER_STAT register before enabling the interrupts.

Address: 12_4000h base + 1418h offset = 12_5418h



IFC_NOR_EVTER_INTR_EN field descriptions

Field	Description
0 OPCIREN_NOR	NOR command sequence operation complete event Interrupt enable

Table continues on the next page...

IFC_NOR_EVTER_INTR_EN field descriptions (continued)

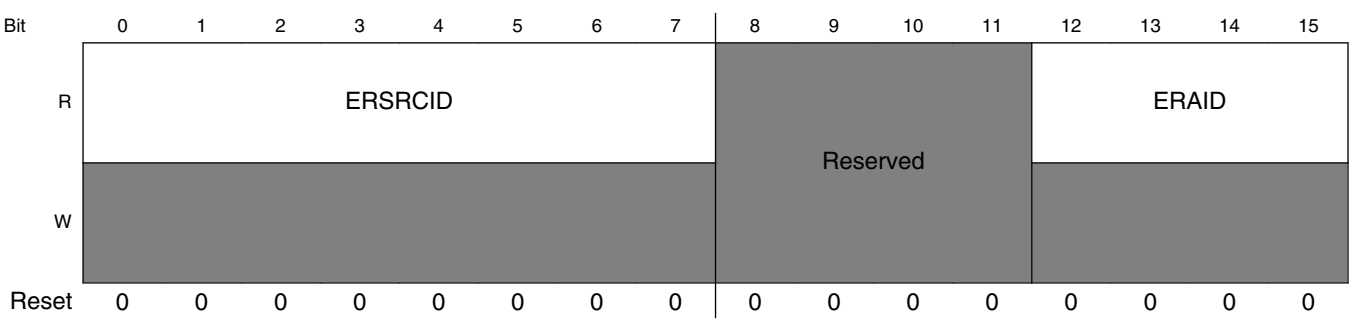
Field	Description
	0 Event Interrupt is not enabled 1 Event Interrupt is enabled
1–4 -	This field is reserved.
5 WPERIREN	Write protect error interrupt enable 0 Write protect error interrupt disable 1 Write protect error interrupt is enabled
6 -	This field is reserved.
7 STOERIREN	NOR command sequence timeout error interrupt enable 0 Command Sequence Timeout error interrupt disable 1 Command Sequence Timeout error interrupt is enabled
8–31 -	This field is reserved.

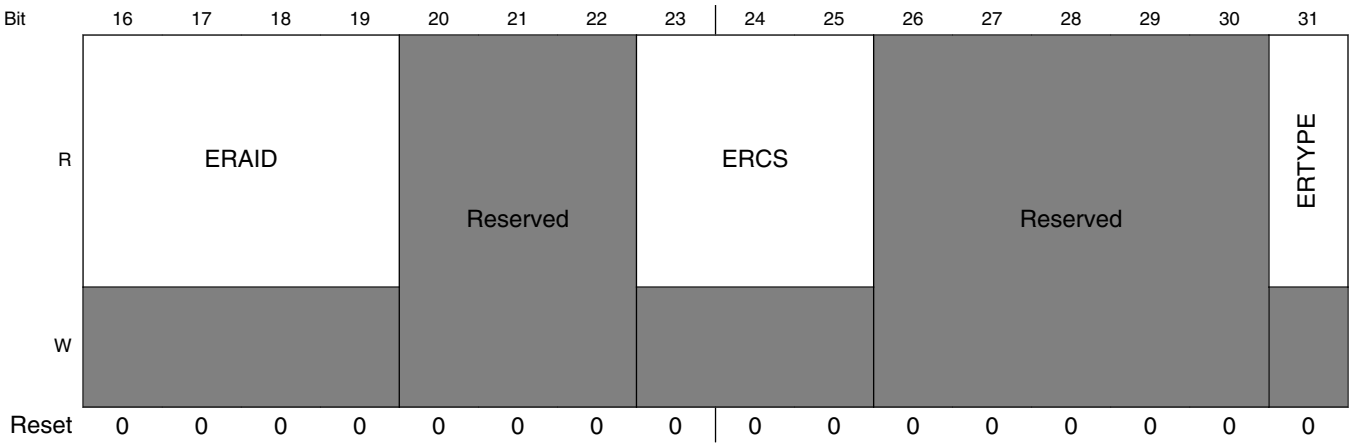
13.3.60 NOR transfer error attributes register 0 (IFC_NOR_ERATTR0)

These registers store the attribute corresponding to the first transaction on which error occured.

Transfer error attribute register 0 (NOR_ERATTR0) is used to register the transaction attributes corresponding to first error occured.

Address: 12_4000h base + 1424h offset = 12_5424h





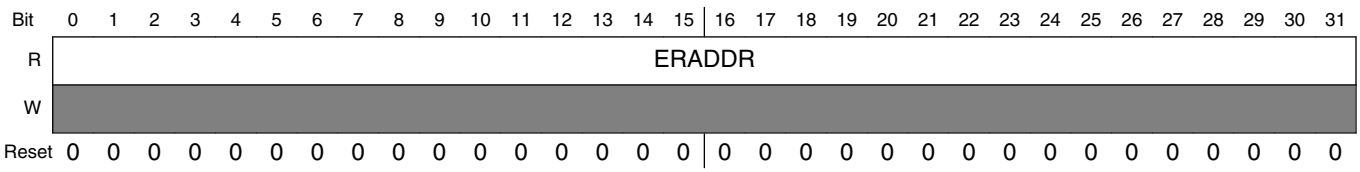
IFC_NOR_ERATTR0 field descriptions

Field	Description
0–7 ERSRCID	SRCID corresponding to error transaction
8–11 -	This field is reserved.
12–19 ERAID	AXI ID of the error transaction. This field is valid only for write protect error.
20–22 -	This field is reserved.
23–25 ERCS	Chip-select corresponding to NOR error 000 Bank0 001 Bank1 010 Bank2 . . . 111 Bank7
26–30 -	This field is reserved.
31 ERTYPE	Type of the transaction 0 Write 1 Read

13.3.61 NOR transfer error attribute register 1 (IFC_NOR_ERATTR1)

Transfer error attribute register (NOR_ERATTR1) is used to register the transaction address corresponding to first error occured.

Address: 12_4000h base + 1428h offset = 12_5428h



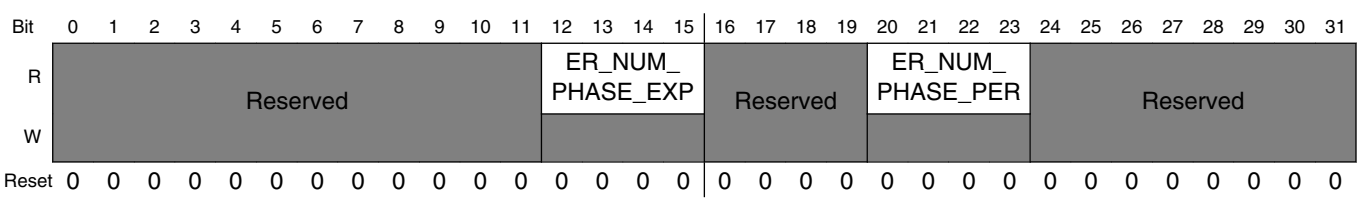
IFC_NOR_ERATTR1 field descriptions

Field	Description
0–31 ERADDR	In the case of a write protect error, this field refelcts the address corresponding to which a write cycle was received from the system side. In the case of a sequence timeout error, this field reflects the last address received from the system side

13.3.62 NOR transfer error attribute register 2 (IFC_NOR_ERATTR2)

Transfer error attribute register (NOR_ERATTR2) is used to register the transaction SRCID corresponding to first error occured.

Address: 12_4000h base + 142Ch offset = 12_542Ch



IFC_NOR_ERATTR2 field descriptions

Field	Description
0–11 -	This field is reserved.
12–15 ER_NUM_ PHASE_EXP	Number of phase expected in command sequence which timed-out by system side. **NOTE:** This attribute is valid only for sequence timeout error. 0000 1 Phase

Table continues on the next page...

IFC_NOR_ERATTR2 field descriptions (continued)

Field	Description
	0001 2 Phase ... 1111 16 Phases
16–19 -	This field is reserved.
20–23 ER_NUM_ PHASE_PER	Actual no. of command sequence phases performed on NOR flash before timeout occured **NOTE:** This attribute is valid only for sequence timeout error 0000 0 Phase 0001 1 Phase ... 1111 15 Phases
24–31 -	This field is reserved.

13.3.63 NOR control register (IFC_NORCR)

Address: 12_4000h base + 1440h offset = 12_5440h

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				NUM_PHASE				Reserved				STOCNT				Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IFC_NORCR field descriptions

Field	Description
0–3 -	This field is reserved.
4–7 NUM_PHASE	No. of Address/Data phase on device for which chip-select has to be asserted: This field is used for the NOR devices for which CS has to be asserted for multiple address data cycles. It can be used for command based NOR, where CS remains asserted during unlock cycles and actual address/data phase. For example, if the total number of phases (that is, command and address/data) to be sent corresponding to a particular operation (such as program, block erase, and chip erase) is 5, then this field should be programmed to 0100. This would indicate to the NOR FCM that the CE needs to be asserted for five consecutive commands sent from the system side (there by for a complete operation) 0000 1 phase 0001 2 phase 0010 3 phase 0011 4 phase 0100 5 phase 0101 6 phase 0110 7 phase

Table continues on the next page...

IFC_NORCR field descriptions (continued)

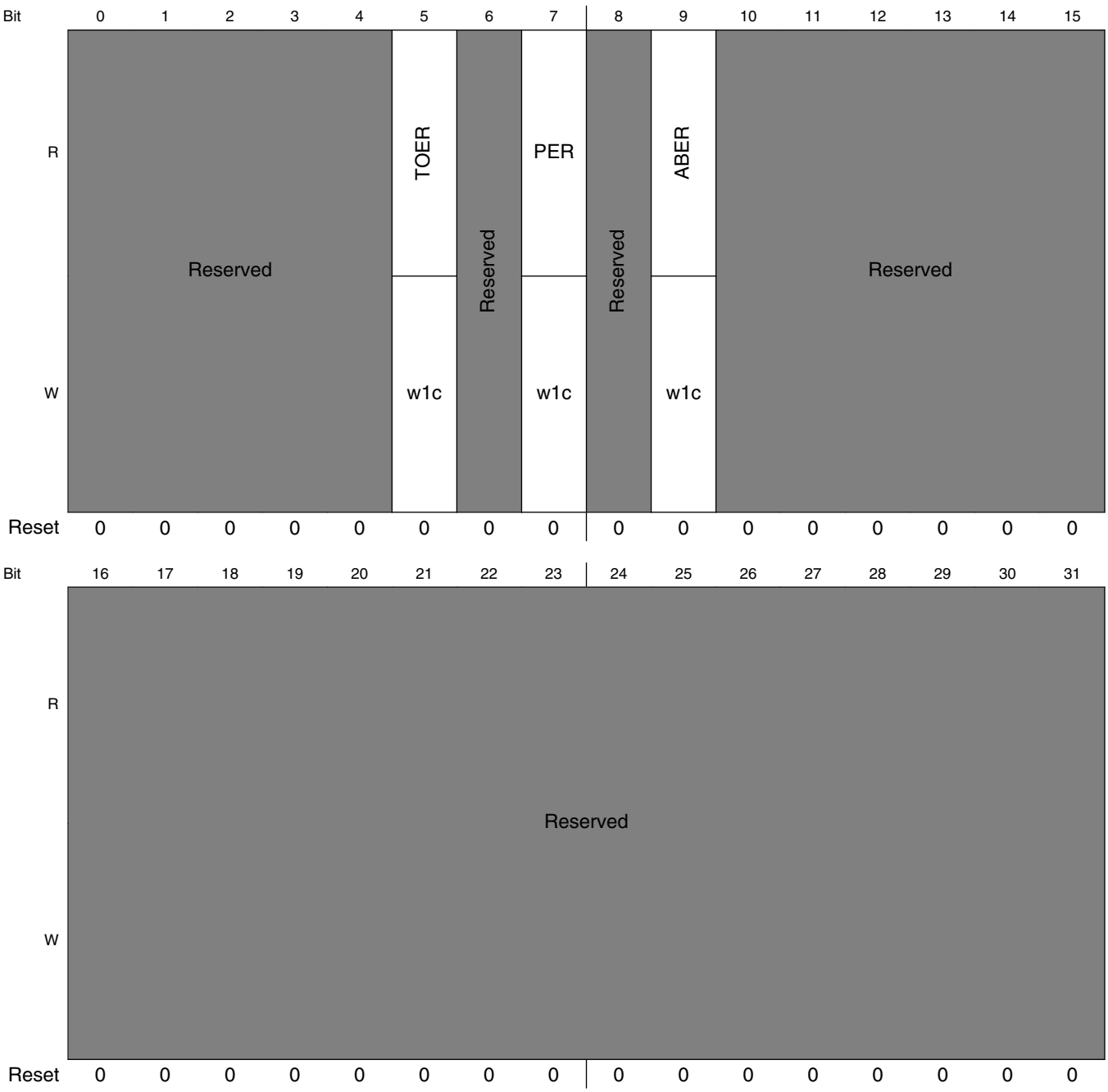
Field	Description
	0111 8 phase 1000 9 phase 1001 10 phase 1010 11 phase 1011 12 phase 1100 13 phase 1101 14 phase 1110 15 phase 1111 16 phase
8–11 -	This field is reserved.
12–15 STOCNT	Sequence Timeout Count **NOTE:** This counter is used for timeout on sequential transactions on command based NOR. 0000 256 cycles of IFC module input clock 0001 512 cycles of IFC module input clock 0010 1024 cycles of IFC module input clock 0011 2048 cycles of IFC module input clock 0100 4096 cycles of IFC module input clock 0101 8192 cycles of IFC module input clock 0110 16,384 cycles of IFC module input clock 0111 32,768 cycles of IFC module input clock 1000 65,536 cycles of IFC module input clock 1001 131,072 cycles of IFC module input clock 1010 262,144 cycles of IFC module input clock 1011 524,288 cycles of IFC module input clock 1100 1,048,576 cycles of IFC module input clock 1101 2,097,152 cycles of IFC module input clock 1110 4,194,304 cycles of IFC module input clock 1111 8,388,608 cycles of IFC module input clock
16–31 -	This field is reserved.

13.3.64 GPCM event and error status register (IFC_GPCM_EVTER_STAT)

A separate 1 KByte memory-mapped region is allocated for the GPCM specific registers.

GPCM Event and Error Status Register (GPCM_EVTER_STAT) indicates the cause of an error or event corresponding to GPCM flash devices.

Address: 12_4000h base + 1800h offset = 12_5800h



IFC_GPCM_EVTER_STAT field descriptions

Field	Description
0–4 -	This field is reserved.
5 TOER	Timeout Error **NOTE:** Timeout for Normal GPCM mode will only be observed if IFCTA_B is configured as acknowledgement signal for transaction access completion, that is, by setting 1 in CSOR*n*[WGETA] and CSOR*n*[RGETA]. Timeout error will occur if IFC is waiting for the

Table continues on the next page...

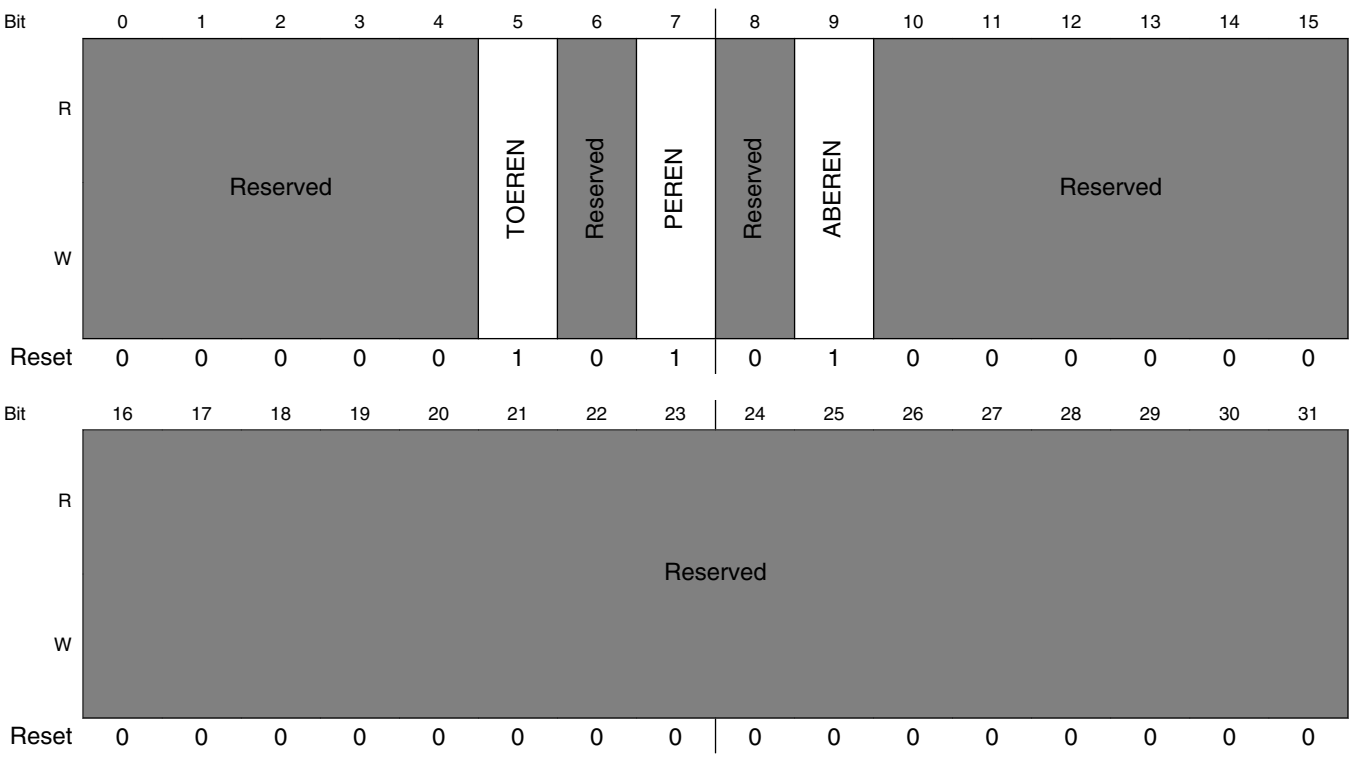
IFC_GPCM_EVTER_STAT field descriptions (continued)

Field	Description
	acknowledgement signal IFCTA_B to come and timeout counter value specified in CSOR[GPTO] counter has expired 0 No Timeout Error 1 Timeout observed for read/write transaction.
6 -	This field is reserved.
7 PER	Parity Error 0 No Parity Error 1 Parity Error for read/write transaction.
8 -	This field is reserved.
9 ABER	Abort Error. Abort for Normal GPCM mode will only be observed if access is terminated by IFCTA_B when CSOR*n*[RGETA] is programmed to 0. It is valid for read transaction and no error is reported for write transaction. 0 No abort error 1 Abort happened for the current read transaction
10–31 -	This field is reserved.

13.3.65 GPCM event and error enable register (IFC_GPCM_EVTER_EN)

Event and Error Enable Register (GPCM_EVTER_EN) is used to enable/disable the logging of event and error indication in the GPCM_EVTER_STAT register.

Address: 12_4000h base + 180Ch offset = 12_580Ch



IFC_GPCM_EVTER_EN field descriptions

Field	Description
0–4 -	This field is reserved.
5 TOEREN	Timeout error checking enable 0 No Timeout Error checking 1 Timeout Error checking is enabled
6 -	This field is reserved.
7 PEREN	Parity error checking enable. 0 No Parity Error Checking 1 Parity Error checking Enabled

Table continues on the next page...

IFC_GPCM_EVTER_EN field descriptions (continued)

Field	Description
8 -	This field is reserved.
9 ABEREN	Abort Error Checking Enable 0 No Abort Error checking 1 Abort Error checking is enabled
10–31 -	This field is reserved.

13.3.66 GPCM event and error interrupt enable register (IFC_GPCM_EVTER_INTR_EN)

Event and Error Interrupt Enable Register (GPCM_EVTER_INTR_EN) is used to enable/disable the generation of interrupt signal corresponding to error and event reporting. Software must clear pending events and errors in GPCM_EVTER_STAT register before enabling the interrupts.

Address: 12_4000h base + 1818h offset = 12_5818h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Reserved					TOERIREN	Reserved	PERIREN	Reserved	ABERIREN	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

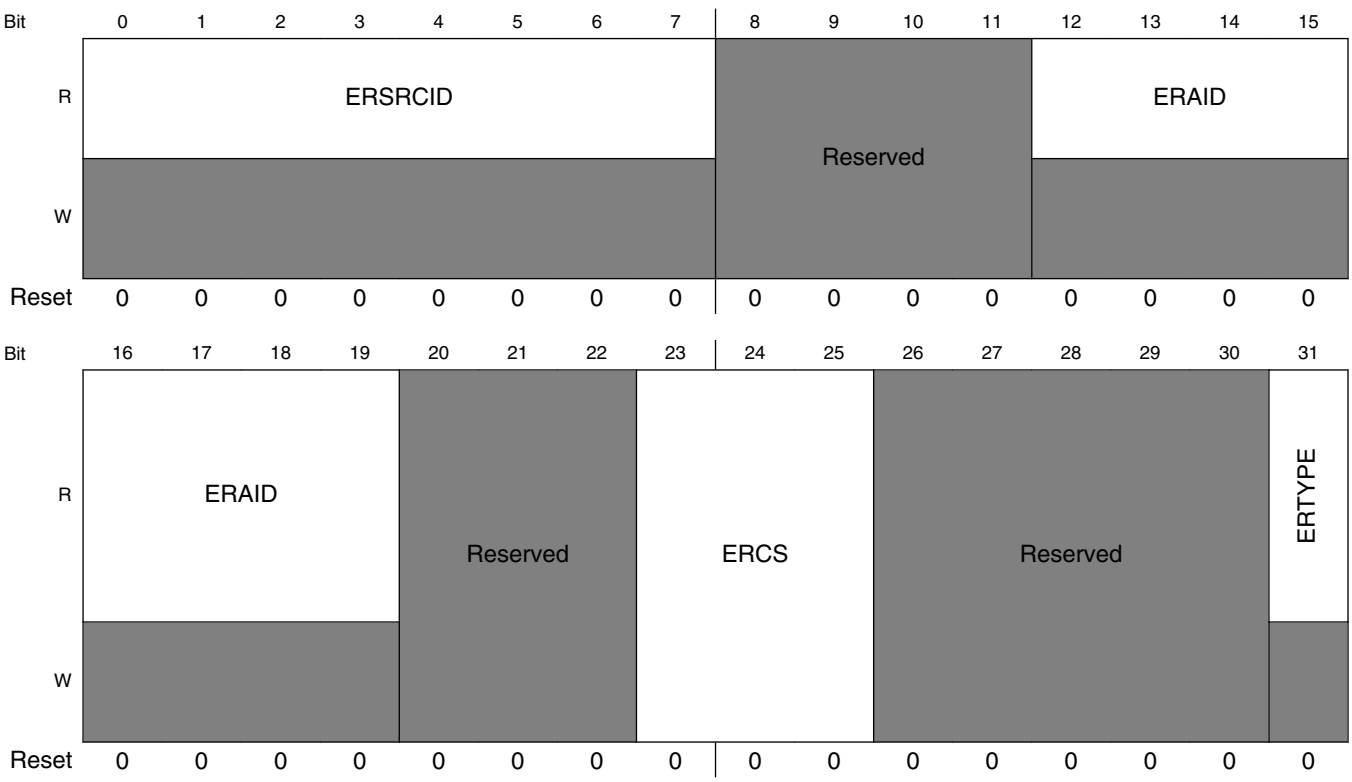
IFC_GPCM_EVTER_INTR_EN field descriptions

Field	Description
0–4 -	This field is reserved.
5 TOERIREN	Tmeout Error Interrupt Enable 0 Timeout Error interrupt disable 1 Timeout Error interrupt is enabled
6 -	This field is reserved.
7 PERIREN	Parity Error Interrupt Enable 0 Parity Error interrupt disable 1 Parity Error interrupt is enabled
8 -	This field is reserved.
9 ABERIREN	Abort Error Interrupt Enable 0 Abort Error interrupt disable 1 Abort Error interrupt is enabled
10–31 -	This field is reserved.

13.3.67 GPCM transfer error attributes register 0 (IFC_GPCM_ERATTR0)

Transfer error attribute register 0 (GPCM_ERATTR0) is used to register the transaction attributes corresponding to error occured.

Address: 12_4000h base + 1824h offset = 12_5824h



IFC_GPCM_ERATTR0 field descriptions

Field	Description
0–7 ERSRCID	SRCID corresponding to error transaction
8–11 -	This field is reserved.
12–19 ERAID	AXI ID of the error transaction
20–22 -	This field is reserved.
23–25 ERCS	000 Bank0 001 Bank1 010 Bank2

Table continues on the next page...

IFC_GPCM_ERATTR0 field descriptions (continued)

Field	Description
	. . . 111 Bank7
26–30 -	This field is reserved.
31 ERTYPE	Type of the transaction 0 Write 1 Read

13.3.68 GPCM transfer error attributes register 1 (IFC_GPCM_ERATTR1)

Transfer error attribute register (GPCM_ERATTR1) is used to register the transaction address corresponding to error occured.

Address: 12_4000h base + 1828h offset = 12_5828h

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	ERADDR																															
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

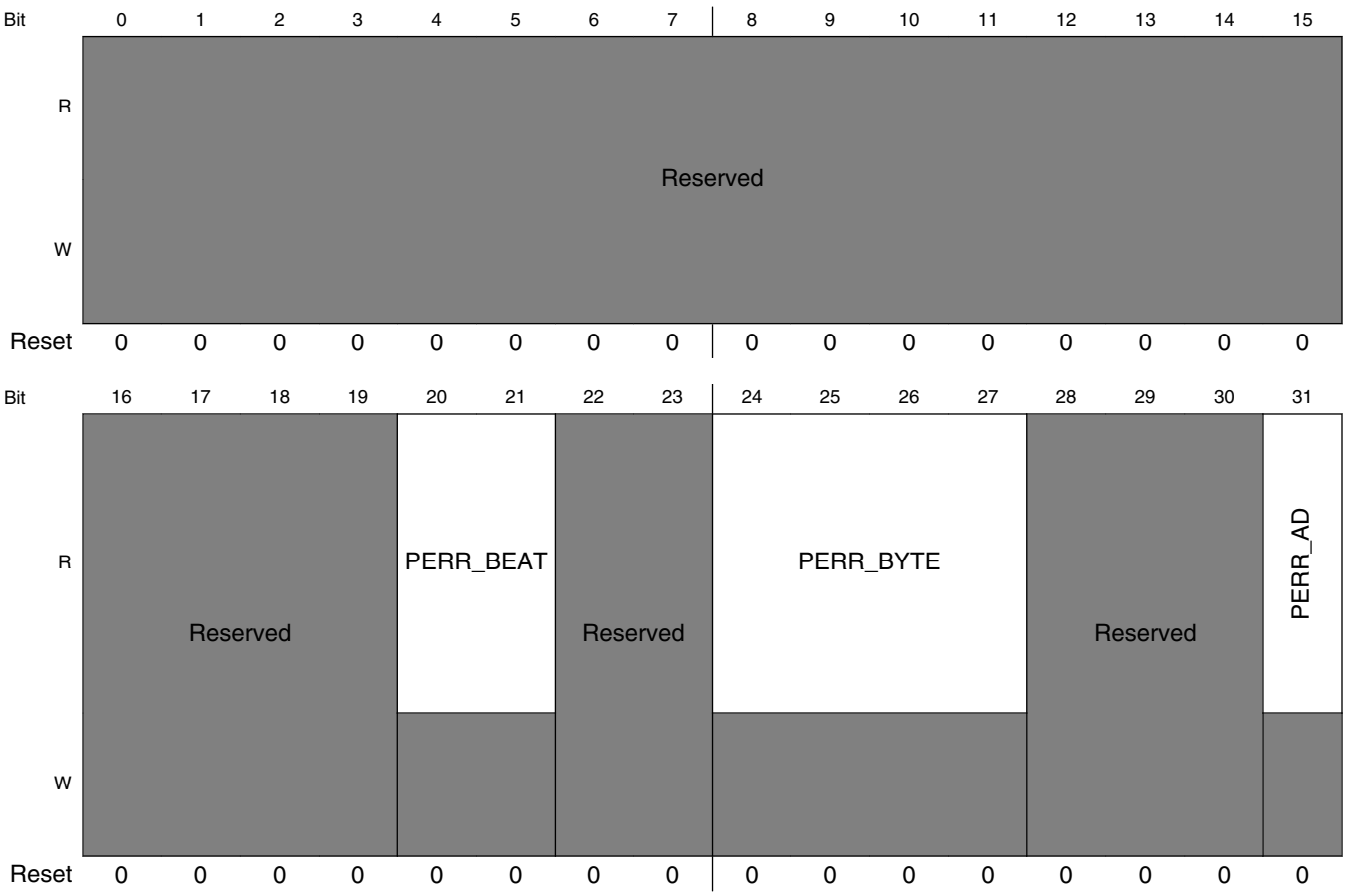
IFC_GPCM_ERATTR1 field descriptions

Field	Description
0–31 ERADDR	Address corresponding to error transaction

13.3.69 GPCM transfer error attributes register 2 (IFC_GPCM_ERATTR2)

Transfer error attribute register (GPCM_ERATTR2) is used to register the transaction attributes corresponding to error occured.

Address: 12_4000h base + 182Ch offset = 12_582Ch



IFC_GPCM_ERATTR2 field descriptions

Field	Description
0–19 -	This field is reserved.
20–21 PERR_BEAT	This gives information on which beat of address/data parity error is observed. This has value from 0 to 3. For example, if port size is 16 bit and parity error is observed on second beat, then value 1 is reported in the status register. This field is valid for Generic ASIC mode.
22–23 -	This field is reserved.

Table continues on the next page...

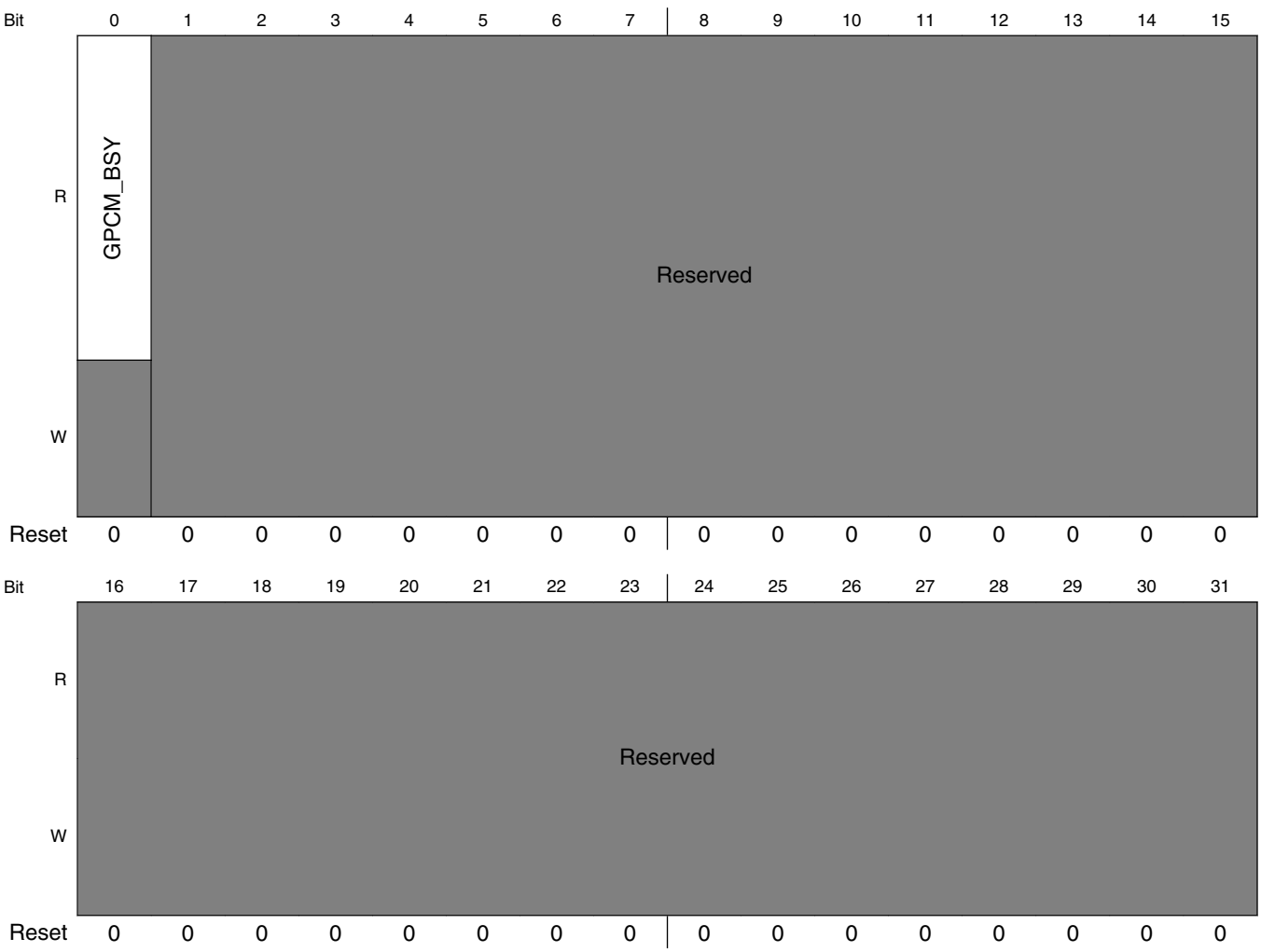
IFC_GPCM_ERATTR2 field descriptions (continued)

Field	Description
24–27 PERR_BYTE	Parity Error on byte. For GPCM there are four parity error status bit, one per byte. A bit is set for the byte that has parity error (bit 24 represents byte 0, the most significant byte lane). This field is valid for Normal GPCM mode.
28–30 -	This field is reserved.
31 PERR_AD	Parity Error reported in address or data phase. Address phase is only valid for Generic ASIC mode of GPCM. 0 Address phase 1 Data Phase

13.3.70 GPCM status register (IFC_GPCM_STAT)

This register is used to reflect the busy status of the GPCM controller.

Address: 12_4000h base + 1830h offset = 12_5830h



IFC_GPCM_STAT field descriptions

Field	Description
0 GPCM_BSY	GPCM_BSY 0 No transaction being done by GPCM controller. 1 GPCM controller is busy with transactions.
1–31 -	This field is reserved.

13.4 IFC functional description

The IFC is used to interface with external asynchronous NAND flash, asynchronous NOR flash, SRAM, EPROM, and generic ASIC devices.

To achieve this functionality, the logic is partitioned in independent, flash control machines (FCMs) in the IFC to control the timings of NAND flash, NOR flash, and GPCM separately. Eight independent chip-selects are provided, but they all share the same pins; hence, only one memory can be accessed at a time based on machine-select bits of the chip-select property register for that bank (CSPR*n*[MSEL]). If a bank match occurs, the corresponding machine (NAND, NOR, or GPCM) takes ownership of the external signals that control the access and maintains control until the transaction ends.

This figure is a functional block diagram of the flash memory interface and its signal muxing.

Figure 13-232. Flash memory interface and muxing

The IFC supports the following types of flash control machines (FCMs):

- NOR FCM provides interfacing with NOR flash memories that have a 8-/16-bit-wide data bus. NOR FCM-controlled banks are used primarily for booting (direct memory-mapped) and code storage.
- The NAND FCM interfaces to NAND flash EEPROMs with 8- and 16-bit data buses. If IFC is chosen as the booting device, after reset, NAND FCM can load boot code into SRAM buffer for execution. Following boot, NAND FCM provides a flexible instruction sequencer that allows a user-defined command, address, and data transfer sequence of up to 15 steps to be executed against a memory-mapped buffer RAM. An advance ECC algorithm (BCH codes) is implemented to correct up to 4-/8-bit errors per sector of 512 bytes.
- GPCM provides an interface to simple, low-performance, memory-mapped devices.

Each memory bank can be assigned to any of these types of machines through the machine-select bits of the chip-select property register for that chip-select.

13.4.1 General architecture

The basic architecture of the IFC allows bank selection through address decoding and address/data pin muxing.

13.4.1.1 Bank selection through address decoding

Banks can be selected via address decoding.

The defined base addresses are written to CSPR*n*[BA] and CSPR*n*_EXT[BA_EXT], while the corresponding address masks are written to AMASK*n*[AM]. Each time a local system access is requested, the internal transaction address is compared with each bank. Address decoding logic is explained in Address mask register (IFC_AMASK*n*). If a match is found on a memory controller bank, the attributes defined in the CSPR*n* and CSOR*n* for that bank are used to control the memory access. If a match is found in more than one bank, the lowest-numbered bank handles the memory access (that is, bank 0 has priority over bank 1).

13.4.1.2 Address/Data pin muxing for external address latch

There are various pin-muxing schemes available at the IFC interface.

The IFC provides muxing of the address and data bits on the same bus (AD), where the muxing is controlled by the AVD/ALE signal. There are two modes supported to supply either the address most significant bit (msb) or least significant bit (lsb) on the AD bus. Chip-select register field CSOR*n*[ADM_SHFT_MODE] determines the mode of address-data pin muxing for the chip. The pin-muxing modes are described in Mode 0 pin muxing (CSOR*n*[ADM_SHFT_MODE] = 0) and Mode 1 pin muxing (CSOR*n*[ADM_SHFT_MODE] = 1).

13.4.1.2.1 Mode 0 pin muxing (CSOR*n*[ADM_SHFT_MODE] = 0)

In this mode of muxing, the IFC supplies the most significant bit (msb) of the address bits on the AD bus. Register field CSORn[ADM_SHFT] controls the amount of address shift to align the msb of address with AD[0]. This shift can be utilized to align the msb of system address with the AD[0] signal during address phase (when AVD/ALE is asserted).

An application of this mode is an external latched-based system where the AVD/ALE signal is connected to the latch enable such that external latch latches the address msb coming through the AD bus during AVD/ALE assertion. This mode is used to connect conventional NOR devices (having dedicated address and data pins) as well as GPCM-based interfaces.

In this mode, system address is left shifted by CSORn[ADM_SHFT] shift value and assigned to AD[0:31]. The address msb will be assigned to data bus msb (AD[0]). The shifted address can be latched by external latch at the falling edge of AVD/ALE.

The system address directly gets assigned to the IFC address bus (ADDR [0:31]) irrespective of the ADM_SHFT value. The least significant bit (lsb) can be retrieved from the ADDR bus. ADDR[31] will carry the lsb of the system address.

To interface with a x16 NOR device, the address lsb must be left unconnected on the board.

For a chip that has a 16-bit AD bus and an 11-bit ADDR bus, which needs to be interfaced with a memory of 8-bit port width and 128 MB memory size (requiring a 27-bit address), the following figure shows how system address bits are placed during IFC address phase (AVD/ALE assertion) with a CSORn[ADM_SHFT] value of 5.

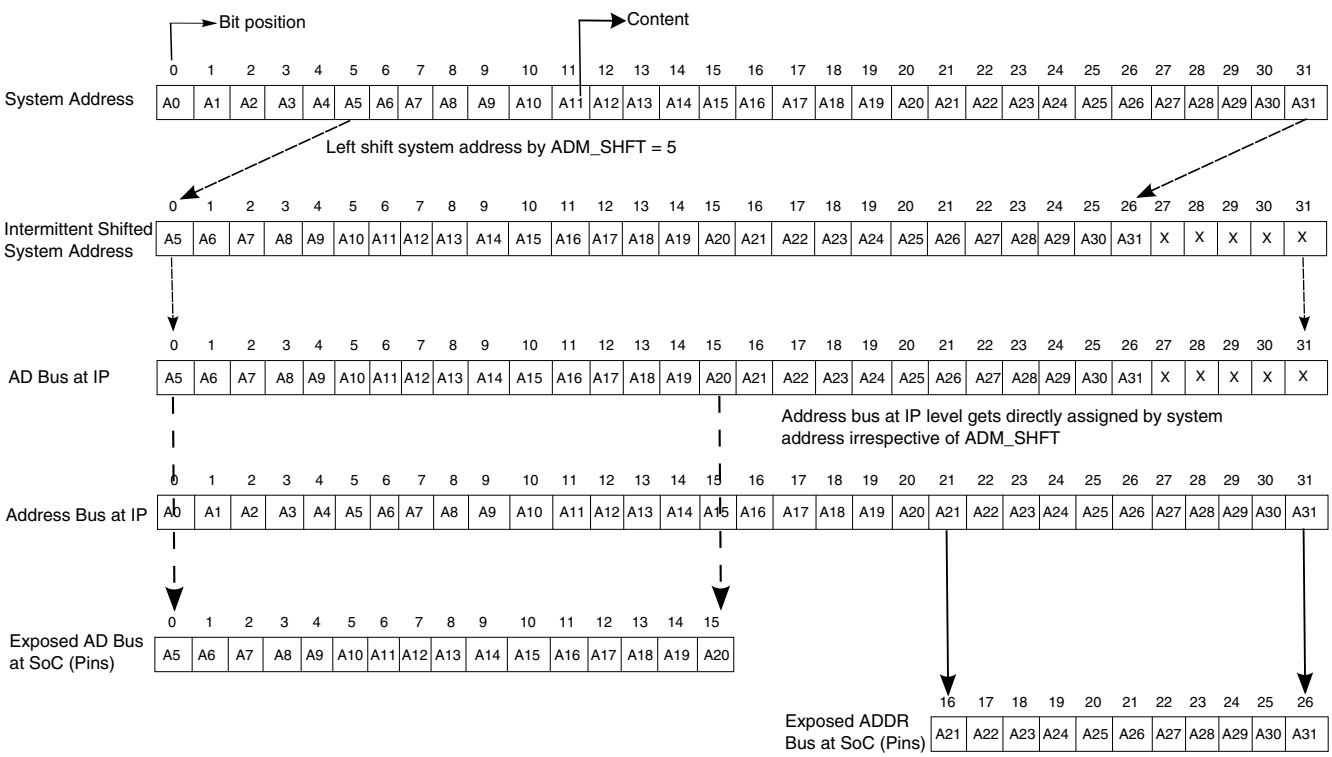


Figure 13-233. System address assignment with CSORn[ADM_SHFT]= 5 for ADM MODE 0

These figures show a x8 and x16 memory connection for the configuration given above, where the shift value is 5.

NOTE

These figures are intended to be examples only and may show ADDR pins that are supported.

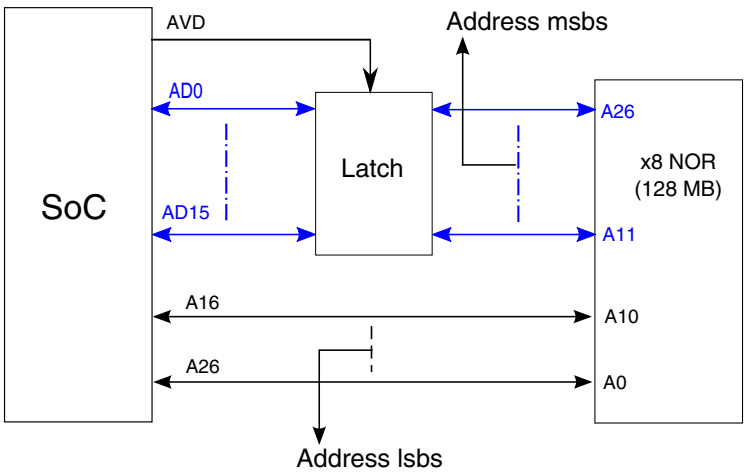


Figure 13-234. Connection of x8 ADM NOR for ADM MODE 0

For a x16 memory connection, since the memory expects a 26-bit word address, the address lsb is left unconnected at the board to supply the word address.

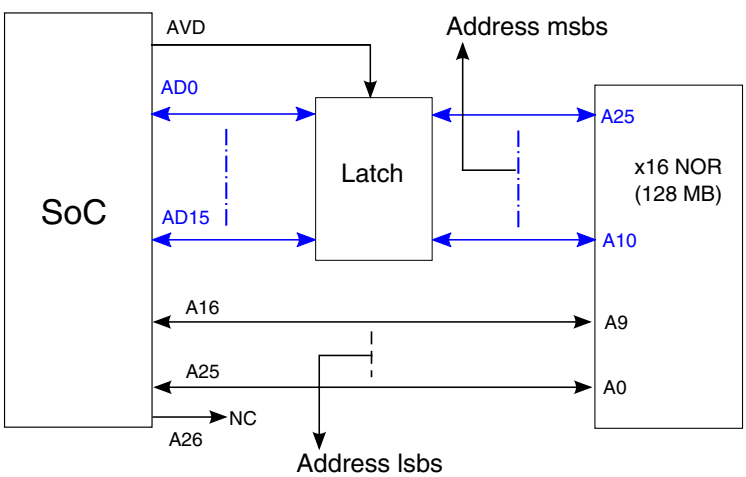


Figure 13-235. Connection of x16 ADM NOR for ADM MODE 0

13.4.1.2.2 Mode 1 pin muxing (CSOR*n*[ADM_SHFT_MODE] = 1)

In this mode of muxing, the IFC supplies the address least significant bit (lsb) on the AD bus. CSORn[ADM_SHFT] controls the address shift to align the system address lsb with AD[0]. In this mode, the ADDR bus carries the address most significant bit (msb). This mode is used to interface address data multiplexed (ADM) NOR devices (internal latch-based).

Connection in this mode should be treated as a special case since the board connectivity is in decrementing (reverse) bit order, as shown in following examples.

This is done to simplify the board connection for the ADM NOR as these devices expect the lsb to get latched at their internal latch with the assertion of AVD_B. In this mode, the system address remains the same and the IFC internally rearranges (based on the port size) the data bus in order to maintain the same data image for both the conventional NOR (external latched-based) as well as the ADM NOR (internal latch-based). This data rearrangement is performed internally within the IFC during both the read and write phases because the board connection for ADM NOR is opposite of the conventional (non-ADM) NOR devices. This mode is valid only for NOR mode of operation and not for GPCM.

Unlike in CSOR*n*[ADM_SHFT_MODE] = 0, there is no need to leave the ADDR lsb unconnected on the board to supply word address to memory for x16- and x32-bit memories. The reason is that the lsb are now carried by the AD bus, which cannot be left unconnected since they carry the data during the data phase. Therefore, the logic is implemented in such a way that the IFC does a necessary address shift internally to supply word addresses to memory.

Example 1: The chip has exposed a 32-bit AD bus and x8 ADM NOR of 128 MB is connected through it.

In this example, ADM_SHFT does not have any role as the chip has not exposed the ADDR bus and all address bits are available through the AD bus.

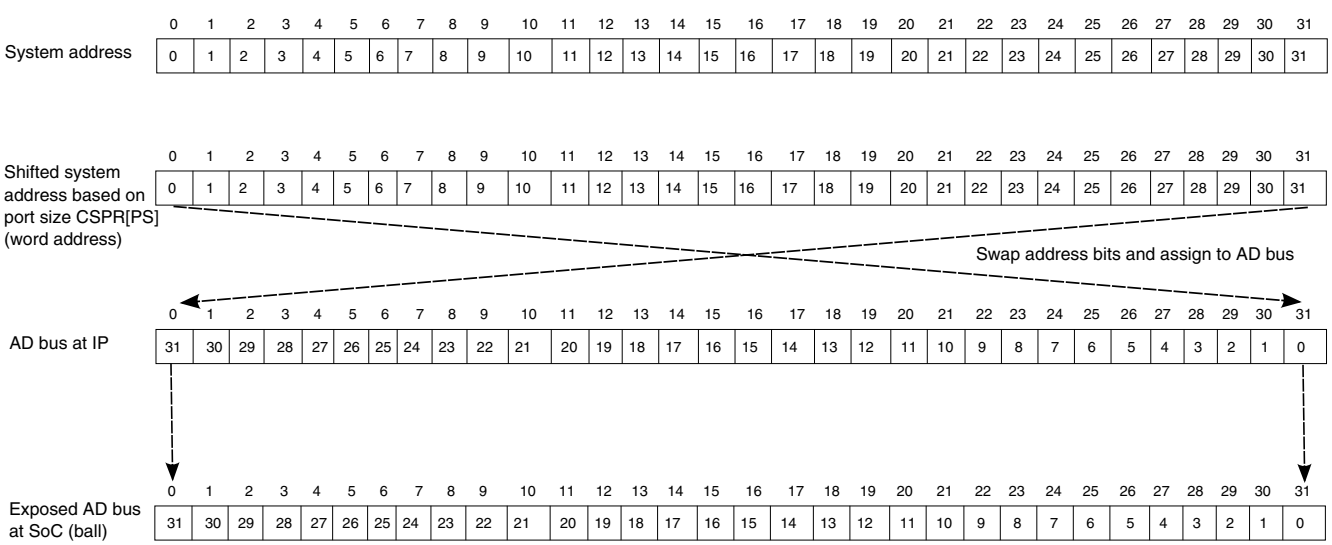


Figure 13-236. Example 1 - System address assignment to the AD bus for ADM MODE 1

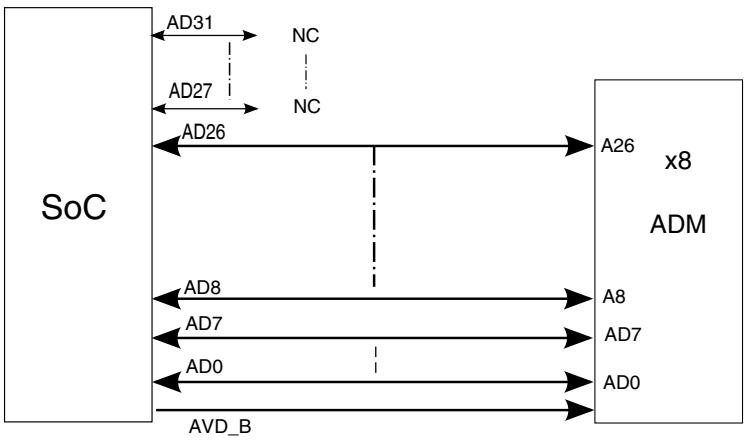


Figure 13-237. Example 1 - Connection of x8 ADM NOR for ADM MODE 1

NOTE

The board connections are in decrementing (reverse) bit order for the ADM NOR only.

Example 2: The chip has exposed a 32-bit AD bus and x16 ADM NOR of 128 MB is connected through it.

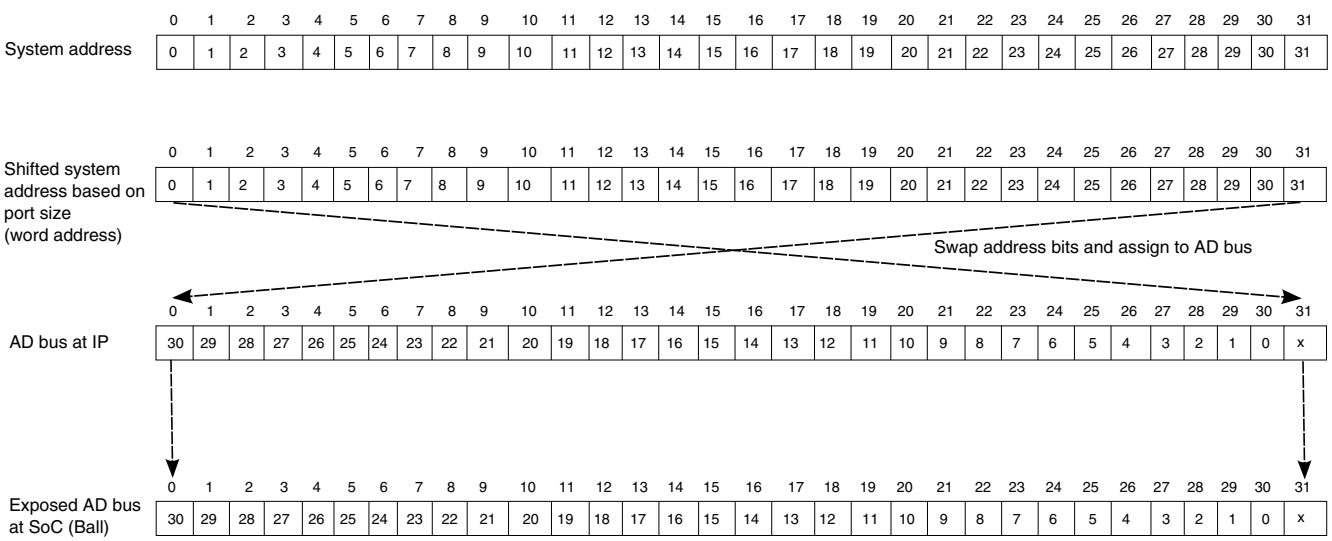


Figure 13-238. Example 2 - System address assignment to the AD bus for ADM MODE 1

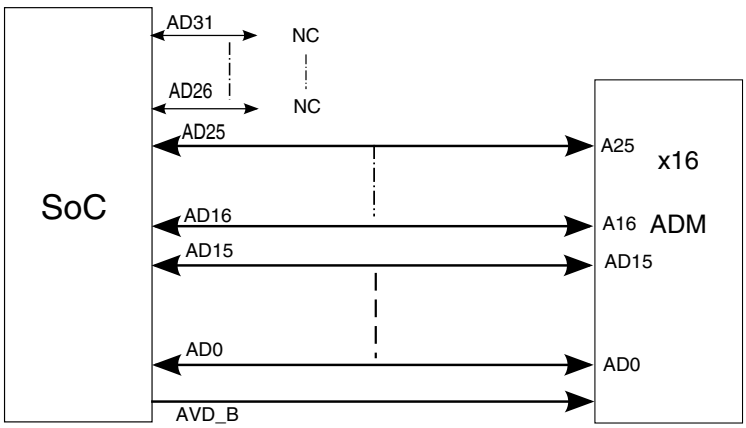


Figure 13-239. Example 2 - Connection of x16 ADM NOR for ADM MODE 1

Based on the port size, the address is generated such that the proper word address goes to the memory, unlike conventional NOR (ADM MODE 0). The lsb cannot be left unconnected as they are carried by AD[0] and is also used in the data phase.

Example 3: The chip has exposed a 16-bit AD bus, 11-bit ADDR bus, and x8 ADM NOR of 128 MB is connected through it.

In this example, ADM_SHFT=5 is required to align the system address msb with the right-most ADDR index as shown in the figure below.

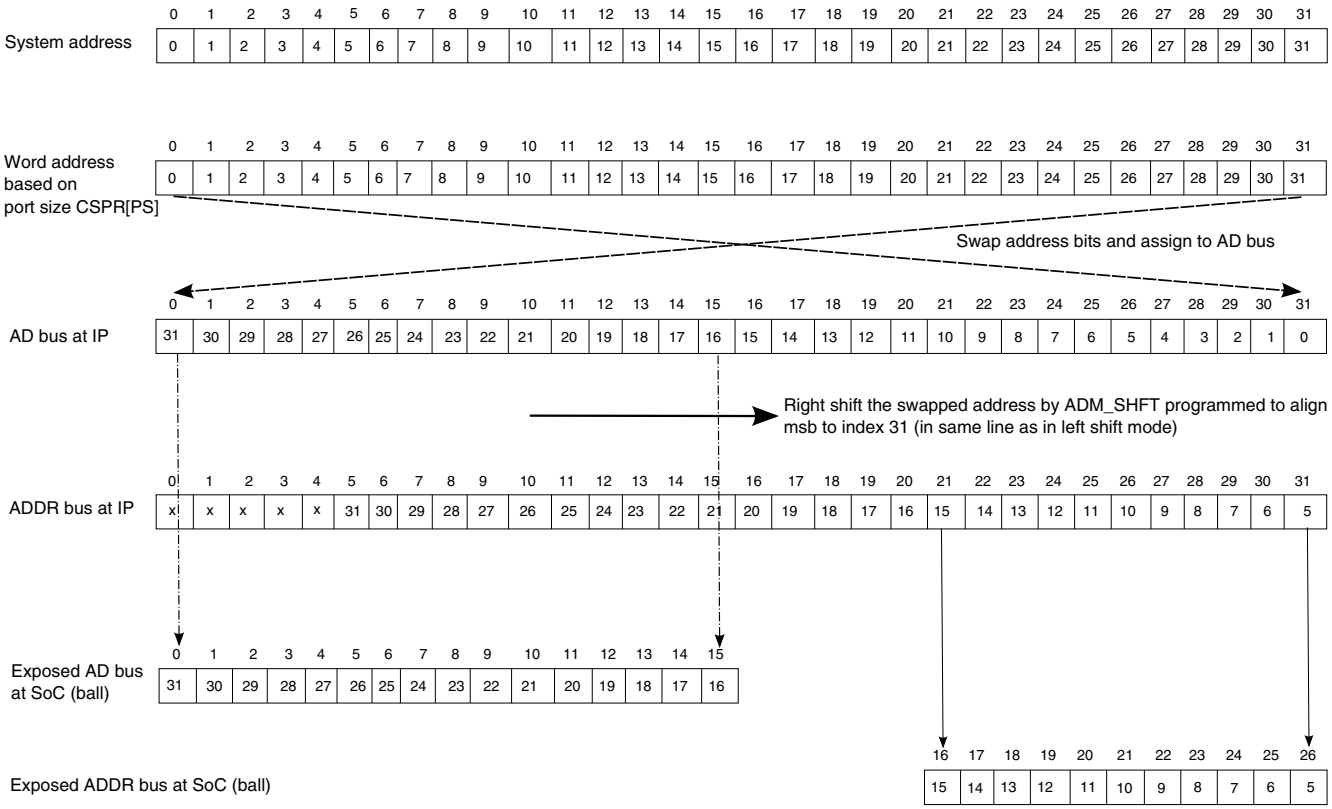


Figure 13-240. Example 3 - System address assignment to the AD and ADDR bus for ADM MODE 1

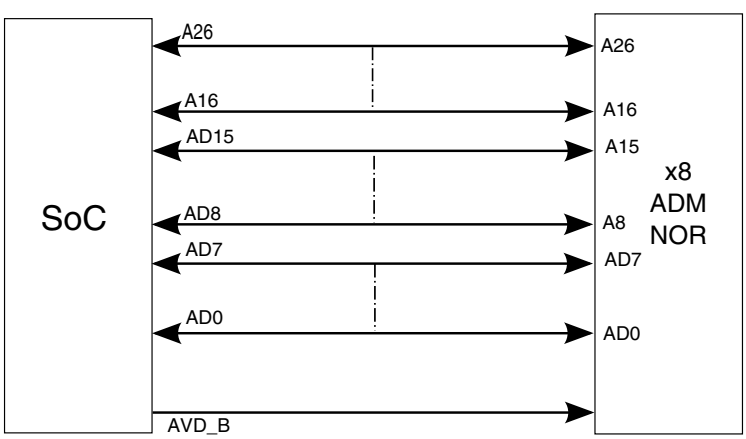


Figure 13-241. Example 3 - Connection of x8 ADM NOR for ADM MODE 1

Example 4: The chip has exposed a 16-bit AD bus, 11-bit ADDR bus, and x16 ADM NOR of 128 MB is connected through it.

In this example, ADM_SHFT=5 is required to align the system address msb with the right-most ADDR index as shown below.

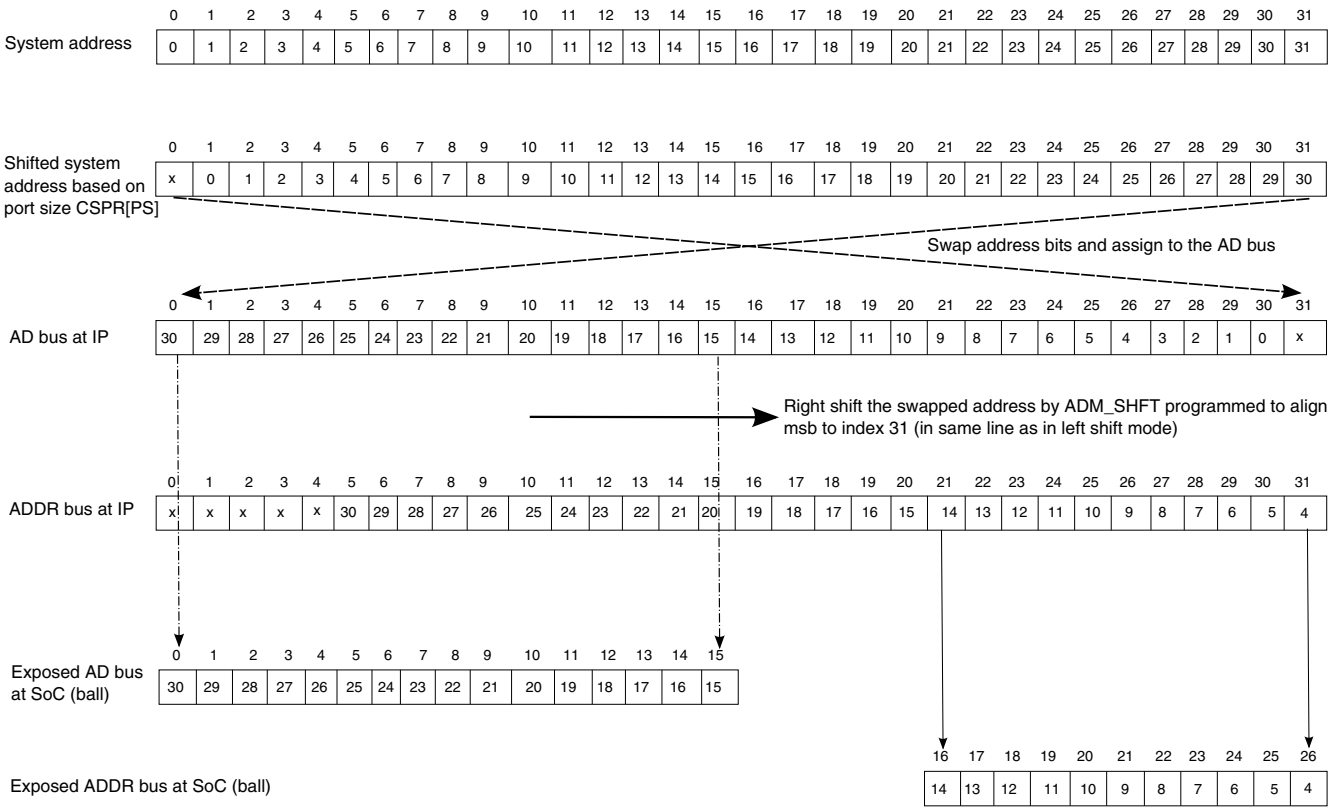


Figure 13-242. Example 4 - System address assignment to AD and ADDR bus for ADM MODE 1

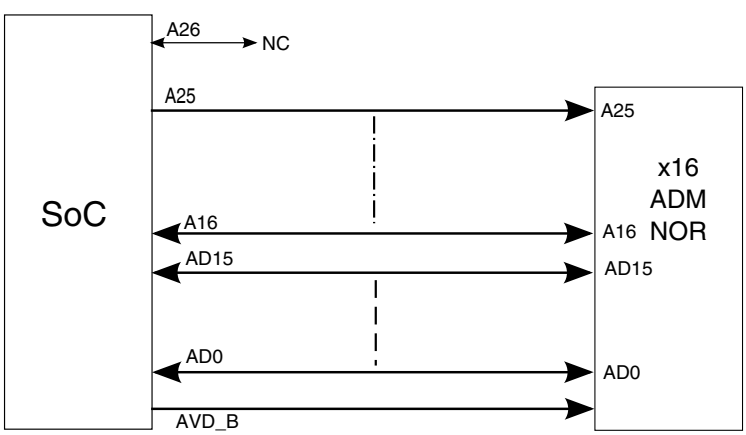


Figure 13-243. Example 4 - Connection of x16 ADM NOR for ADM MODE 1

NOTE

Timing of AVD and AVD_B is same. Both are generated by the same logic except they are opposite in polarity. If the chip is only exposing AVD and not exposing AVD_B, then it can be generated by using an on-board inverter and can be used for interfacing with ADM NOR.

13.4.1.3 Programming model for flash interface timing

The IFC follows a counter-based approach to generate the required timings on the flash side.

Programming registers (FTIM*n*) are provided that correspond to the various flash timing parameters. The values to be programmed in these registers are in terms of the number of IFC input clock-cycles derived by the following formula:

Timing counter value = [{Flash Timing Parameter(ns) x IFC module input clock frequency (MHz)} + 1000 - 1] / 1000

In this approach, counters running at the IFC module input clock generate the appropriate values for the flash interface signals depending on the value programmed in the Timing Registers (FTIM0-FTIM3). These timing registers must be programmed per the values specified in the flash device datasheet mapped to the waveforms shown in NAND asynchronous mode timings and Simple asynchronous NOR timings.

13.4.1.4 Booting methods

Booting can be performed from NAND or NOR flash on the IFC.

- NOR is directly memory-mapped, so booting from NOR can be done as simple read operations without any special arrangements.
- NAND is not directly memory-mapped, so the boot code is loaded first into an SRAM buffer. From there, the code can be executed. See NAND flash control machine for more information about booting from NAND.

The flash timing registers (Flash Timing Register 0 for Chip-Select *n* - NAND Flash Mode (IFC_FTIM0_CS*n*_NAND) through Flash Timing Register 3 for Chip-Select n - NAND Flash Mode (IFC_FTIM3_CS*n*_NAND)) that control chip-select 0 have default values corresponding to the slowest-mode devices to guarantee boot.

Reset initialization is performed by the reset controller and the default configuration is determined by cfg_rcw_src and/or RCW settings.

13.4.1.5 Software reset handling

Software reset requires that software read and clear specific registers.

Software follows these steps as part of software reset handling:

1. Read all the event and error status registers.
2. Clear the NANDSEQ_STRT register.
3. Clear the NAND_AUTOBOOT_TRGR register.
4. Apply soft reset.

After the soft reset has been applied, the hardware automatically clears all the event and error status registers. None of the other registers are reset. Therefore, the initial configuration done by the software is preserved.

13.4.1.5.1 System bus handling

Any pending transactions on the system bus are terminated with a valid response.

During software reset assertion, providing a good response that corresponds to the pending transaction on the IFC helps prevent a corresponding interrupt. The core expects an error interrupt whenever there is an error response from the system bus. Therefore, in the case of a software reset and an error response is provided, the core does into a hang state, because there is no corresponding interrupt. If there are pending read transactions, the garbage data is supplied back; if there is a pending write transaction, the data is not written in the memory.

The hardware does not allow the software to clear the soft reset bit until all pending transactions on the system bus have been gracefully terminated (with a good response). When there are no more transactions pending on the system bus, the hardware allows the software to clear the soft reset bit.

13.4.1.5.2 Flash interface handling

All the state machines (NAND, NOR, GPCM, and GASIC) are reset to the idle state.

All related FIFOs and registers are flushed, so the flash interface would be in the idle state.

13.4.1.6 Data buffer control (BCTL)

The IFC provides a data buffer control signal (BCTL) that can be disabled (remain high) by setting CSOR*n*[BCTLD]. BCTL should be used to signify the write direction when high.

NOTE

BCTL is recommended to be used as a static value; it cannot be programmed on the fly.

If the access is a write, BCTL remains high for the whole duration. However, if the access is a read, BCTL is negated (low) so that the memory device is able to drive the bus. Note that the default (reset and bus idle) value of BCTL is also high.

While accessing the slow memories, the data driven by the flash is available on the bus after deassertion of read enable. To avoid the bus contention, BCTL remains low (read mode) for the time defined by the CSORn[TRHZ] field.

Apart from CSORn[TRHZ], another signal timing requires attention during the external buffer, and that is buffer turn-around. The external buffer takes time to reverse the direction of the shared I/O bus. This situation is more relevant when a read is followed by a write. To handle this, GCR[TBCTL_TRN_TIME] is defined, which represents the time for which BCTL remains high before starting another flash access. The timings are shown in the following figures.

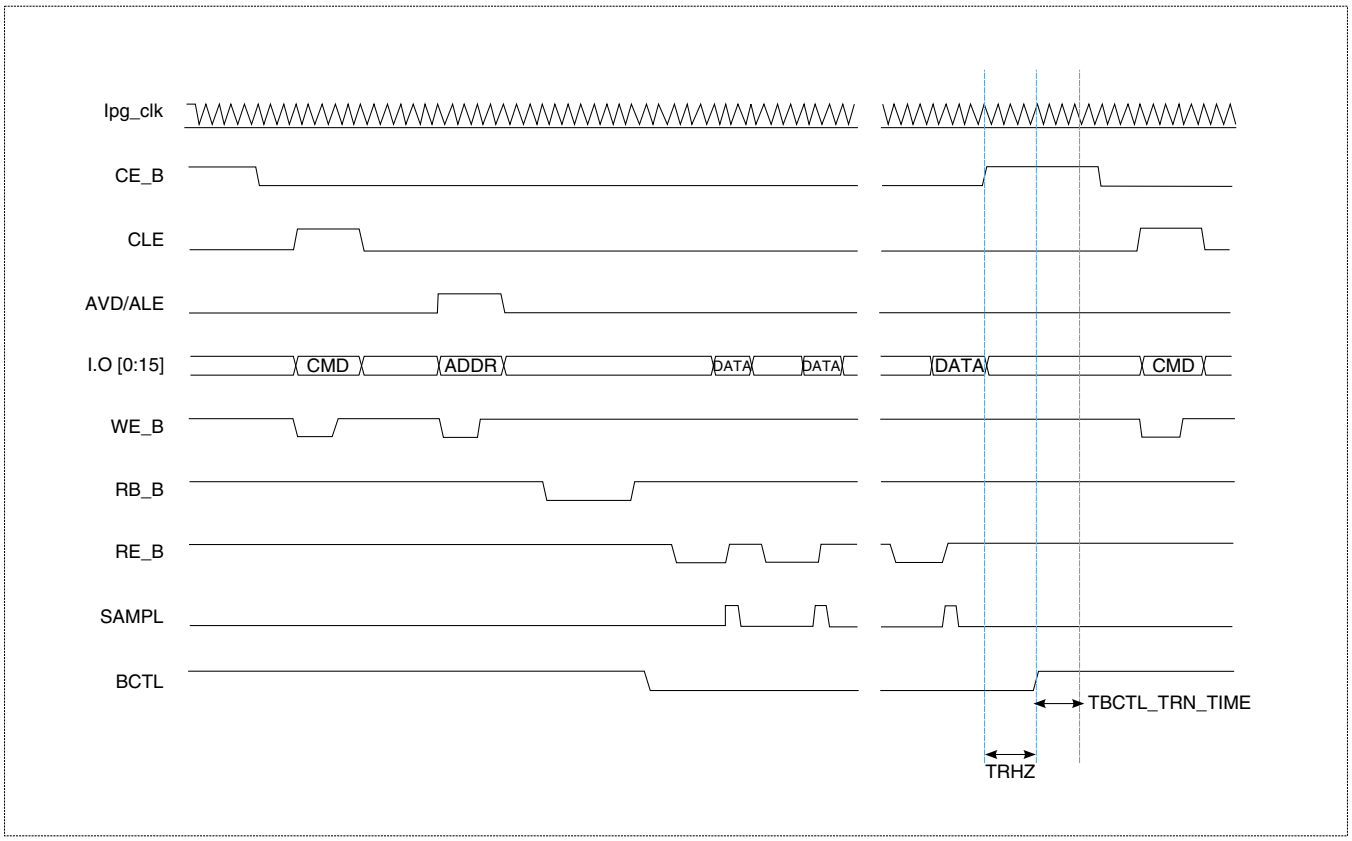


Figure 13-244. BCTL signal in NAND read followed by any other operation

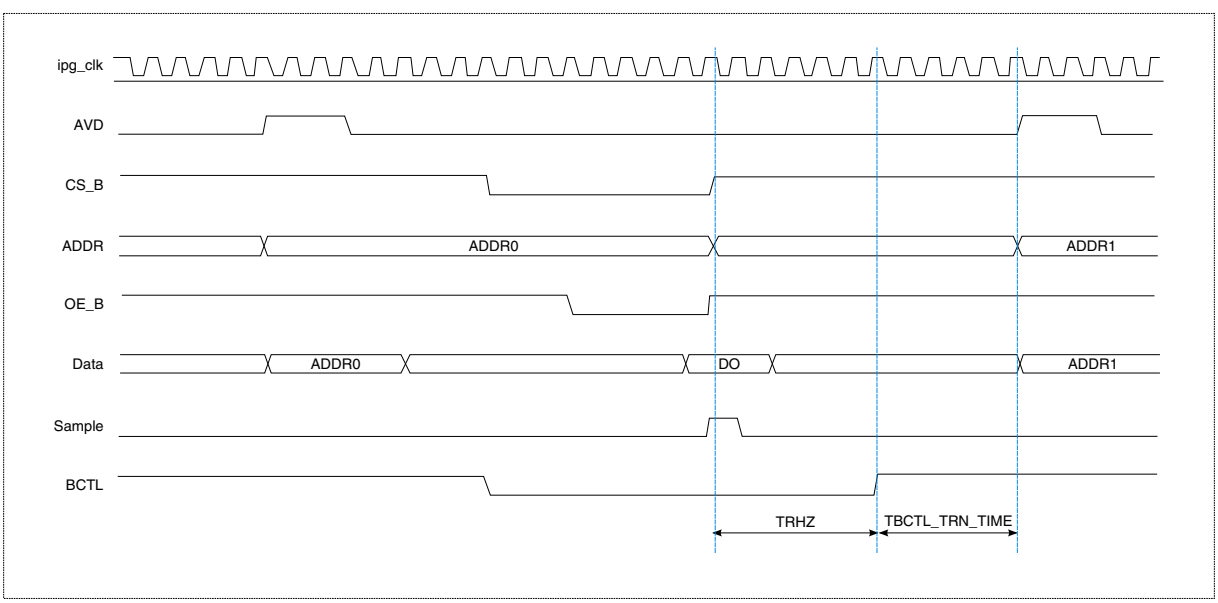


Figure 13-245. BCTL signal in NOR read followed by any other operation

13.4.1.7 External transceiver enable (TE)

This signal is used to enable/disable the external transceiver depending on whether a slow/fast memory is connected to the bank servicing the current request.

If various fast/slow memories are connected to different banks of the IFC, the AD bus may become loaded due to sharing. Such a scenario results in poor rise/fall times and therefore limits the speed of operation of fast memories given their strict timing requirements. To address this issue, fast memories should be segregated from the slower memories using an external transceiver that is connected to the AD bus. In these scenarios, the IFC provides the configurable transceiver enable (TE) pin.

13.4.1.7.1 Transceiver enable during boot

The IFC can be used for booting after reset.

In such cases, boot code is fetched from NV memories, which are inherently slower (for example, NOR/NAND flash devices). Therefore, the external transceiver should be enabled by default for booting.

The IFC can obtain the cfg_ifc_te value, which is used during booting, in the following ways:

- By sampling the TE pin.

  In this scenario, the TE pin acts as an input during reset. During this time, the external transceiver should not drive anything onto the AD bus (that is, keep it tristated). This is accomplished by connecting a weak pull up/down resistor on the TE pin so that the external transceiver is disabled. With rcw_load/boot_load, the IFC samples the TE pin in order to know the polarity of external transceiver's enable pin. With rcw_load/boot_load, the value stored in CSPR0[TE] as the default value will be opposite of the polarity of external transceiver's enable pin. Therefore, the polarity of external transceiver's enable pin is configurable with the help of pull up/down resistors.

Table 13-243. Programming of BRn[TE]

Transceiver Enable (TE) Input	Value of BRn[TE]	
	Slow Memory	Fast Memory
Active low	0	1
Active high	1	0

- By the chip.

In this scenario, the IFC samples the cfg_ifc_te signal from the chip. The IFC registers the inverted value of por_cfg_te with rcw_load/boot_load and drive the same value at the TE pin output.

13.4.1.7.2 Transceiver enable post-boot

After booting, and during normal read/write transactions, the TE pin acts as an output pin.

The configuration of this pin is done on a per-bank basis using CSPRn[TE] bits:

- If a transaction hits bank n, the logic specified by CSPRn[TE] appears on the TE pin when CS*n* gets selected by a flash interface arbitration.
- If no CS*n* is selected, the default (board tied) value appears on the TE pin. (By default, the transceiver is disabled.)

This table shows how CSPR*n*[TE] should be programmed, depending on polarity of the transceiver's enable pin and type of memory connected to that particular bank.

Table 13-244. Programming of CSPRn[TE]

External transceiver input	Value of CSPRn[TE]	
	Slow memory	Fast memory
Active low	0	1
Active high	1	0

13.4.1.7.3 Transceiver enable example scenario

In this scenario, three chip-selects are used.

The connections are as follows:

- CS0: Slow device
- CS1: Fast device
- CS2: Slow device

Because the transceiver used in this scenario has an active-low enable (TOE), the programming of the CSPRn[TE] should:

- Be 0 when slow device is connected to the corresponding bank (to enable the transceiver).
- Be 1 when fast device is connected to the corresponding bank (to disable the transciver).

Therefore, configuration would be as follows:

- CSPR0[TE] = 0
- CSPR1[TE] = 1
- CSPR2[TE] = 0

13.4.2 NAND flash control machine

The NAND flash control machine (FCM) provides an interface to parallel-bus NAND flash devices.

13.4.2.1 NAND flash asynchronous mode

For connections between an 8-bit port size NAND flash EEPROM and the IFC in FCM mode, commands, address bytes, and data are all transferred on AD[0:7] with WE_B asserted for transfers written to the chip, or RE_B asserted for transfers read from the chip. IFC signals CLE and AVD/ALE to determine whether writes are of the type command (only CLE asserted), address (only AVD/ALE asserted), or write data (neither CLE nor AVD/ALE asserted).

For connection to a 16-bit NAND flash EEPROM to IFC in FCM mode, commands, address bytes, and the high byte of data appear on AD[0:7], whereas the low byte of 16-bit data is placed on AD[8:15] during read and write data transfers.

13.4.2.2 Write protect

NAND/NOR flash devices come with an input pin, write-protect (WP_B), used to protect the flash data from any write.

Register bit CSPR*n*[WP] per chip-select indicates whether or not write accesses are allowed on the particular chip-select.

If a NOR device is connected and a write transaction arrives to a write-protected chip-select, then do not assert chip-select on the NOR flash device and consume the transaction internally. Hence, the protection is controlled by software and the IFC hardware. A write protect error is also generated.

If a NAND device connected to a write-protected chip-select, then the write protect signal should be asserted before asserting chip-select as shown in in the figure below. As a NAND device is not directly memory-mapped like a NOR device and is accessed through the instruction register (FIR), hence there is no prior information of the type of operation to be performed on the NAND. Due to no prior information of the NAND access type

(read/write) before asserting the chip-select, it cannot be handled the same way as it is handled in the write protect for the NOR. Thus, assert chip-select even if the device is write-protected and expect the device to ignore the data sent.

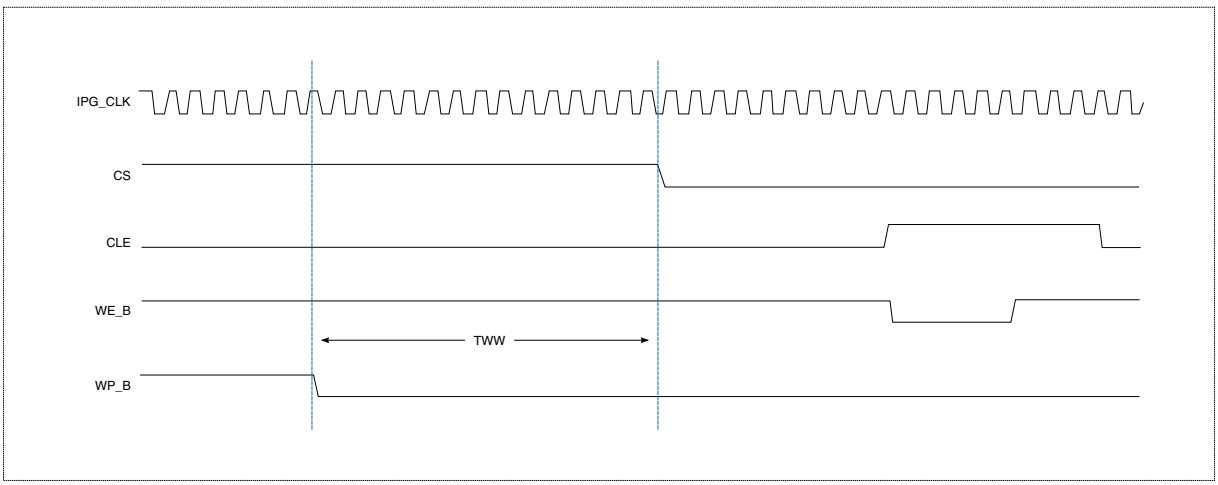


Figure 13-246. Write-protect timing in NAND

13.4.2.3 SRAM buffer

Read and write accesses to IFC banks controlled by NAND FCM do not access attached NAND flash EEPROMs directly. Rather, these accesses read and write the buffer RAM (a single, shared 16-KB space internal to the IFC and mapped by the base address of every NAND FCM bank).

Even though each NAND FCM-controlled bank has a different base address to differentiate it, all accesses to such banks access the same buffer space. External IFC signals, such as AVD/ALE and CS*n*, do not assert upon accesses to the buffer RAM. The buffer RAM is logically divided into two or more buffers, depending on the setting of CSOR*n*[PGS], with different buffers being accessible concurrently by software and NAND FCM.

- To perform a page-read operation from a NAND flash device, software initializes the FCM command, mode and address registers, and triggers the read operation on a particular bank by setting the bit in NANDSEQ_STRT register. FCM executes the sequence of op-codes held in FIR, reading data from the flash device into the shared buffer RAM. While this read is taking place, software should not access buffer RAM. After loading the complete page on buffer RAM ECC decoding can be performed if it is enabled. Once all the errors have been fixed and if operation completion

interrupt is enabled, an interrupt is generated. When FCM has completed its last command, buffer RAM can be read.

- To perform a page-write operation, the software first prepares data to be written in a fresh buffer. Then, the FCM command, mode, and address registers are initialized, and triggers the write operation on a particular BANK by setting the bit in NANDSEQ_STRT register. FCM will execute the sequence of op-codes held in FIR, writing data from shared buffer RAM to the flash device. While this write is taking place, the software is free to write data in other buffers of the FCM buffer RAM.

By programming NCFGR[NUM_LOOP] field, the NAND FCM can be used for multi-page read and write operation. A maximum of 16 pages (for 512-byte page), 4 pages (2-KB page), and 2-page (4-KB page) can be used for multi-page operation. Operation completion interrupt will be sent only after completing last page operation.

13.4.2.3.1 Guidelines for SRAM buffer for NAND access

Guidelines for SRAM buffer for NAND access are as follows:

- The IFC supports only one operation at a time on the NAND flash; that is, the user can either read data from the SRAM buffer for a NAND flash read or write data in the SRAM buffer for a NAND flash-write.
- NAND program: The user must fill the complete program data in the IFC's SRAM buffer before setting the trigger on the IFC to start the data transfer from the SRAM buffer to the NAND flash. When the trigger is set to transfer the data to a NAND device, the SRAM buffer must not be accessed for either read or write. Accessing the SRAM buffer during this time may result in undesirable outcome. The user must wait for the current program operation to complete before accessing the SRAM buffer for the next operation.
- NAND read: When the trigger is set in the IFC, the read data coming from the NAND device is stored in an SRAM buffer. In this operation, the SRAM buffer should not be accessed until the event-completion flag is set.

13.4.2.3.2 Buffer layout and page mapping for 512-byte page NAND flash

The FCM buffer space is divided into 16 1-KB buffers for 512-byte-page devices (CSOR*n*[PGS] = 00), mapped as shown in the figure below.

The EEPROM's page numbered P is associated with buffer number (P mod 16), where P = ROW*n*. Because the bank size set by AMASK*n*[AM] is greater than 16 KB, an identical image of the FCM buffer RAM appears replicated every 16 KB throughout the bank address space.

In the case where NAND_BC[BC] = 0, FCM transfers an entire page, comprising the 512-byte main region followed by the 16-byte spare region; the 496-byte reserved region is not accessed and remains undefined for software. However, for commands given a specific byte-count in NAND_BC[BC], COL*n*[MS] locates the starting address in either the main region (MS = 0) or the spare region (MS = 1).

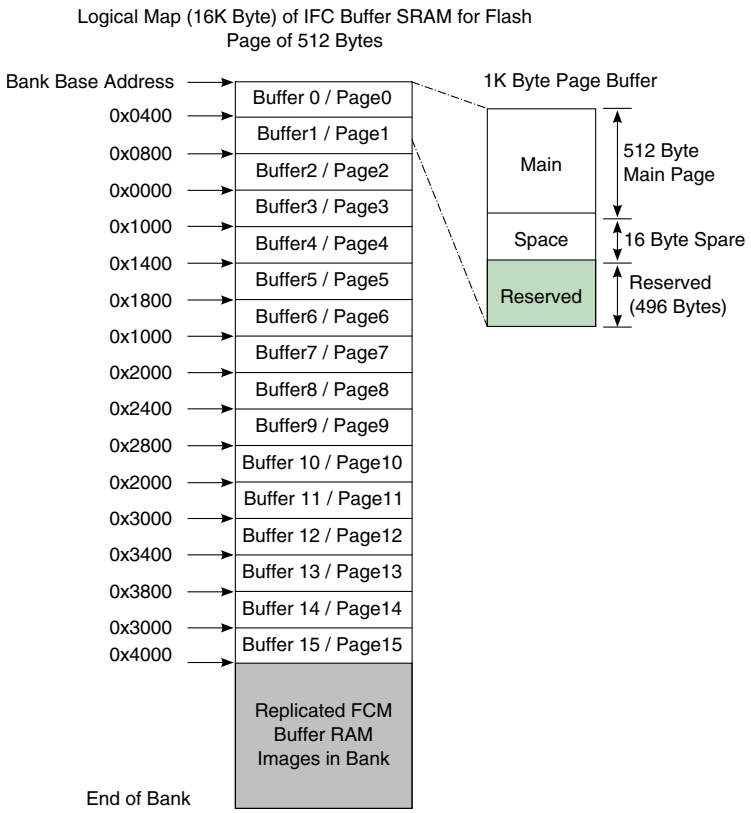


Figure 13-247. SRAM buffer layout for 512-byte page device

13.4.2.3.3 Buffer layout and page mapping for 2-KB page NAND flash

The FCM buffer space is divided into four 4 KB buffers for 2-KB page devices (CSOR*n*[PGS] = 01).

Each page in a 2 KB-page NAND flash comprises 2112 bytes, where 2048 bytes appear as main-region data and 64 bytes as spare-region data. The EEPROM's page numbered P is associated with buffer number (P mod 4), where P = ROW*n*. Because the bank size set by AMASK*n*[AM] is greater than 16 KB, an identical image of the FCM buffer RAM appears replicated every 16 KB throughout the bank address space.

If NAND_BC[BC] = 0, the FCM transfers an entire page comprising the 2048-byte main region followed by the 64-byte spare region; the 1984-byte reserved region is not accessed, and remains undefined for software. However, for commands given a specific byte count in NAND_BC[BC], COL*n*[MS] locates the starting address in either the main

region (MS = 0) or the spare region (MS = 1). When different IFC banks control both page devices, a 4 KB-page buffer must be assigned to either the first four or last four 512-byte page buffers.

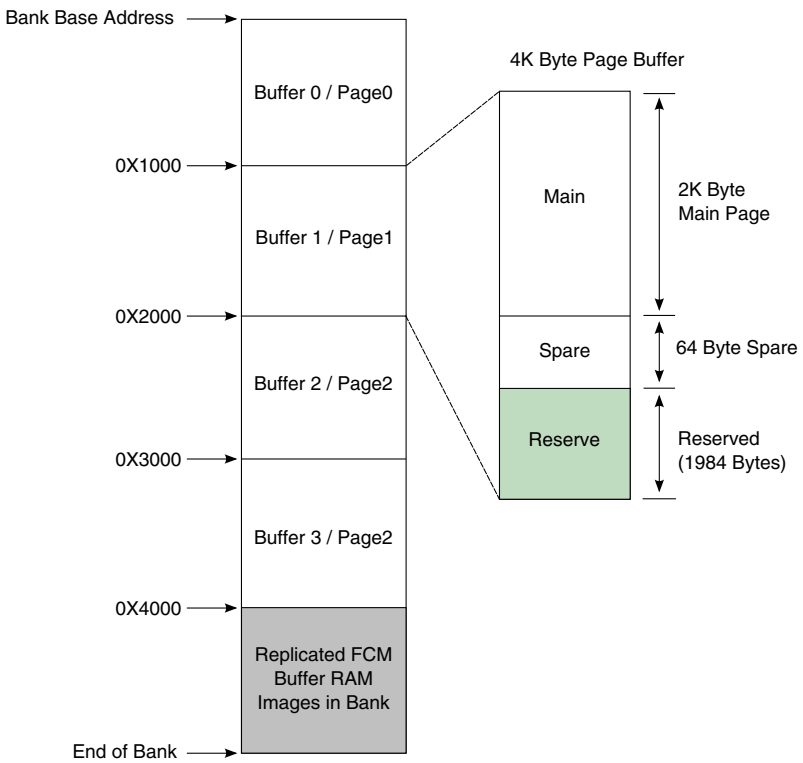


Figure 13-248. SRAM buffer layout for 2-KB page device

13.4.2.3.4 Buffer layout and page mapping for 4 KB page NAND flash

The FCM buffer space is divided into two 8 KB buffers for 4 KB-page devices (CSOR*n*[PGS] = 10).

Each page in a NAND flash comprises 4224 bytes, where 4096 bytes appear as main region data and 128 bytes as spare region data. The EEPROM's page numbered P is associated with buffer number (P mod 2), where P = ROW*n*. Because the bank size set by AMASK*n*[AM] is greater than 16 KB, an identical image of the FCM buffer RAM appears replicated every 16 KB throughout the bank address space.

If NAND_BC[BC] = 0, FCM transfers an entire page comprising the 4096-byte main region followed by the CSOR*n*[SPRZ]-byte spare region.

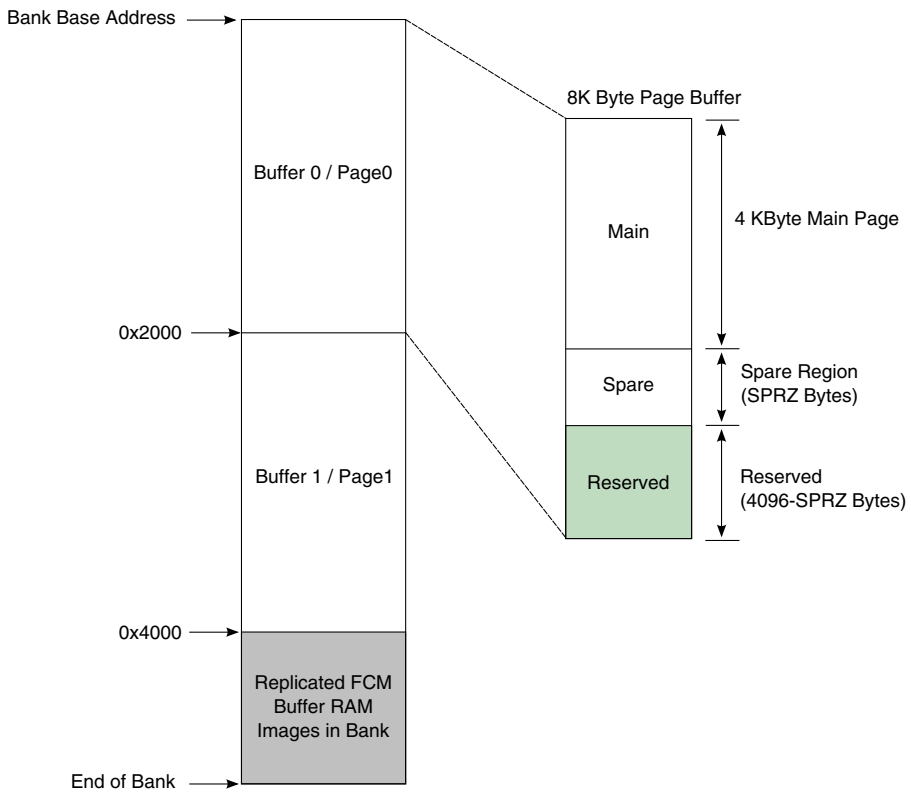


Figure 13-249. SRAM buffer layout for 4 KB-page device

13.4.2.3.5 Buffer layout and page mapping for 8 KB page NAND flash

The FCM buffer space is divided into single 8 KB buffers for 8 KB large-page devices (CSOR*n*[PGS] = 11).

Each page in a large-page NAND flash comprises 8192 + spare region bytes, where 8192 bytes appear as main region data, and spare region bytes appear as spare region data. Because the bank size set by AMASK*n*[AM] will be greater than 16 KB, an identical image of the FCM buffer RAM appears replicated every 16 KB throughout the bank address space.

If NAND_BC[BC] = 0, the FCM transfers an entire page comprising the 8192-byte main region followed by the CSORn[SPRZ]-byte spare region.

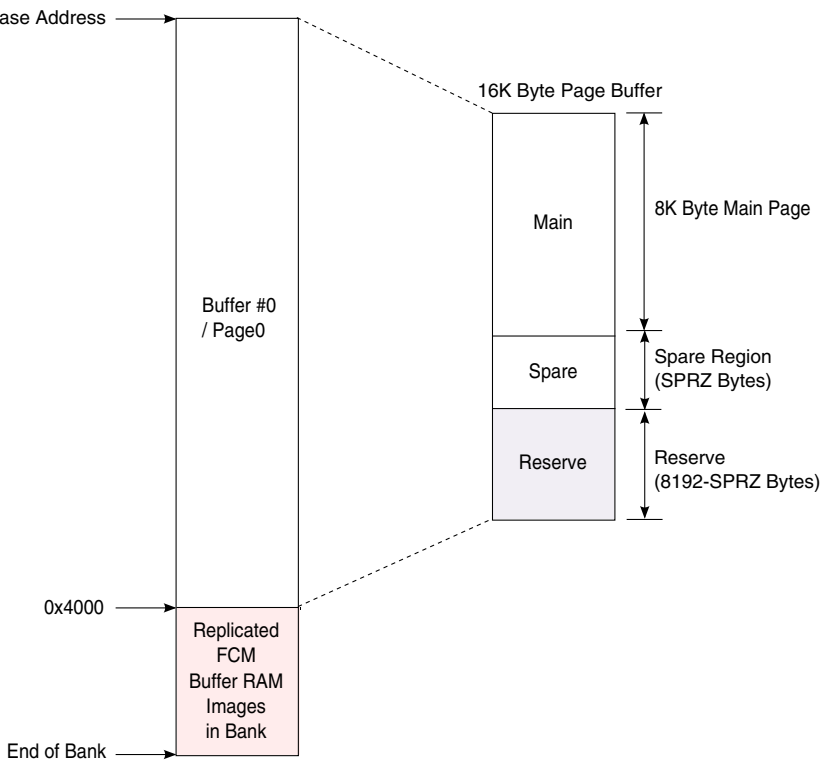


Figure 13-250. SRAM buffer layout for 8 KB-page device

13.4.2.3.6 SRAM buffer initialization requirement

Byte-select is not supported in the SRAM; therefore, read-modify-write is implemented for any system-side write into the SRAM buffer.

Reading uninitialized memory results in ECC error from the SRAM.

NOTE

The user must initialize/prefill the SRAM buffer by any data before writing anything in the SRAM from the chip. To initialize the SRAM, the user can read any page from the NAND flash. This is a one-time activity post-boot.

The software workaround for this ECC error is as follows:

Set NCFGR[SRAM_INIT_EN] bit to initialize the SRAM before initiating the first program operation. See NAND configuration register (IFC_NCFGR) for details.

13.4.2.4 Use of ECC algorithms

BCH encoder and decoder algorithms are implemented to detect and correct up to 4/8 bits in each 512-byte sector and 24/40 bits in each 1 KB sector.

A Galois Field 2^{13} is used for the encoding and decoding of 4- and 8-bit ECC and a Galois field 2^{14} is used for 24-/40-bit ECC.

13.4.2.4.1 Generating Galois field elements

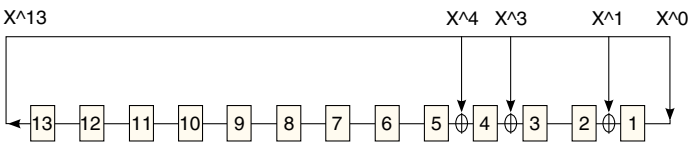
The Galois field elements can be generated by shifting the LFSR.

The following primitive polynomials are used for the computation of respective field elements.

Galois Field 2^{13} field elements

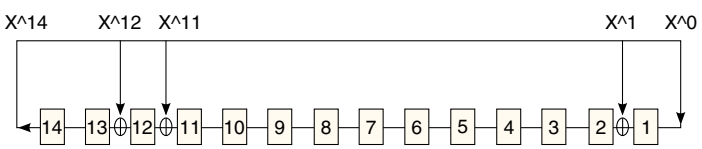
As shown in this figure, the field elements can be generated by shifting the LFSR. Each state of LFSR represents the field element.

P(x) = 11011000000001 in binary form and $1+x+x^3+x^4+x^{13}$ in polynomial form.



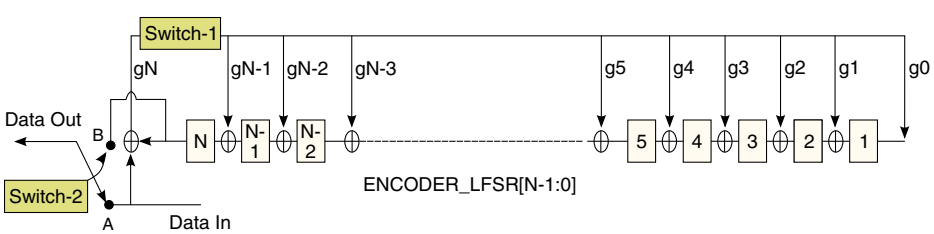
Galois Field 2^{14} field elements

Primitive polynomial P(x) = 110000000001101; that is, $1 + x + x^{11} + x^{12} + x^{14}$



13.4.2.4.2 BCH encoding

This figure represents the encoder implementation.



- 4-bit ECC: g(x) = 1101010101100001110101011100001000001100010010100 0101

- 8-bit ECC:
  g(x)=1100010011011111001000111010001110000010111000011100100000110000
  1101111000000111001010001001111110101
- 24-bit ECC:
  g(x)=1010000011110100000110100100000100010001000001011000111111011001
  1111001111110100001000110000001111000100100010001001000100010100101010
  1101111010100000111100101101110111110110111011000110001111101101111010
  0011000000111001000000111000010110101111011011001111101110011111011
  1011001101110001011100011010000000001101100000000011001000011010011111
  1
- 40-bit ECC:
  g(x)=1110000011101110011001100111111011010010111001110100001100000000000
  0001110011111000100111100000000010100011011101010001110110110101110101
  0100011110010111111100001101011010100100000110011000111010001111010101
  0011010000011001101101110111101010110100011111100001001110010000111101
  0100101100101111101001110111011011011000010101101110001001001110001010
  1111111011010011101101000000010101110101000101100111011010101011010000
  0000100001100110000110010100101111111000110111011001000111011000111111
  1111100111000000111110110100010110000011111010110011010010100110011101
  01111

In each of the generator polynomial binary forms mentioned above, the msb represents the lowest polynomial power (g0) and the lsb represents highest polynomial power (gN). The following LFSR can be used for BCH encoding.

In encoding, operation data corresponding to a sector is given to the LFSR. During this time,

- Switch 1 is closed and switch 2 is in the down position (A) to allow the transfer of data to the output.
- After the transfer of a sector's data, switch 1 is opened and switch 2 is moved to the up position (B).
- Now N parity bits can be read out from the LFSR. ENCODER_LFSR[N-1] represents the first parity bit out and ENCODER_LFSR[0] the last parity bit.
  - The value of N is 52, 104, 336 and 560 for 4-, 8-, 24- and 40-bit ECC.

When the IFC accesses the NAND flash device for a read operation, it performs a BCH detection algorithm and indicates how many bit errors were detected and corrected. The number of errors per sector gets registered in the ECCSTAT0/1/2/3 registers. For 24- and 40-bit ECC, only ECCSTAT0/1 are valid as only 8 sectors of 1 KB is possible for SRAM buffer.

ECC is kept in spare region of page at offset 08h. BCH ECC bytes are first written to the SRAM buffer and then to the NAND flash device. During program, the user can read the ECC bytes by reading the appropriate locations in the SRAM buffer.

During encoding, ECC is always calculated for 512-byte data for 4-/8-bit ECC and 1 KB data for 24-/40-bit ECC. CSORn[ECC_MODE] can be used to select 4-/8-/24-/40-bit correction mode, and CSORn[ECC_ENC_EN] and CSORn[ECC_DEC_EN] can be used to enable/disable the ECC logic.

If the encoder is enabled to calculate the ECC, then the data to the encoder is fed in the following manner:

- 16 bits are fed to the encoder every other clock cycle
- The encoder processes the bits in the following manner:
  - Bit #15 (d0), bit #14 (d1), bit #13 (d2), ... , bit #0 (d15), that is, bit #15 is the first bit to enter the encoder
  - Parity bit #0 (p0) is the first parity bit which comes out of the encoder

This table explains the manner in which the data and parity bits are stored in the 16-bit NAND flash memory for 4-bit encoding/decoding.

NOTE

- Even when storing data in an 8-bit NAND flash memory, the manner in which the data is fed to the encoder remains unchanged.
- The data to the decoder also needs to be fed in the same manner.

Table 13-245. 16-bit NAND flash memory organization with 4-bit/sector ECC

	bit #15	bit #14	bit #13	bit #12	bit #11	bit #10	bit #9	bit #8	bit #7	bit #6	bit #5	bit #4	bit #3	bit #2	bit #1	bit #0
Data Word 0	d0	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10	d11	d12	d13	d14	d15
Data Word 1	d16	d17	d18	d19	d20	d21	d22	d23	d24	d25	d26	d27	d28	d29	d30	d31
...																
Data Word 255	d4080	d4081	d4082	d4083	d4084	d4085	d4086	d4087	d4088	d4089	d4090	d4091	d4092	d4093	d4094	d4095
Spare Word 0	s0	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14	s15
Spare Word 1	s16	s17	s18	s19	s20	s21	s22	s23	s24	s25	s26	s27	s28	s29	s30	s31

Table continues on the next page...

Table 13-245. 16-bit NAND flash memory organization with 4-bit/sector ECC (continued)

.																
Spare Word 3	s48	s49	s50	s51	s52	s53	s54	s55	s56	s57	s58	s59	s60	s61	s62	s63
Spare Word 4	p0	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11	p12	p13	p14	p15
Spare Word 5	p16	p17	p18	p19	p20	p21	p22	p23	p24	p25	p26	p27	p28	p29	p30	p31
.																
Spare Word 7	p48	p49	p50	p51	0	0	0	0	0	0	0	0	0	0	0	0

Table 13-246. 16-bit NAND flash memory organization with 8-bit/sector ECC

	bit #15	**bit #14**	**bit #13**	**bit #12**	**bit #11**	**bit #10**	**bit #9**	**bit #8**	**bit #7**	**bit #6**	**bit #5**	**bit #4**	**bit #3**	**bit #2**	**bit #1**	**bit #0**
Data Word 0	d0	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10	d11	d12	d13	d14	d15
Data Word 1	d16	d17	d18	d19	d20	d21	d22	d23	d24	d25	d26	d27	d28	d29	d30	d31
...																
Data Word 255	d4080	d4081	d4082	d4083	d4084	d4085	d4086	d4087	d4088	d4089	d4090	d4091	d4092	d4093	d4094	d4095
Spare Word 0	s0	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14	s15
Spare Word 1	s16	s17	s18	s19	s20	s21	s22	s23	s24	s25	s26	s27	s28	s29	s30	s31
.																
Spare Word 3	s48	s49	s50	s51	s52	s53	s54	s55	s56	s57	s58	s59	s60	s61	s62	s63
Spare Word 4	p0	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11	p12	p13	p14	p15
Spare Word 5	p16	p17	p18	p19	p20	p21	p22	p23	p24	p25	p26	p27	p28	p29	p30	p31
...																
Spare Word 10	p96	p97	p98	p99	p100	p101	p102	p103	0	0	0	0	0	0	0	0
Spare Word 11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 13-247. 16-bit, large page (2K) NAND flash memory organization with 4-bit/sector ECC

	bit #15	bit #14	bit #13	bit #12	bit #11	bit #10	bit #9	bit #8	bit #7	bit #6	bit #5	bit #4	bit #3	bit #2	bit #1	bit #0
Data Word 0	d0	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10	d11	d12	d13	d14	d15
Data Word 1	d16	d17	d18	d19	d20	d21	d22	d23	d24	d25	d26	d27	d28	d29	d30	d31
...																
Data Word 1023	d16368	d16369	d16370	d16371	d16372	d16373	d16374	d16375	d16376	d16377	d16378	d16379	d16380	d16381	d16382	d16383
Spare Word 0	s0	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10	s11	s12	s13	s14	s15
Spare Word 1	s16	s17	s18	s19	s20	s21	s22	s23	s24	s25	s26	s27	s28	s29	s30	s31
.																
Spare Word 3	s48	s49	s50	s51	s52	s53	s54	s55	s56	s57	s58	s59	s60	s61	s62	s63
Spare Word 4	p0_0	p0_1	p0_2	p0_3	p0_4	p0_5	p0_6	p0_7	p0_8	p0_9	p0_10	p0_11	p0_12	p0_13	p0_14	p0_15
Spare Word 5	p0_16	p0_17	p0_18	p0_19	p0_20	p0_21	p0_22	p0_23	p0_24	p0_25	p0_26	p0_27	p0_28	p0_29	p0_30	p0_31
.																
Spare Word 7	p0_48	p0_49	p0_50	p0_51	0	0	0	0	0	0	0	0	0	0	0	0
Spare Word 8	p1_0	p1_1	p1_2	p1_3	p1_4	p1_5	p1_6	p1_7	p1_8	p1_9	p1_10	p1_11	p1_12	p1_13	p1_14	p1_15
Spare Word 9	p1_16	p1_17	p1_18	p1_19	p1_20	p1_21	p1_22	p1_23	p1_24	p1_25	p1_26	p1_27	p1_28	p1_29	p1_30	p1_31
.																
Spare Word 11	p1_48	p1_49	p1_50	p1_51	0	0	0	0	0	0	0	0	0	0	0	0
Spare Word 12	p2_0	p2_1	p2_2	p2_3	p2_4	p2_5	p2_6	p2_7	p2_8	p2_9	p2_10	p2_11	p2_12	p2_13	p2_14	p2_15
Spare Word 13	p2_16	p2_17	p2_18	p2_19	p2_20	p2_21	p2_22	p2_23	p2_24	p2_25	p2_26	p2_27	p2_28	p2_29	p2_30	p2_31
.																

Table continues on the next page...

Table 13-247. 16-bit, large page (2K) NAND flash memory organization with 4-bit/sector ECC (continued)

Spare Word 15	p2_48	p2_49	p2_50	p2_51	0	0	0	0	0	0	0	0	0	0	0	0
Spare Word 16	p3_0	p3_1	p3_2	p3_3	p3_4	p3_5	p3_6	p3_7	p3_8	p3_9	p3_10	p3_11	p3_12	p3_13	p3_14	p3_15
Spare Word 17	p3_16	p3_17	p3_18	p3_19	p3_20	p3_21	p3_22	p3_23	p3_24	p3_25	p3_26	p3_27	p3_28	p3_29	p3_30	p3_31
.																
Spare Word 19	p3_48	p3_49	p3_50	p3_51	0	0	0	0	0	0	0	0	0	0	0	0
Spare Word 20																
...																
Spare Word 31																

Note that pn_m, such that

- n = sector number (0, 1, 2, 3), where:
  - Sector 0: d0 to d4095
  - Sector 1: d4096 to d8191
  - Sector 2: d8192 to d12287
  - Sector 3: d12288 to d16383
- m = parity bit number, where:
  - 4-bit ECC: 0 to 51 (12 bits padded to 0 to align the number of parity bytes to the 8 bytes boundary)
  - 8-bit ECC: 0 to 104 (24 bits padded to 0 to align the number of parity bytes to the 8 bytes boundary)

During decoding, the full page is loaded into buffer RAM from flash and then decoding begins. This is required because the BCH decoder can work when a sector's data and corresponding ECC bytes are supplied to the decoder.

For 4-bit correction, 8 parity bytes, 8-bit correction 16 parity bytes, 24-bit correction 42 parity bytes, and 40-bit correction 70 parity bytes per sector are required. These parity bytes are stored in the spare region of the page at offset 08h. For small pages, only 4-bit

mode is allowed. For a 2 KB page, four sectors of 512 bytes each can be present in the main region; hence a total of 4x8= 32 parity bytes are store at offset 08h. For a 4 KB page size, eight sectors of 512 bytes each can be present; hence 8x8 =64 parity bytes are required for 4-bit mode and 128 bytes for 8-bit mode.

In 24- and 40-bit mode, the parity information is stored adjacent to each other in the spare region; that is, the ECC bytes will be placed one after another. With 4-/8-bit ECC mode, there is no need to pad 0 bits in order to align it to the 8-byte boundary, unlike the padding that is performed with the 4-/8-bit ECC mode.

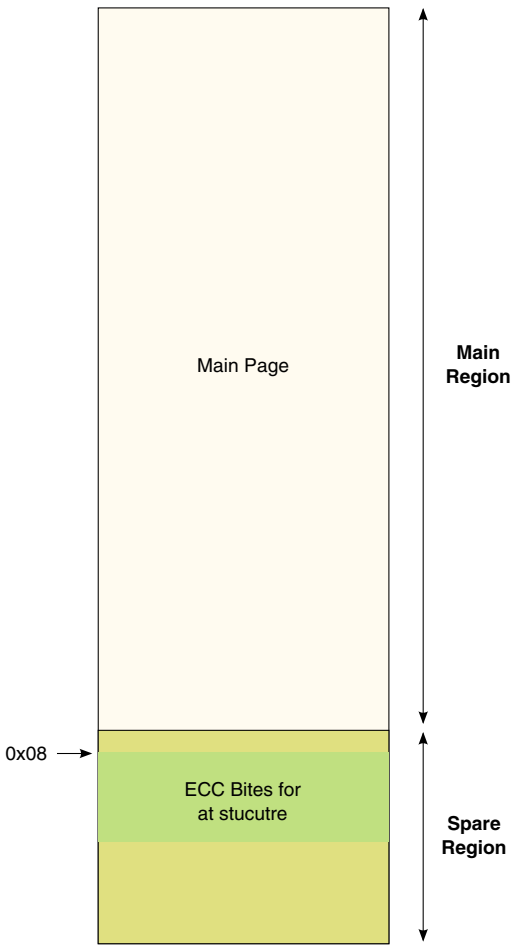


Figure 13-251. ECC arrangement on a page

For example, there is a memory of 4 KB page size and a 24-bit ECC is needed to protect it. A 4 KB page size memory will have four sectors of 1 KB each. The encoder will generate 42 bytes of parity information for each sector. Therefore, there will be a total of 42x4 = 168 ECC bytes. All 168 bytes are placed in the spare region of the NAND flash at offset 08h: { Sector-1 {P0, P1,...P335}, Sector-2 {P0,P1,...P335}, Sector-3 {P0, P1,...P335}, Sector-4 {P0,P1,...P335} } from offset 08h where P0 is the first parity bit coming out from encoder and P335 the last bit. The data and parity feeding and arrangement for all the modes remains same and as shown in Table 13-247.

This figure represents the logical flow of encoding and decoding operation during program and read. During program, ECC bytes are also written back in SRAM. During read, first, all data bytes corresponding to a page are written in SRAM buffer, and then the decoding starts on sector basis. At any time, only one operation can be performed on the NAND; that is, it can only be either read or programmed.

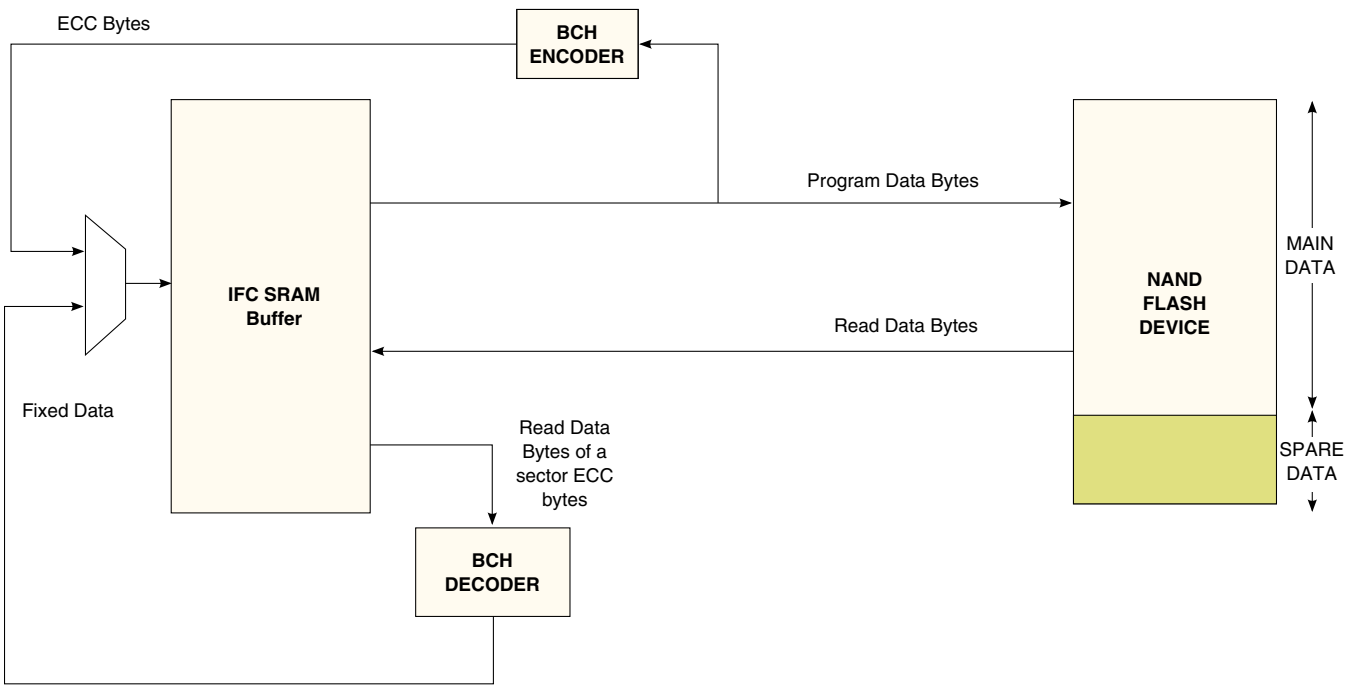


Figure 13-252. BCH encoding/decoding during flash program/read

NOTE

The BCH encoder and decoder should be enabled only when performing full-page operations. If there are partial page operations, the encoder and decoder should be disabled.

NOTE

The IFC computes the ECC only for the main region data (as the ECC is not computed for the spare region). The decoder detects and correct 4/8 bits of error on the main region data and the corresponding ECC bytes stored.

13.4.2.5 Programming the NAND FCM

The NAND FCM performs operations on the NAND flash interface based on the values programmed in the common and NAND register spaces.

The user is expected to program the following registers with the appropriate values before triggering the NAND FCM operation:

- Chip-select option registers
- Flash timing registers
- General control register
- NAND configuration register
- NAND flash command registers
- Flash row and column address registers
- NAND flash byte count register
- NAND flash instruction registers
- NAND chip-select register
- NAND event and error interrupt enable register (in case if an interrupt needs to be generated)
- NAND control register

When all the above registers have been programmed, the NAND FCM can then be triggered through the NAND operation sequence start register.

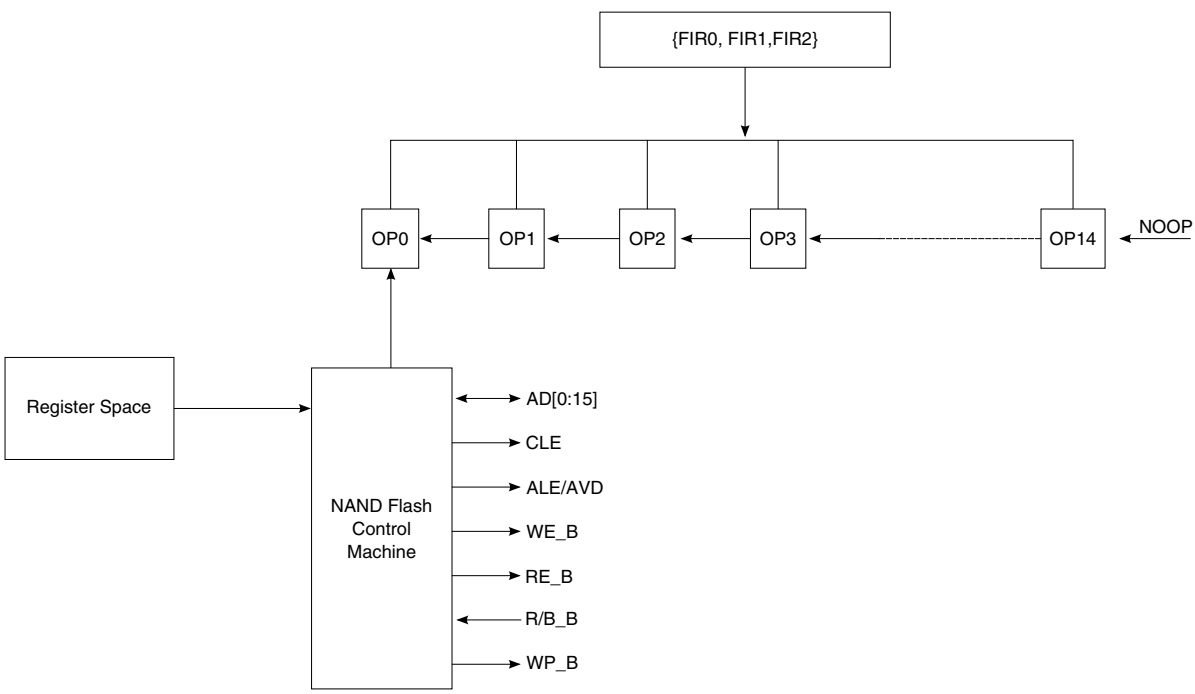


Figure 13-253. FCM instruction sequence mechanism

NOTE

User must use the WFR opcode at the end of NAND FIR programming to probe the RDY_B/BSY_B signal before end of operation. Status poll is also recommended after every write or erase command.

13.4.2.5.1 FCM command instructions

There are different types of command instructions.

- Commands that issue immediately (CMD0-CMD7)
  - Asynchronous mode: These commands write a single command byte by asserting CLE and WE_B while driving an 8-bit command on the AD[0:7]/ AD[8:15] (8-/16-bit port) bus. The opcode CMD*n* sources its command byte from the field FCR[CMD*n*]. Therefore, up to eight different commands can be issued in any instruction sequence.
- Commands that wait for RB_B to be sampled high before issuing (CW0-CW7)
  - Asynchronous mode: These commands first poll the RB_B signal to be sampled high before writing a single command write on the AD[0:7]/AD[8:15] (8-/16-bit port) bus sourced from FCR[CMDn] for opcode CW*n*. It is necessary to use CW*n* opcodes whenever the memory is expected to be in the busy state (such as following a page read, block erase or program operation) and therefore, initially unresponsive to commands.

Consult the manufacturer's datasheet to determine the values to be programmed into the FCR register and whether a given command in the sequence is expected to initiate device busy behavior.

13.4.2.5.2 FCM address instructions

Address instructions are used to issue addresses to the NAND flash device.

- Asynchronous mode: A complete address is formed by a sequence of 1 or more bytes, each written onto AD[0:7]/AD[8:15] (8-bit/16-bit port) with AVD/ALE and WE_B asserted together.

13.4.2.5.3 FCM data read instructions

Read data instructions assert RE_B repeatedly to transfer one or more bytes of read data from the NAND flash device.

The different types of data read instructions provided are as follows:

- Read BC bytes of data from the NAND flash device into the internal SRAM
  - Asynchronous mode: This instruction reads FBCR[BC] into the current FCM SRAM buffer addressed. The data driven by the NAND flash device is sampled on the basis of the TRAD timing parameter and the RE_B is asserted for a time period equivalent to the TRP timing parameter. If FBCR[BC] = 0, an entire page (including the main and the spare regions) is transferred in a burst.

- Read BC bytes of data from the NAND flash device into the internal SRAM during boot
  - Asynchronous mode: This instruction reads FBCR[BC] into the current FCM SRAM buffer addressed. The data driven by the NAND flash device is sampled on the basis of the TRAD timing parameter and the RE_B is deasserted once the read data has been sampled by the NAND FCM. If FBCR[BC] = 0, an entire page (including the main and the spare regions) is transferred in a burst. This instruction can also be used for non-boot applications wherein the RE_B signal needs to be asserted until the read data has been sampled.
- Read the NAND flash device status
  - Asynchronous mode: This instruction performs a status read operation on the NAND flash device. The status returned by the device is stored in the NAND_FSR register.
- Read data into the MDR register
  - Asynchronous mode: This instruction reads 8/16 bits of data from a given page in the NAND flash device (based upon the column address programmed) into the MDR register.

13.4.2.5.4 FCM data write instructions

Write data instructions assert WE repeatedly to transfer program data to the NAND flash device.

- Asynchronous mode: Write data instructions assert WE repeatedly to transfer 1 or more bytes of program data to the NAND flash device. If FBCR[BC] = 0, an entire page (including the main and the spare regions) is transferred in a burst.

13.4.2.6 Looping FIR sequences

The loop feature provides the flexibility to loop certain FIR sequences (such as program, read, and erase) a finite number of times based on the value programmed in NCFGR[NUM_LOOP].

The row and column addresses sent to the NAND flash device during a loop operation depend on NCFGR[ADDR_MODE]. For a more detailed description of the addresses performed on the NAND flash device for every subsequent execution of the FIR sequence during a loop operation, see NAND configuration register (IFC_NCFGR).

NOTE

For this feature to correctly execute, it is important to note that the opcodes programmed in the FIR sequence corresponding to the column and row addresses should always be CA0 and RA0.

Therefore, it is recommended that only one type of operation (such program/read/erase) be performed through a loop sequence.

13.4.2.7 NAND asynchronous mode timings

This section describes the basic NAND flash timing waveforms and the programmable timing parameters.

Figure 13-254 explains the NAND program cycle; Figure 13-255 and Figure 13-256 explain the basic read cycle waveforms and corresponding timing parameters for NON-EDO and EDO mode flash. Non-EDO mode corresponds to ONFi2.0 Async mode 0/1/2/3 while EDO mode corresponds to ONFi mode 4/5.

13.4.2.7.1 NAND asynchronous mode program data timing

This figure shows the NAND flash program.

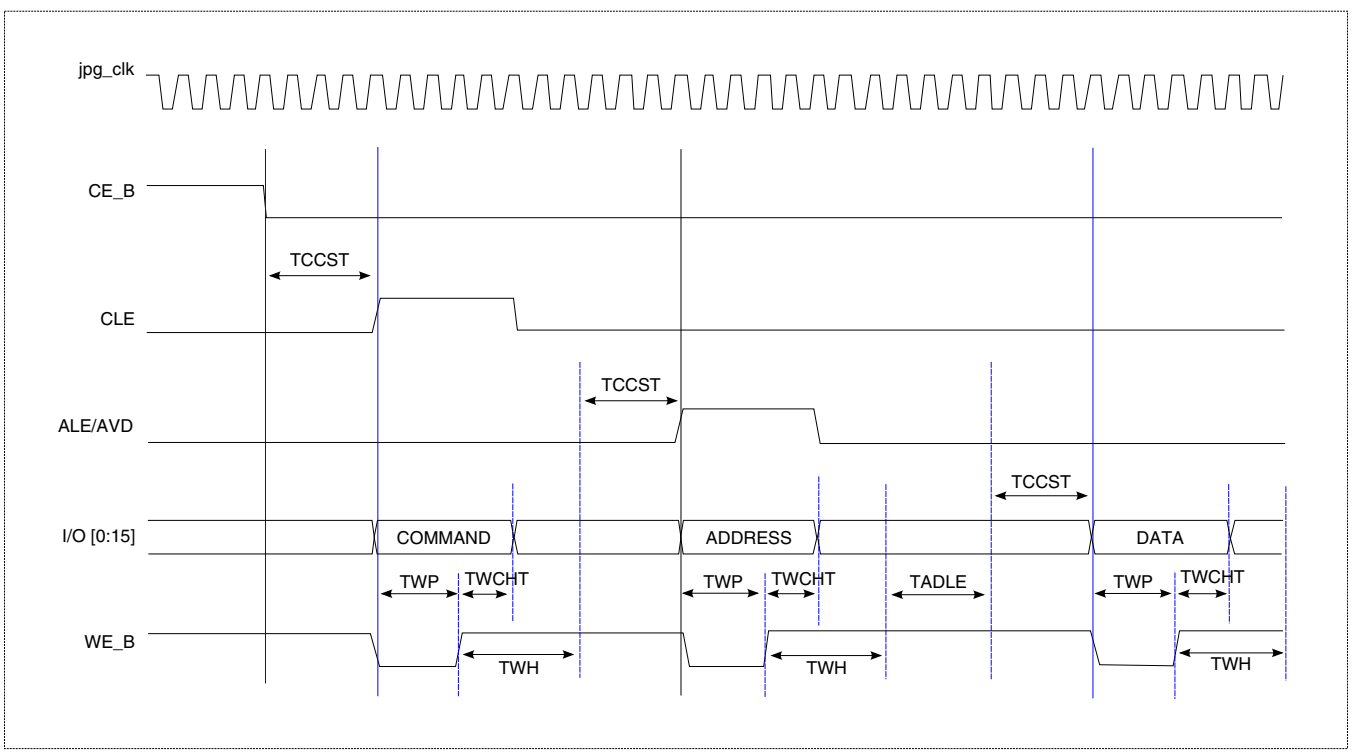


Figure 13-254. NAND flash program

13.4.2.7.2 NAND asynchronous mode read data timing

This figure shows the NAND flash read.

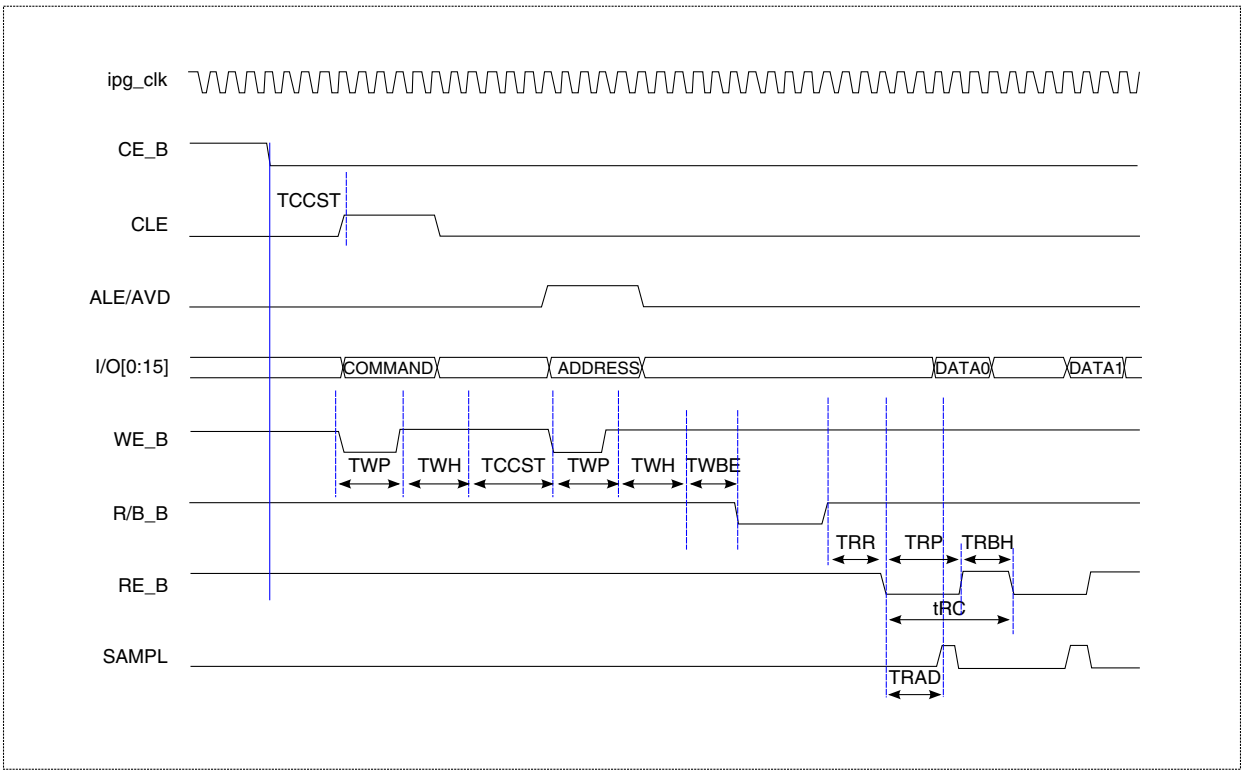


Figure 13-255. NAND flash read

13.4.2.7.3 NAND asynchronous mode calculating read data window width

Calculating the read data window width can be done in several ways.

- Non-EDO mode NAND
  - The window size for the received data is calculated as t_{RP} - t_{REA} + t_{RHOH}.
- EDO mode NAND
  - The window size for the received data (except the last data beat) is calculated as t_{REH} - (t_{REA} - t_{RP}) + t_{RLOH}.
  - The window size for last data beat received is calculated as t_{RHOH} - (t_{REA} - t_{RP}).

Based on these calculations, the minimum size of the data window width is 9 ns (for EDO ONFi mode 5). To support this data window width, a minimum of 333 MHz for the IFC module input clock is required. By increasing the value of timing parameter t_{RP}, that is, the RE pulse width, the data window size can be increased.

NOTE

The read cycle time should satisfy this condition: ((t_{RP} + t_{REH})≥ t_{RCmin}). If the sum of the t_{RPmin} and the t_{REHmin} values specified by the datasheet is less than the t_{RCmin} (from the datasheet), it is then recommended that the t_{RPmin} value be increased to meet the t_{RCmin} timing requirement. This is recommended because

increasing the t_{RPmin} value causes the data window size to increase.

Read data sampling time (TRAD) represents the read data sampling time on the NAND. The value should be programmed where the sampling at the IFC should be done at the center of the received data eye. Its value can be calculated as [T_{REAmax} (from NAND datasheet) + 2 x board delay + 1/2 received data window size]. If in EDO mode, the user should choose the TRAD value corresponding to the smaller data window calculated using the two formulae explained above for EDO.

NOTE

For EDO mode, the value of TRAD should always be less than t_{RP} + t_{REH}. If the board delay value used in the TRAD calculation results in this condition to not be met, the user may not be able to run the interface at the EDO mode frequency. In this situation, the read timing should be relaxed by increasing t_{RP} value to access the NAND device at a slower speed.

13.4.2.7.4 NAND asynchronous mode read data sampling approach

The IFC follows the approach in which the timing parameters are programmable.

The IFC logic runs at the IFC module input clock and can generate these timings on the flash interface through the programmed value in the timing registers. Based on the timing values as given in the ONFi 2.0 spec for each mode, received data window width is calculated (as explained in the previous section).

13.4.2.7.5 NAND asynchronous mode read status timing

This figure shows NAND read status timing.

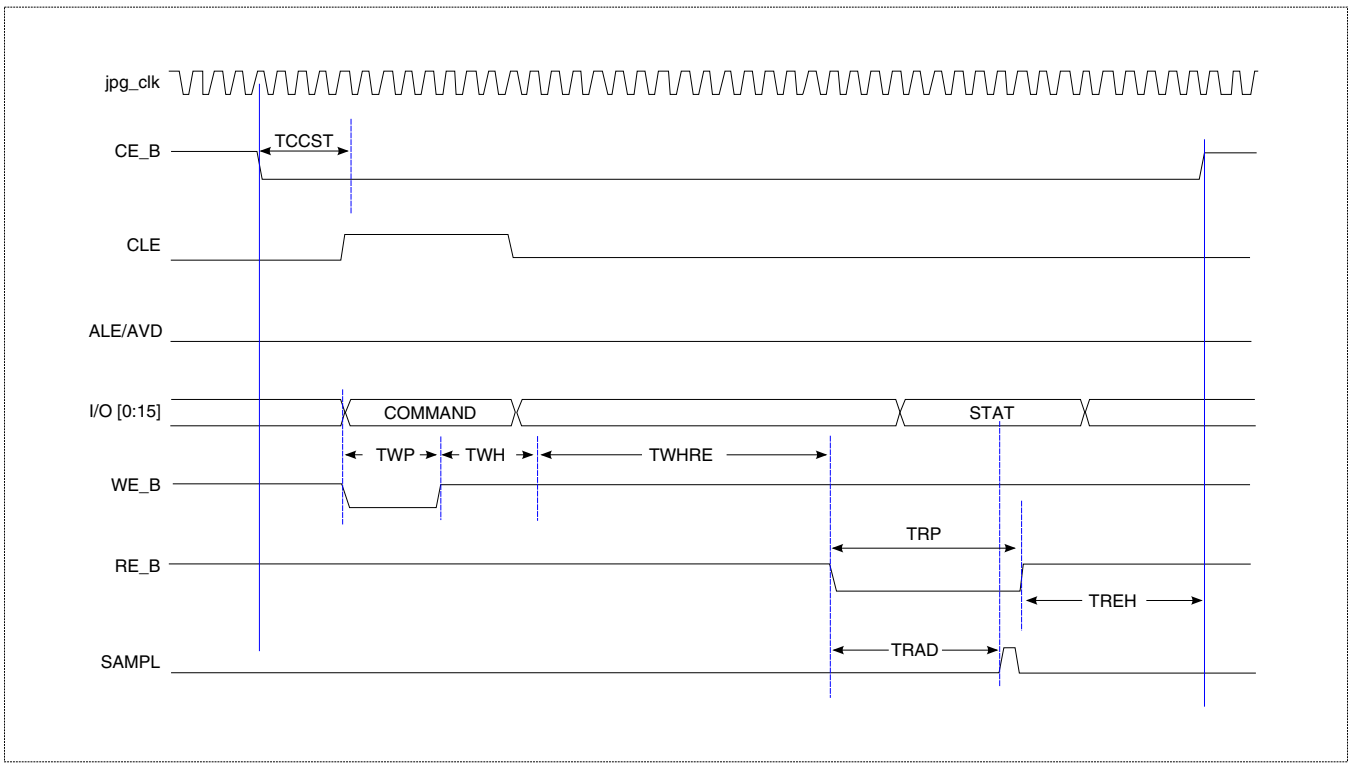


Figure 13-256. NAND status read

13.4.2.8 NAND asynchronous mode boot mechanism

If the FCM is selected as the boot ROM controller from power-on-reset configuration, the IFC automatically loads up to 8 KB of boot code from BANK0 to the NAND FCM buffer RAM depending on the RCW load or BOOT load indication.

The CPU can execute boot code directly from the FCM buffer RAM, but must ensure that any further data read from the NAND flash EEPROM is transferred under software control in order to continue the bootstrap process.

Because AMASK0[AM] is initially cleared during reset, all CPU fetches to the IFC access the FCM buffer RAM. First, 8 KB of SRAM holds the valid boot data, which appears in the memory map as a 8 KB RAM. No NAND flash spare regions are mapped during boot, therefore only 8 KB of contiguous, main-region data loaded from the first pages of the boot block (good block), are accessible in IFC bank 0, as indicated in this figure. In boot mode, addresses coming to SRAM buffer from system bus wraps at 8 KB physical address boundary.

The software must clear NCFGR[BOOT] to enable the logical to physical address mapping.

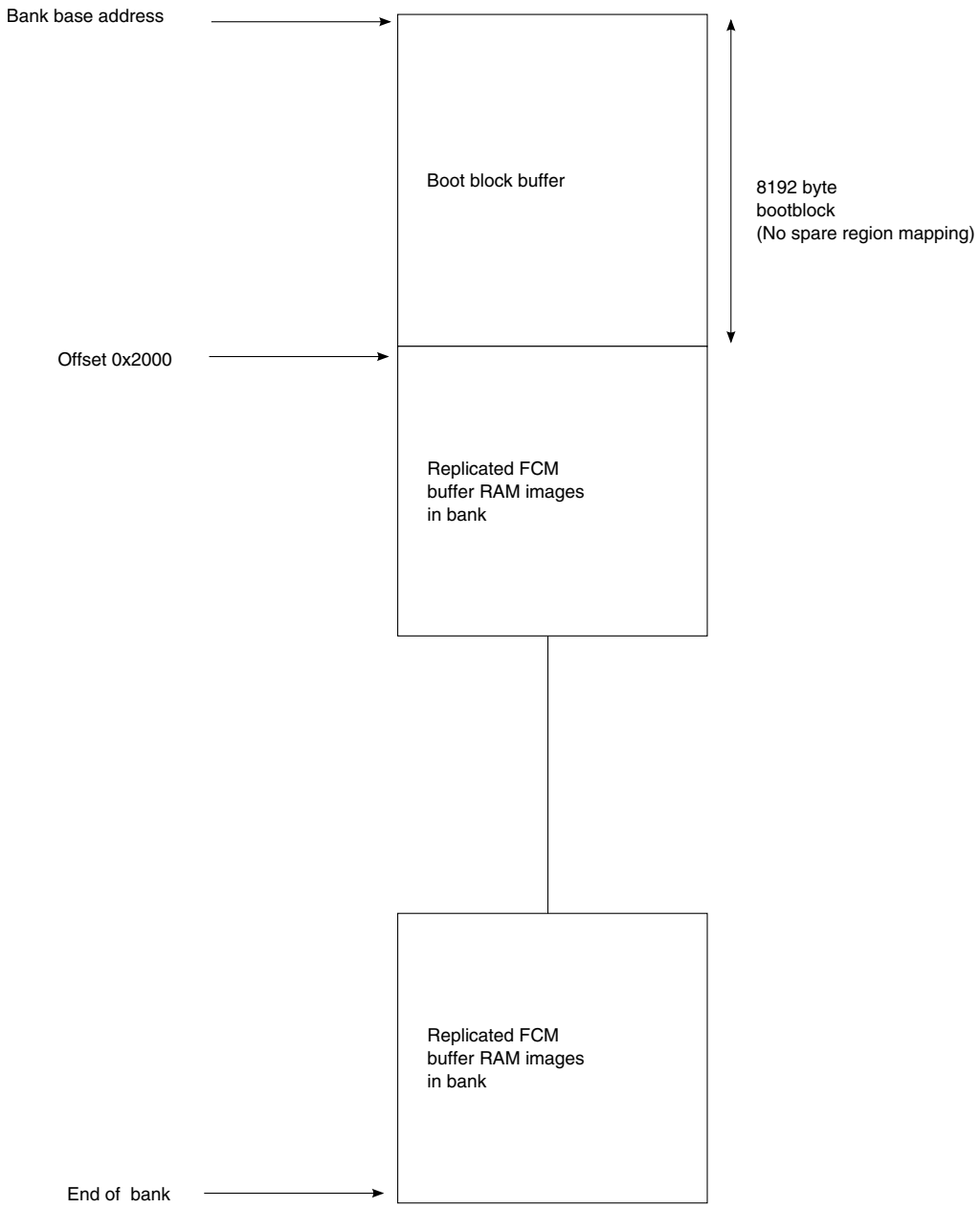


Figure 13-257. FCM buffer RAM map during boot

The IFC employs the counter-based read data sampling approach as described earlier. Here the read data sampling is determined by the counter value whose initial value is determined by the *NAND Flash Specification* and the IFC module input clock , but there can be two different clocks for boot. That is, if the system PLL is not locked, the RCW clock will be coming to the IFC and the counter values for read data sampling will be

governed by this clock frequency; otherwisethe IFC would receive the IFC module input clock post-system PLL lock for boot, which results in a different value of the timing counters.

These two clock frequencies are passed as parameters into the IFC during instantiation. Based on this, the IFC initializes timing parameters corresponding to the slowest mode of asynchronous NAND device (ONFi Mode-0) during boot. During normal operation, timing parameters can be programmed as per the device specification. Read data sampling is governed by read access time of NAND flash and board delay. This value is represented by timing parameter FTIM2_CS0[TRAD]. It can be initialized by the device by passing the TRAD parameter during instantiation.

During boot, the read data sampling mechanism is made slightly different than normal read to assure guarantee booting. Also, the read-enable signal remains asserted (low) until the data gets sampled, while in normal read operation the read-enable can be de-asserted. This mechanism ensures that there is not any case where the read data cannot be sampled by the IFC as data retains its state until the time read enable is asserted.

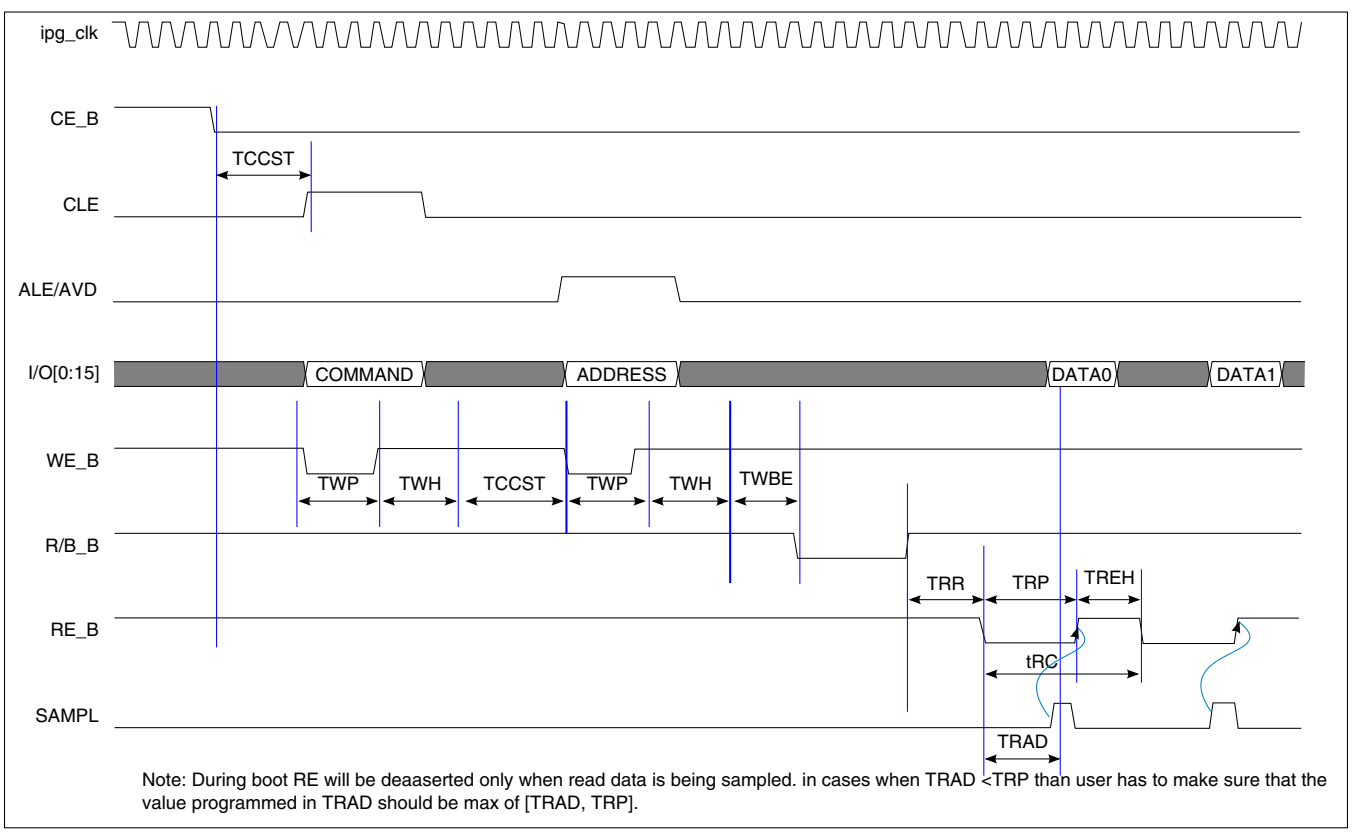


Figure 13-258. Read data sampling during boot

13.4.2.8.1 NAND asynchronous mode boot process

The steps below describe the boot process:

1. After receiving rcw_load or boot_load input signals, the IFC commences automatic boot block loading if the FCM is selected as the boot ROM location. Small-page (512-byte page) or large-page (2/4/8-Kbyte page), 8-bit or 16-bit NAND flash devices can be used for boot loading when enabled with CS0. With rcw_load or boot_load indication, the CSPR0[WP] bit also is set. With this bit set, the IFC drives WP_B low on the IFC output during boot accesses to prevent accidental erasure of the NAND flash boot ROM. Software needs to clear the CSPR0[WP] bit to commence normal write operation on CS0 after finishing the booting process.
2. The NAND FCM issues a reset command to the NAND device and then begins searching for a valid boot block from block 0.
3. The NAND FCM reads the spare regions of the first two pages or the first and the last page (depending upon pof_cfg_bbi_srch_sel) of the current block, checking the bad block indication (BI) bytes to validate the block for reading. BI bytes must hold the value 0xFF for the page to be considered readable.
   - For small-page (512-byte page) devices, the BI is a single-byte read from spare region byte offset 5 (8-bit port size). For a 16-bit port size device, the BI is a single byte read from spare region byte offset 11. The IFC supports booting from a 16-bit port size device only when no ECC is stored in the spare region , that is, the boot block is guaranteed to be good.
   - For large-page (2/4/8-Kbyte page) devices, the BI is a single-byte read from spare region byte offset 0 (8-bit port size), or two-bytes read from spare region byte offsets 0 and 1 (16-bit port size).

   If either of the above-mentioned pages of the current block are marked invalid, then the boot block index is incremented by 1, and the FCM repeats step 3. The IFC will continue searching for a bootable block indefinitely; therefore, at least one block must be marked valid for boot loading to proceed. At the conclusion of the boot block search, the value of ROW0 points to the boot block.
4. The FCM optionally checks the ECC at boot time depending on configuration selected during reset (por_cfg).
5. The FCM reads entire pages from the boot block until 8 Kbytes have been saved to the FCM buffer RAM when booting via the IFC NAND FCM. ECC errors are corrected if possible. If the FCM is unable to correct the ECC errors, the IFC signals an unrecoverable error by asserting the boot_err signal.
6. The CPU now commences fetching instructions, in random order, from the FCM buffer RAM. Boot software must clear NCFGR1[BOOT] to enable normal operation of FCM.

NOTE

During boot, the IFC performs a total of 5/6 address cycles, based on whether the NAND flash device used for booting is a small-page (512-byte page) or a large-page (2/4 /8-Kbyte page)

device. It is expected that the NAND flash device used to boot will ignore any excess address cycles which it may not need.

The following table describes the hard-coded FIR sequences used during auto-boot operation from the NAND flash.

Table 13-248. FIR sequences used during boot

FIR sequence	Description
{CMD3, NOOP}, where CMD3 = 0xFF	Issue soft reset to NAND device
{CW1, CA0, RA0, BTRD, NOOP}, where CW1 = 0x50	Partial page read from small-page device for bad block indicator read (good block search)
{CW0, CA0, RA0), BTRD, NOOP}, where CW0 = 0x00	Full page read from small-page device fom boot page read
{CW0, CA0, RA0, CMD2, BTRD, NOOP}, where CW0 = 0x00, CMD2=0x30	Partial page read from large-page device for bad block indicator read (good block search)
{CW0, CA0, RA0, CMD2, BTRD, NOOP}, where CW0 = 0x00, CMD2= 0x30	Full page read from large-page device fom boot page read

NOTE

Since fixed values of commands are used for read during auto-boot operations, only NAND flash devices supporting the commands mentioned in the above table can be used for auto-boot read operations.

13.4.3 NOR flash control machine

The NOR FCM allows an interface to asynchronous, simple, and internal latch-based NOR flash devices.

13.4.3.1 NOR boot

For NOR boot, CS0 works as the boot chip-select.

Therefore, during NOR boot, the external device should be connected on CS0. Before fetching the first instruction from an external device (NOR flash), the IFC flash timing registers are loaded with default timing parameters.

13.4.3.2 Simple asynchronous NOR read timings

The figures show the read cycle and burst read cycle timing diagrams.

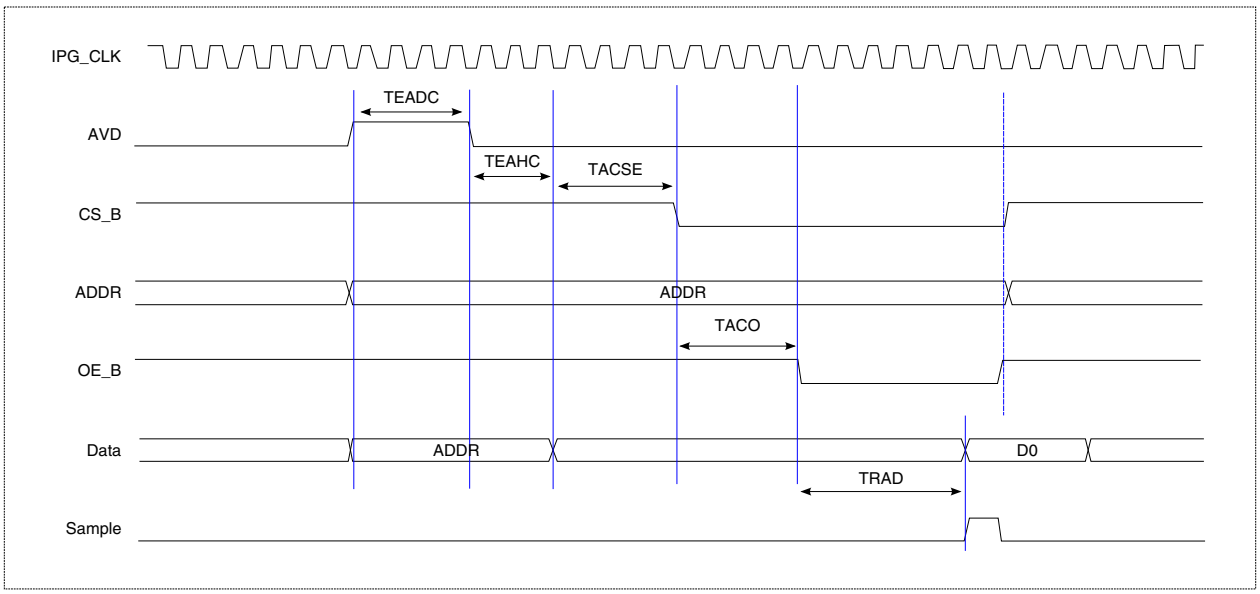


Figure 13-259. Read cycle timing

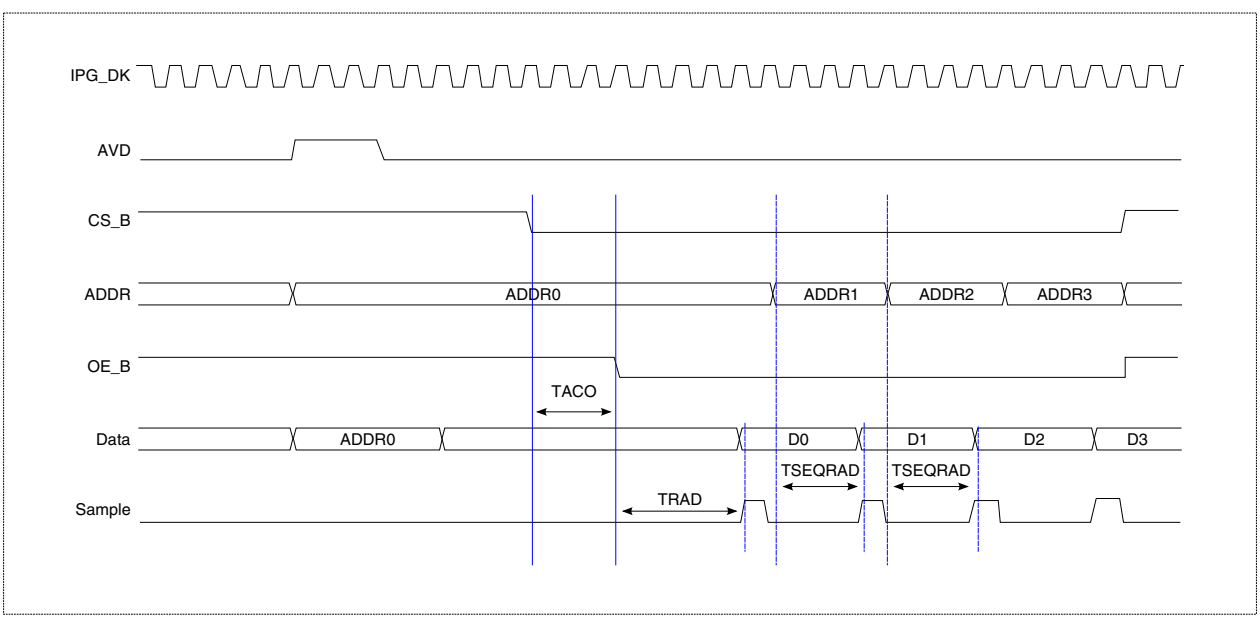


Figure 13-260. Burst read cycle timing

13.4.3.3 Simple asynchronous NOR write timings

The figures show the write cycle and burst write cycle timing diagrams.

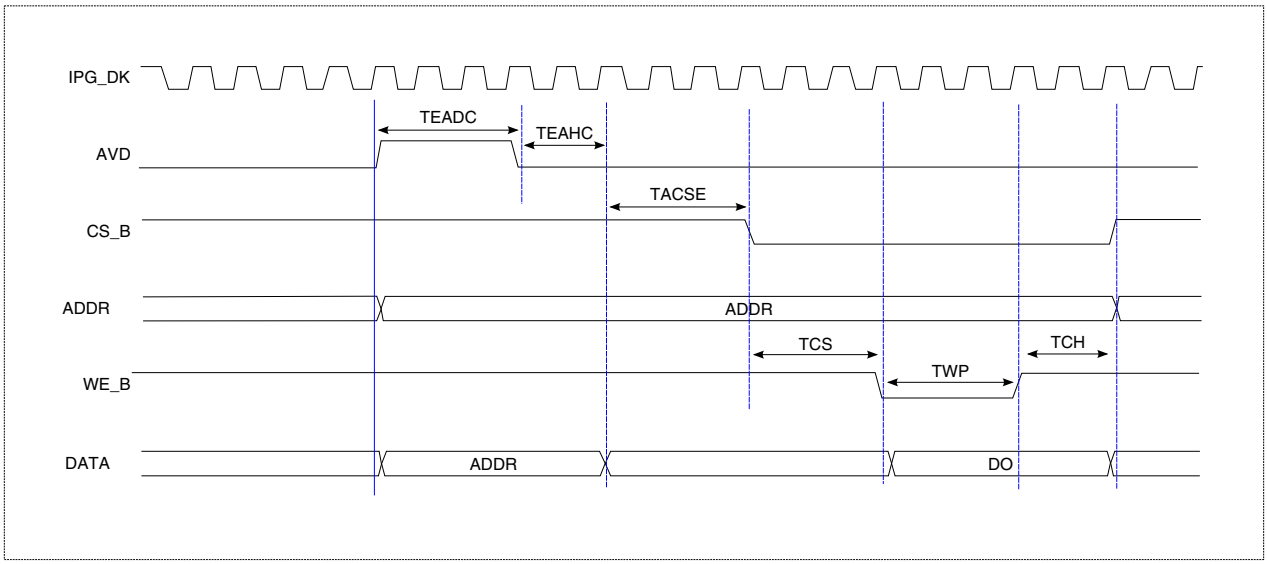


Figure 13-261. Write cycle timing

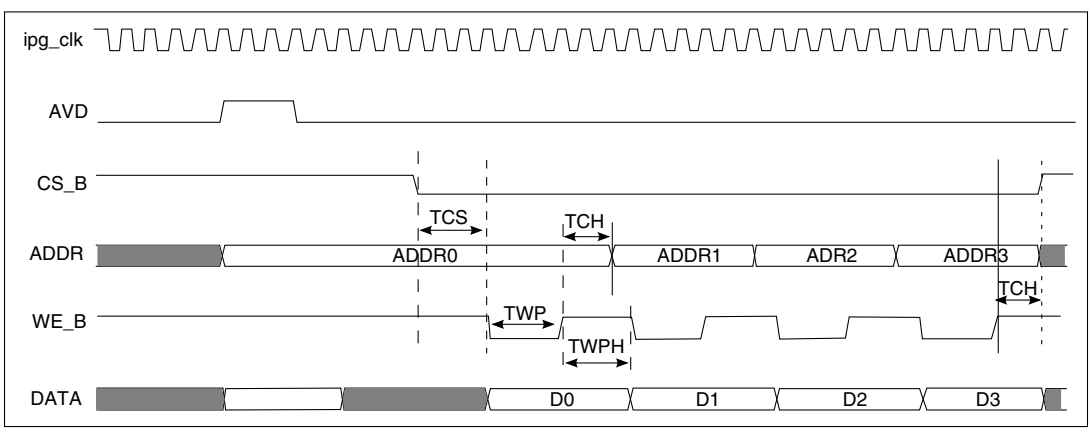


Figure 13-262. Burst write cycle timing

13.4.3.4 Internal latch-based asynchronous NOR read timings

This figure shows the read cycle timing diagram.

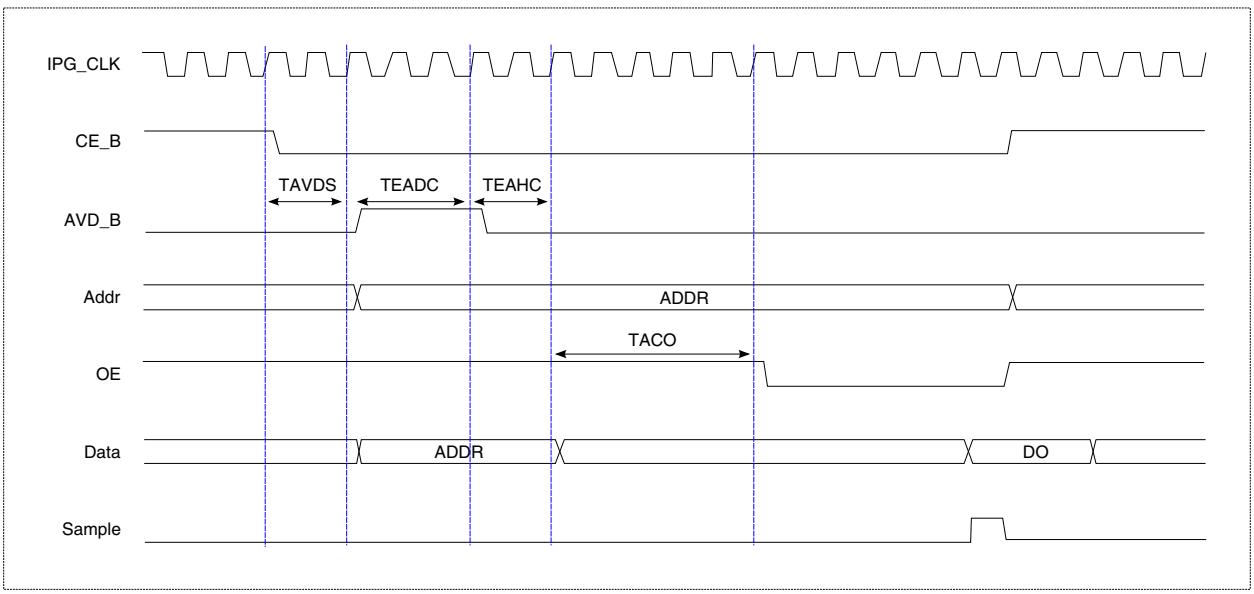


Figure 13-263. Read cycle timing

13.4.3.5 Internal latch-based asynchronous NOR write timings

This figure shows the write cycle timing diagram.

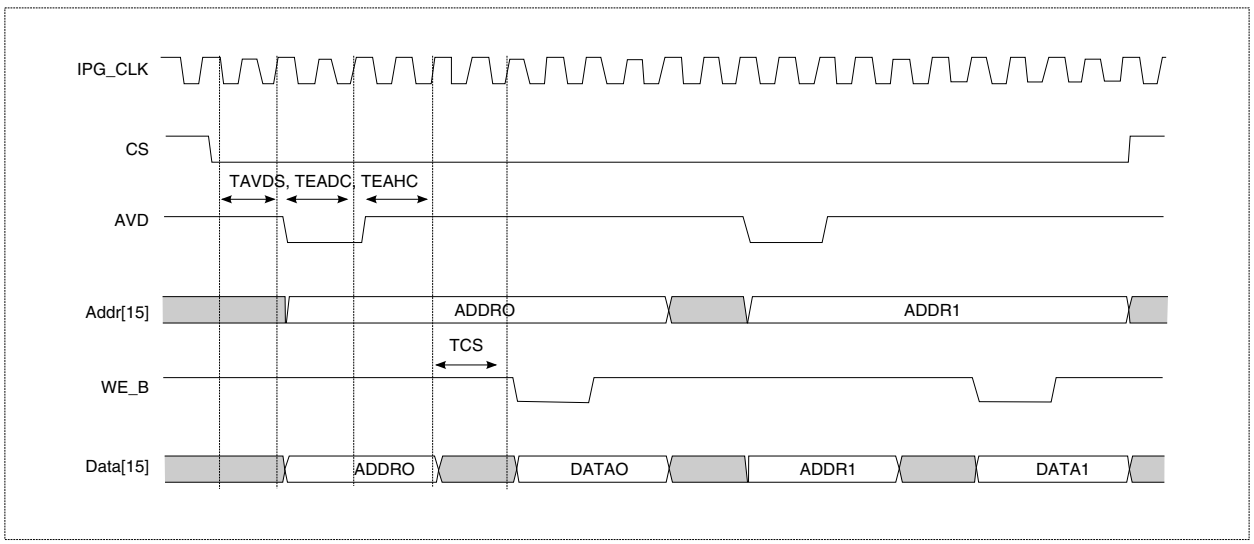


Figure 13-264. Write cycle timing

13.4.4 General purpose chip-select machine (GPCM)

The GPCM controller has two modes of operation: normal GPCM and generic ASIC.

Only one mode is operational at a time. The mode is selected by the value programmed in GPMODE field of CSOR*n*_GPCM register.

13.4.4.1 Normal GPCM mode of operation

The figure below explains the interface of the IFC to a device connected through the normal GPCM FCM.

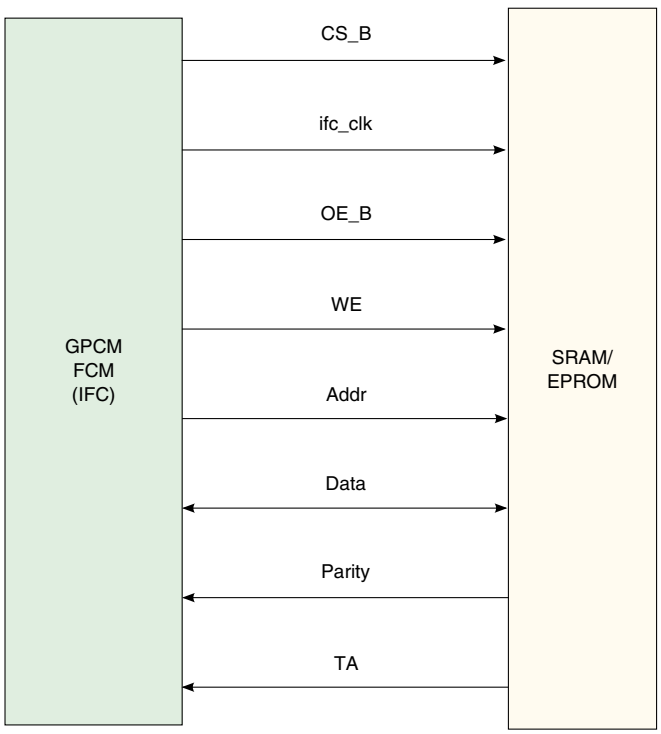


Figure 13-265. Normal GPCM interface

13.4.4.1.1 Normal GPCM program operation

In normal GPCM, the write transaction depends on the register field CSORn[WEGTA].

The following options, through an internal counter or through an external device giving an access termination signal, are explained below.

13.4.4.1.1.1 Normal GPCM internal counter-based program operation

Transactions are based on control-signal timings that are programmed in the flash timing register for GPCM mode (IFC_FTIM*x*_CS*n*_GPCM) when CSOR*n*[WGETA]=0.

Assertion of AVD/ALE is aligned to the rising edge of ifc_launch_clk after that all the timing parameters are in terms of the IFC module input clock.

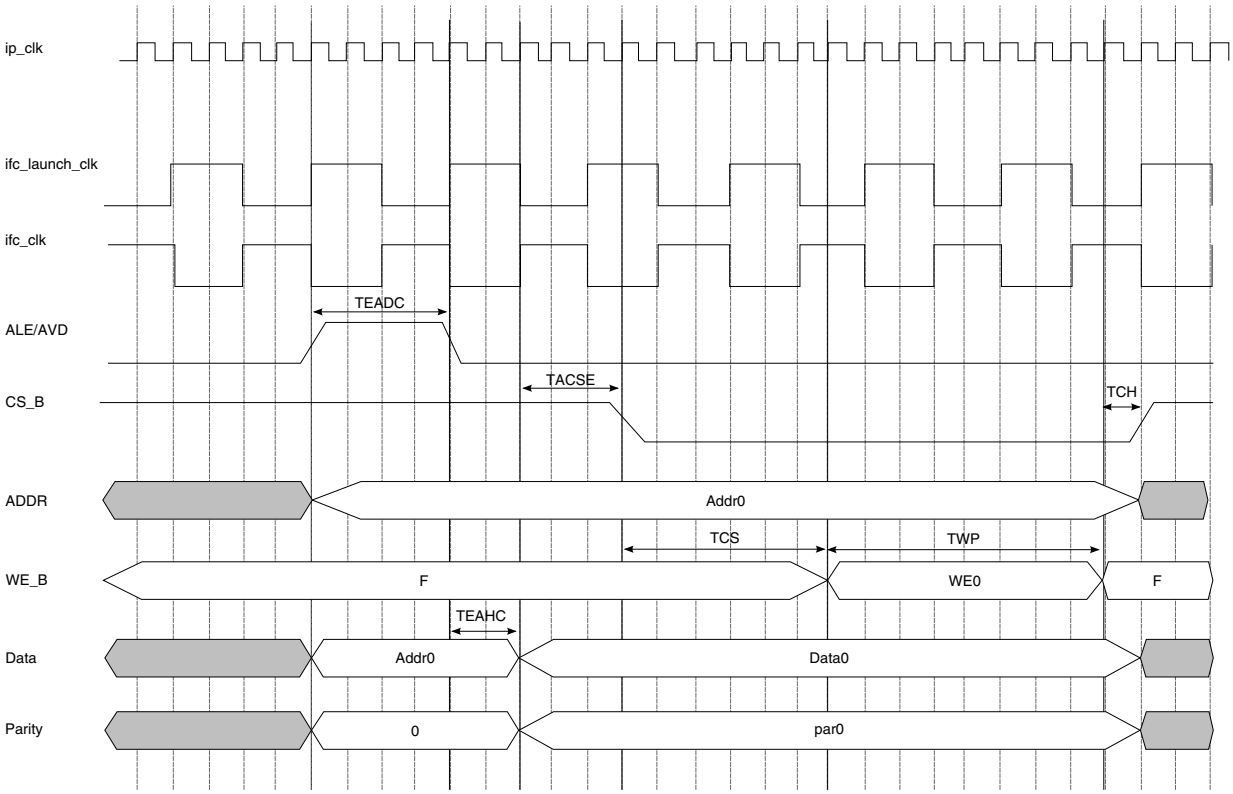


Figure 13-266. Normal GPCM program operation - non-burst mode

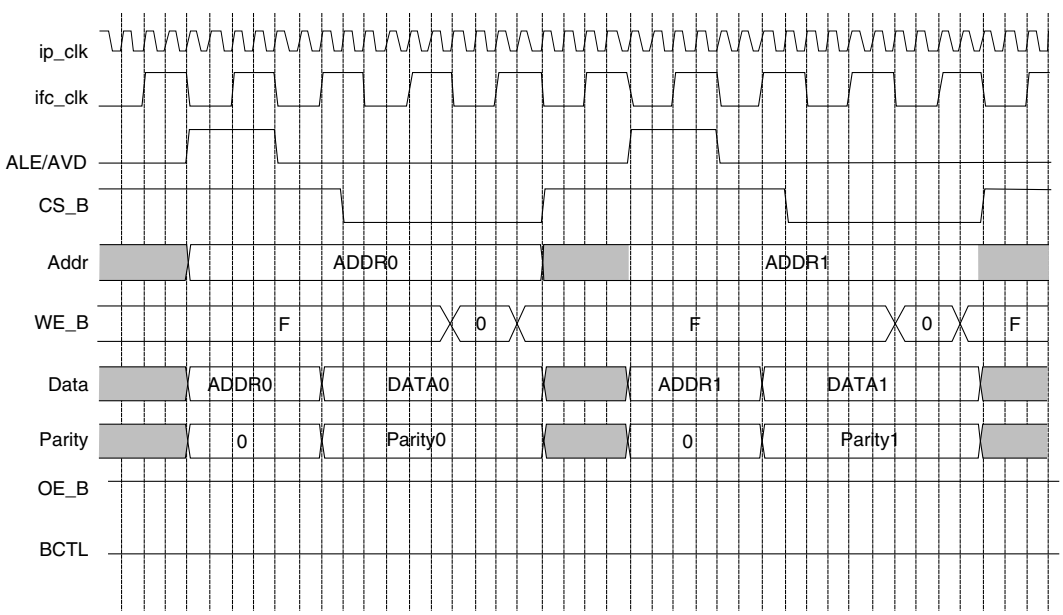


Figure 13-267. Normal GPCM back-to-back program operation

In burst mode, the next data and next address are sent after one TWP period (the amount of time between address/data cycles). If ifc_ta_b is deasserted and the burst is controlled by the timing parameter programmed in FTIM registers, the burst will continue until either the burst length is reached or complete data is transferred. If the number of bytes of data to be transferred is less than programmed burst length multiplied by port size, then the burst will be terminated before the programmed maximum number of burst transfers.

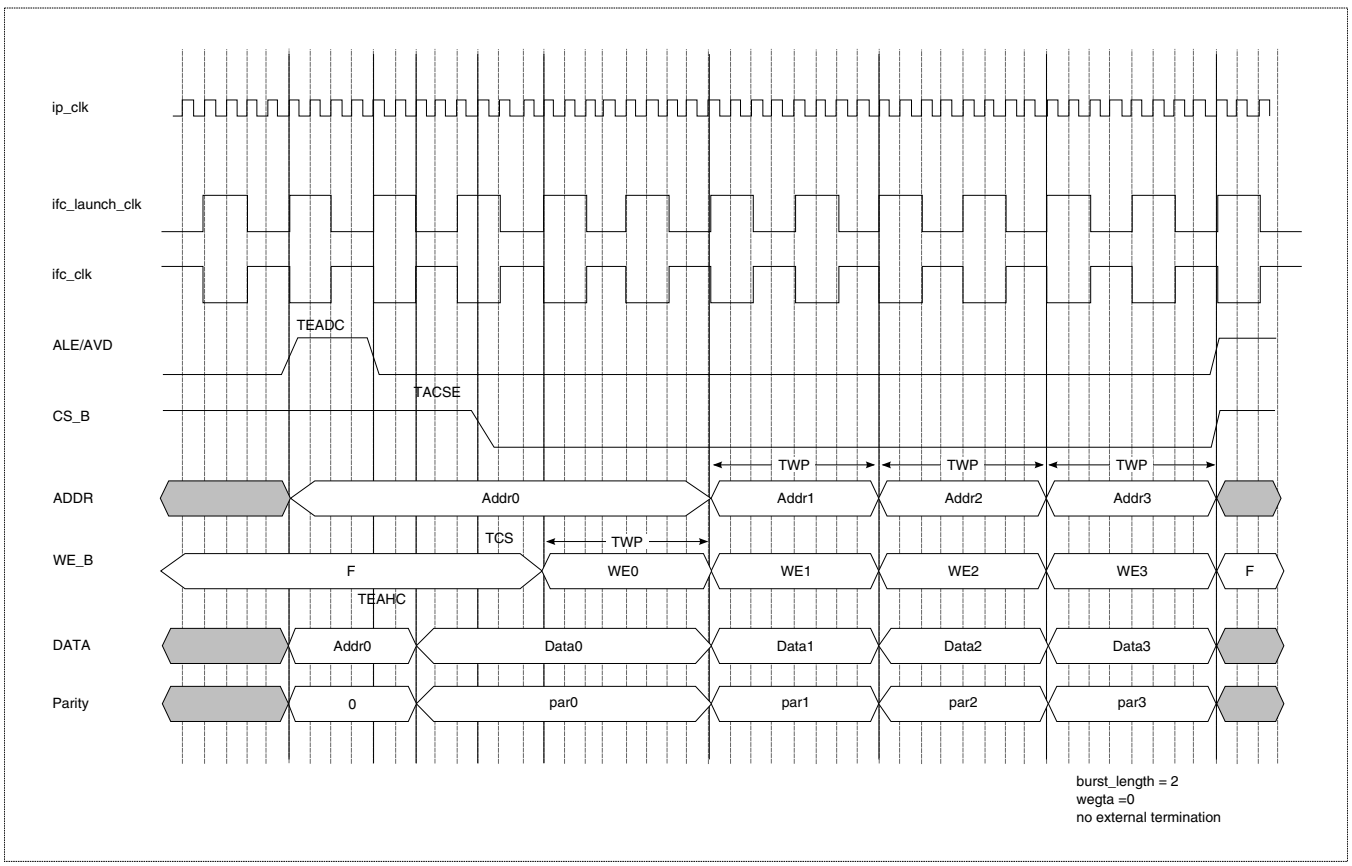


Figure 13-268. Normal GPCM program operation - burst mode

For a normal timing-based transaction, CSOR*n*[WGETA]=0. If the access is aborted by ifc_ta_b and there is remaining data to be transferred, the current burst is terminated and a new burst transaction is launched. Ongoing burst terminates at the next rising edge of ifc_launch_clk after the assertion of ifc_ta_b. The next burst's starting address is the last burst address + the number of bytes that needed to be transferred in the last burst.

$Address_{n+1} = Address_n + (number\ of\ bytes\ for\ current\ burst\ /\ port\_size)$

The next burst starts only after deassertion of ifc_ta_b. This is registered as an abort error in the status register and all transaction attributes are locked.

Since ifc_ta_b is synchronized through asynchronous FIFO, bus termination occurs only after a synchronization delay of ifc_ta_b. ifc_ta_b should be asserted for at least one ifc_clk cycle if it is synchronous (that is, meeting setup and hold time); otherwise, asynchronous ifc_ta_b should be asserted for a minimum of two ifc_clk cycles.

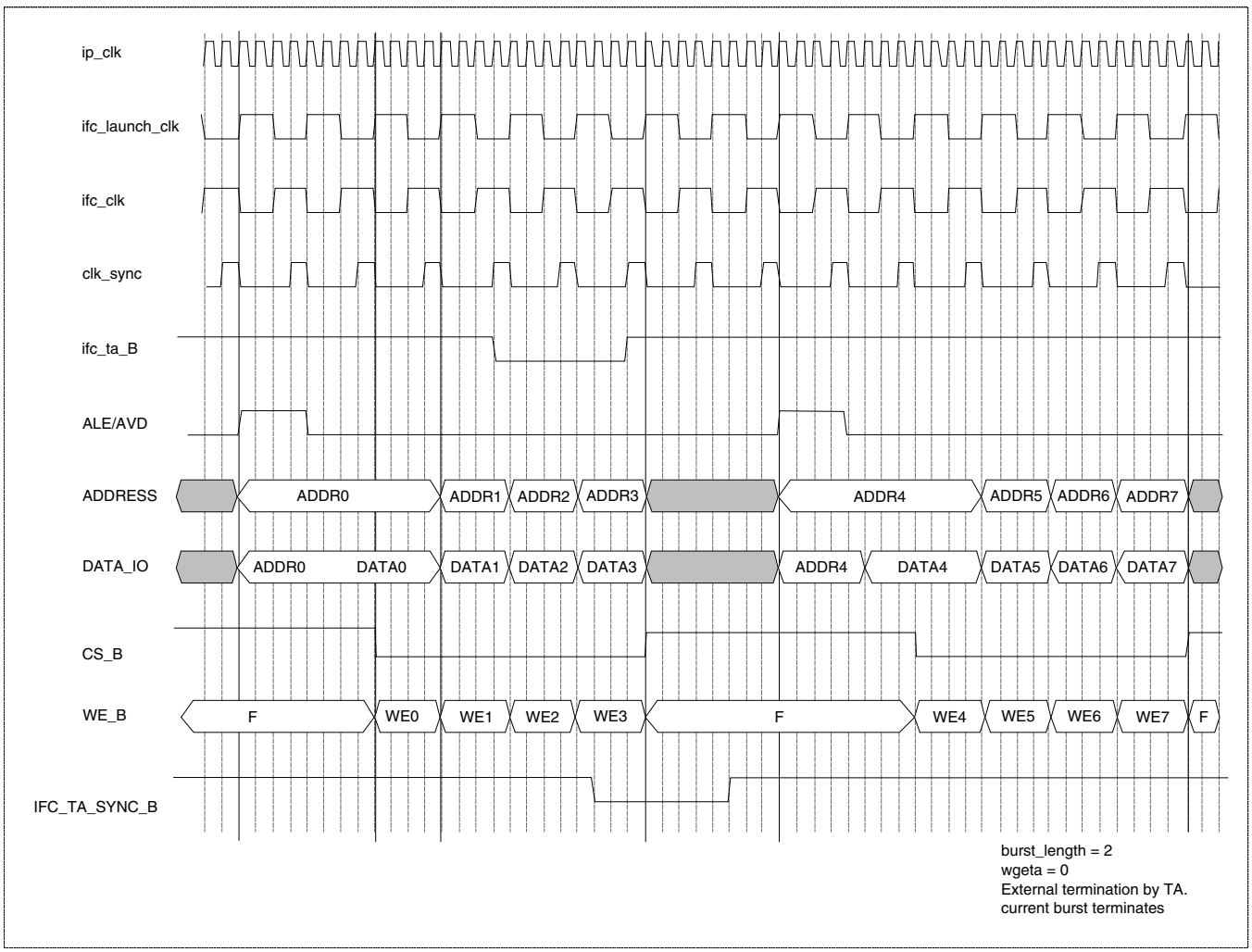


Figure 13-269. Normal GPCM program operation - transaction aborted

13.4.4.1.1.2 Normal GPCM external termination-based program operation

When WGETA is set, the next address and data are sent only after sampling the rising edge of ifc_ta_b or after timeout.

ifc_ta_b is synchronized to the IFC module input clock through asynchronous FIFO and only after the rising edge of ifc_ta_b, the new beat is sent over the interface. The next data and address beat is sent in sync with the positive edge of ifc_launch_clk. The last beat of the last burst terminates at the next rising edge of ifc_launch_clk after assertion of ifc_ta_b, without waiting for deassertion of ifc_ta_b.

For non-burst mode current access is terminated at the next rising edge of ifc_launch_clk after assertion of ifc_ta_b or after timeout.

As ifc_ta_b is synchronized through asynchronous FIFO, bus termination occurs only after the synchronization delay of the ifc_ta_b signal. ifc_ta_b should be asserted for at least one ifc_clk cycle if it is synchronous (that is, meeting setup and hold time); otherwise, if it is asynchronous, ifc_ta_b should be asserted for a minimum of two ifc_clk cycles.

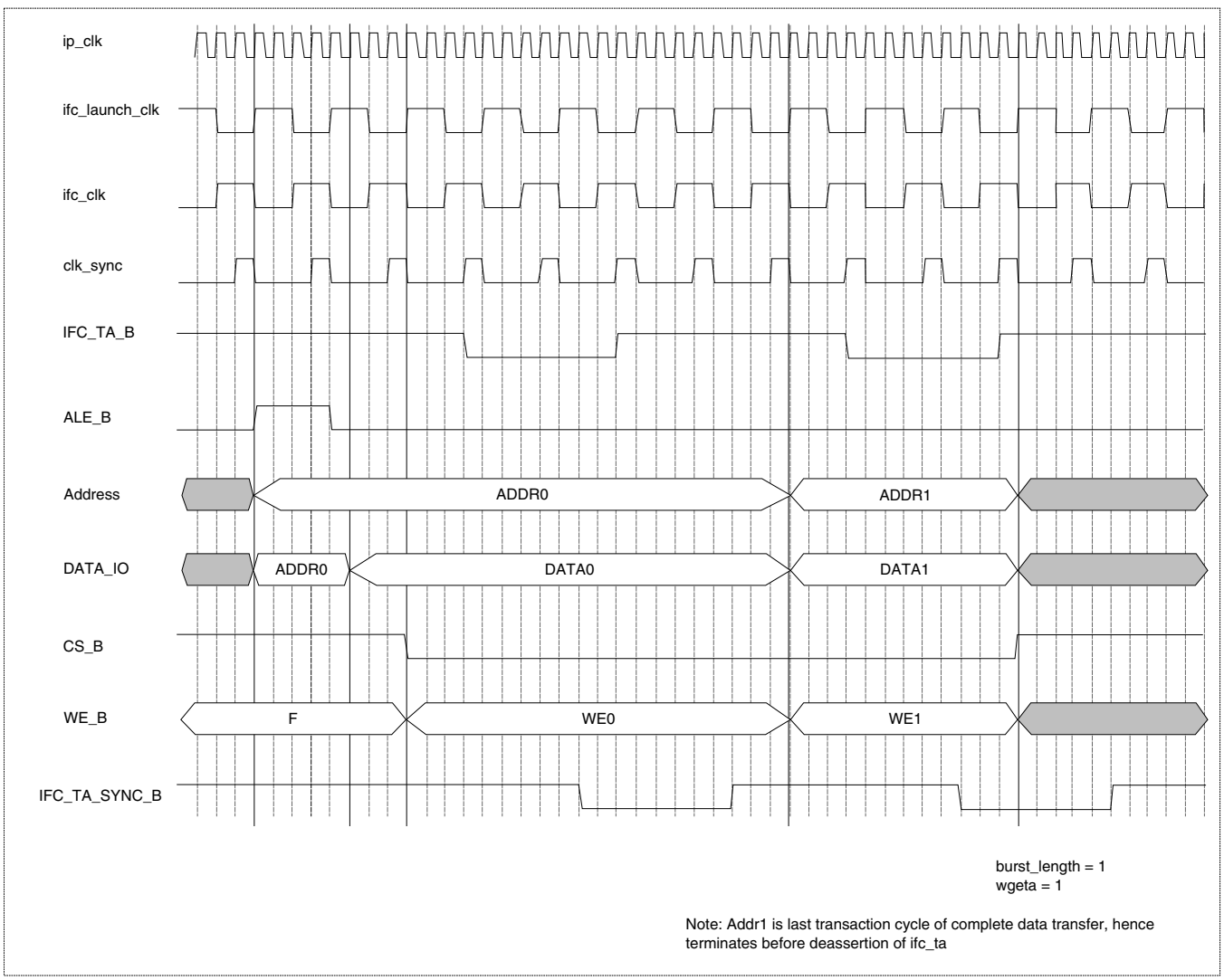


Figure 13-270. Normal GPCM program operation - acknowledgment mode

13.4.4.1.2 Normal GPCM read operation

In normal GPCM, the read operation depends on the register field CSOR*n*[RGETA]. The following options, through an internal counter or through an external device giving an access termination signal, are explained below.

13.4.4.1.2.1 Normal GPCM internal counter-based read operation

In this approach, read data from memory is sampled based on the programming of FTIM1_CSn_GPCM[TRAD] and FTIM3_CSn_GPCM[TAAD].

In non-burst mode, TRAD defines the pulse width of OE, which is synchronized through the asynchronous FIFO, along with the data and parity from memory. Data from the memory will be sampled after TRAD time of OE assertion.

By default, (CCR[INV_CLK_EN]=1 and CCR[CLK_DLY]=0) inverted clock of ifc_launch_clk goes out (ifc_clk) to the external device. Because the address is launched at the rising edge of ifc_launch_clk, the setup time on an external device for the address phase is half the cycle period of ifc_clk. The read data launched by the device will be sampled by the IFC at the next rising edge of ifc_clk. The programming of TRAD_GPCM should be as follows:

- For synchronous interfaces, TRAD programmed will be:

  (1 + *n*) x (CCR[CLK_DIV] + 1)

- where,
  - *n* defines the memory access time in terms of ifc_clk.
  - Memory takes *n* ifc_clk to output data after output enable is asserted.
  - *n* can take values 0,1,2,3....

The following figures explain the above-mentioned read data sampling schemes:

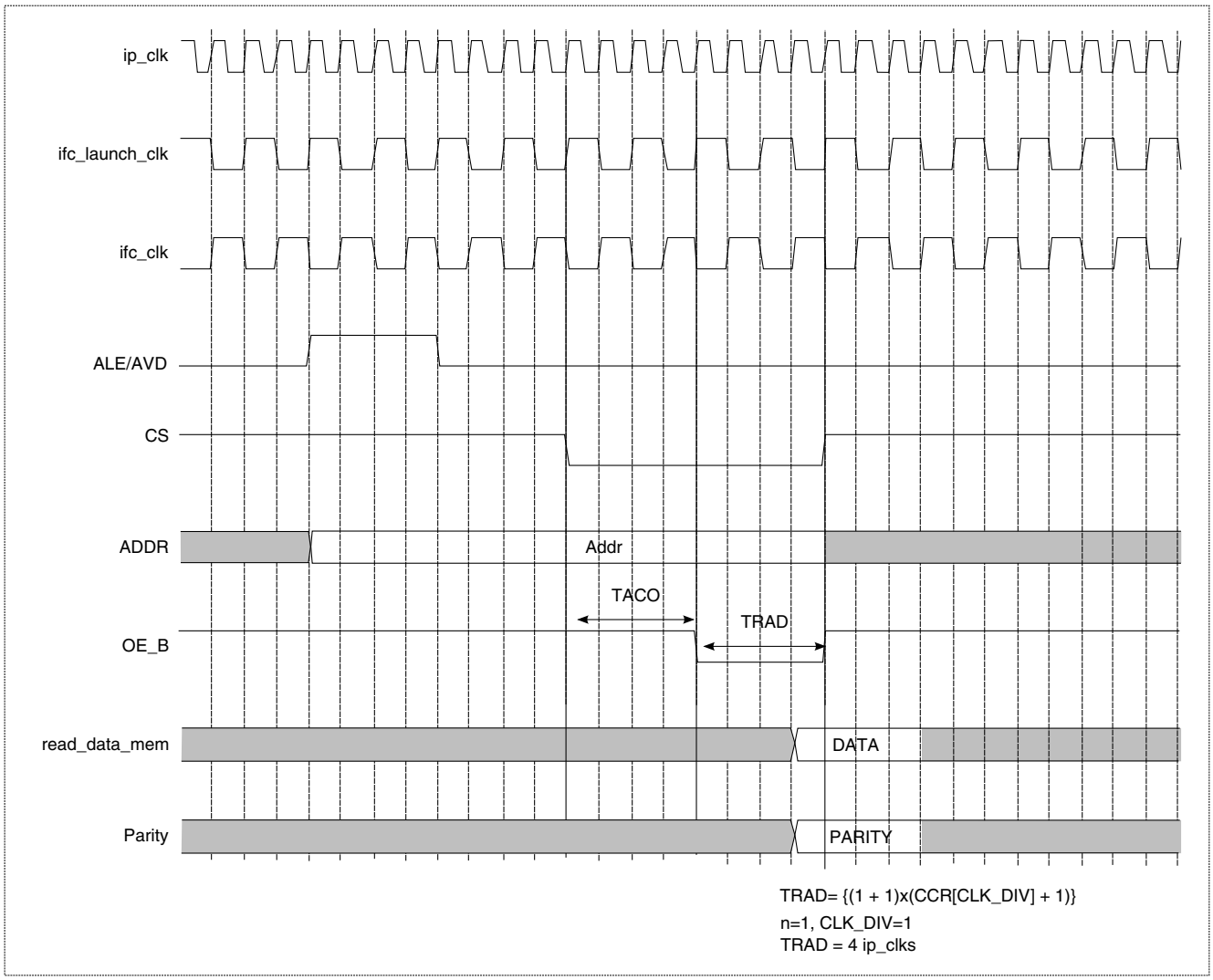
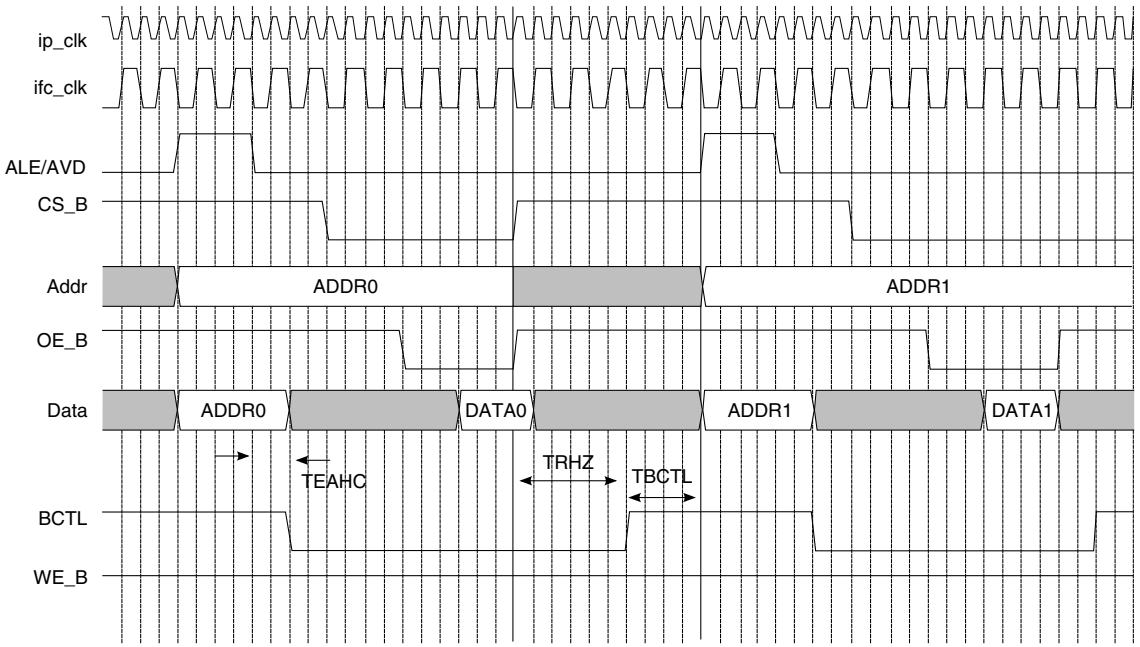


Figure 13-271. Normal GPCM read operation - non-burst mode



TRHZ: Extended hold time for slow memories to disable their bus drivers
TBCTL: Bus turn-around time

Figure 13-272. Normal GPCM back-to-back read operation

When RGETA is programmed to 0 and access is aborted by ifc_ta, then the current burst is terminated and new burst transaction is launched. The ongoing burst is terminated at the next rising edge of ifc_launch_clk after assertion of ifc_ta. The next burst starting address will be the last burst address + number of bytes that needed to be received in the last burst.

$Address_{n+1} = Address_n + (number\ bytes\ for\ current\ burst\ /\ port\_size)$

The next burst starts only after deassertion of ifc_ta.

An abort error is registered in the status registers and all the trasaction attributes are locked. Since ifc_ta is synchronized through asynchronous FIFO, bus termination occurs only after synchronization delay of ifc_ta signal. ifc_ta should be asserted for atleast one ifc_clk cycle if it is synchronous (that is, meeting setup and hold time), otherwise if asynchronous ifc_ta should be asserted for minimum two ifc_clk cycles.

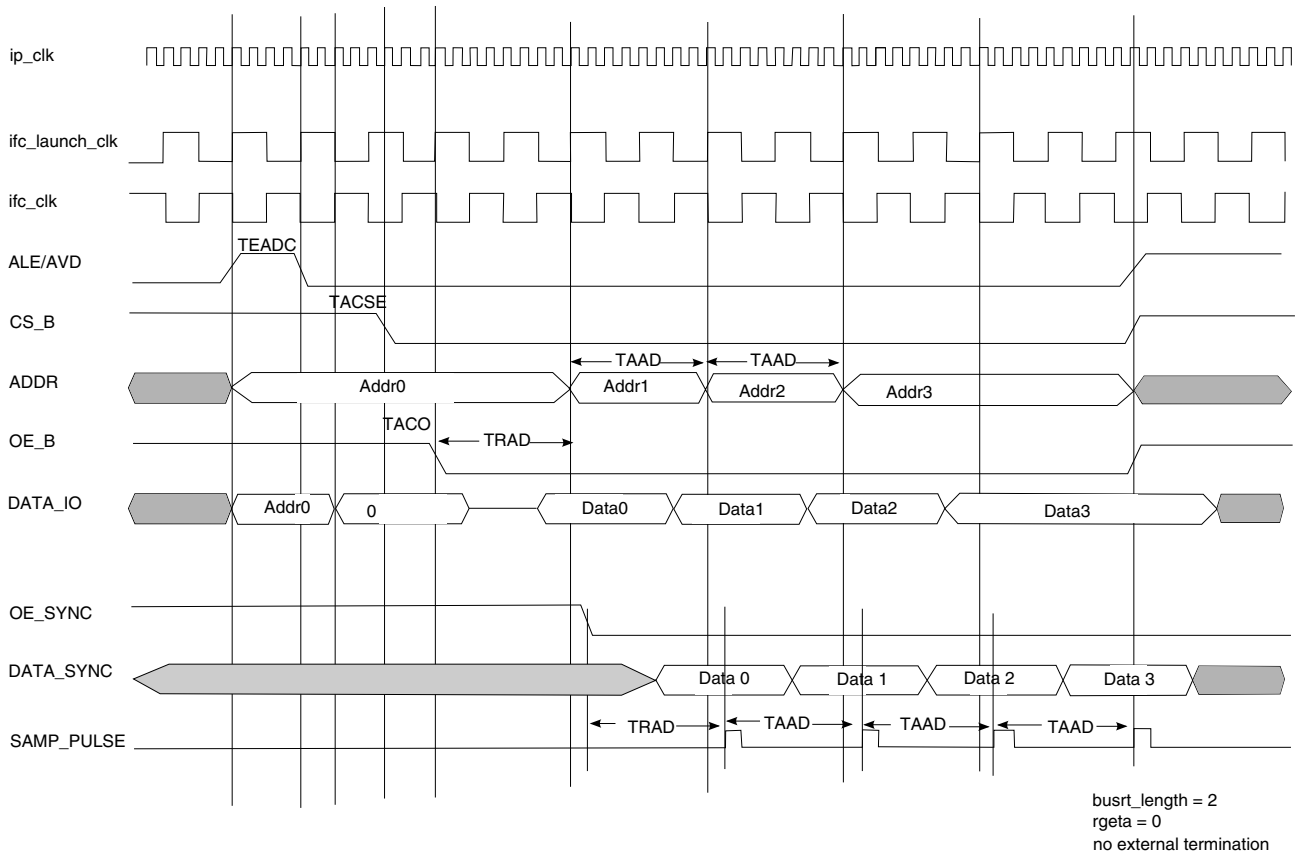


Figure 13-273. Normal GPCM read operation - burst mode

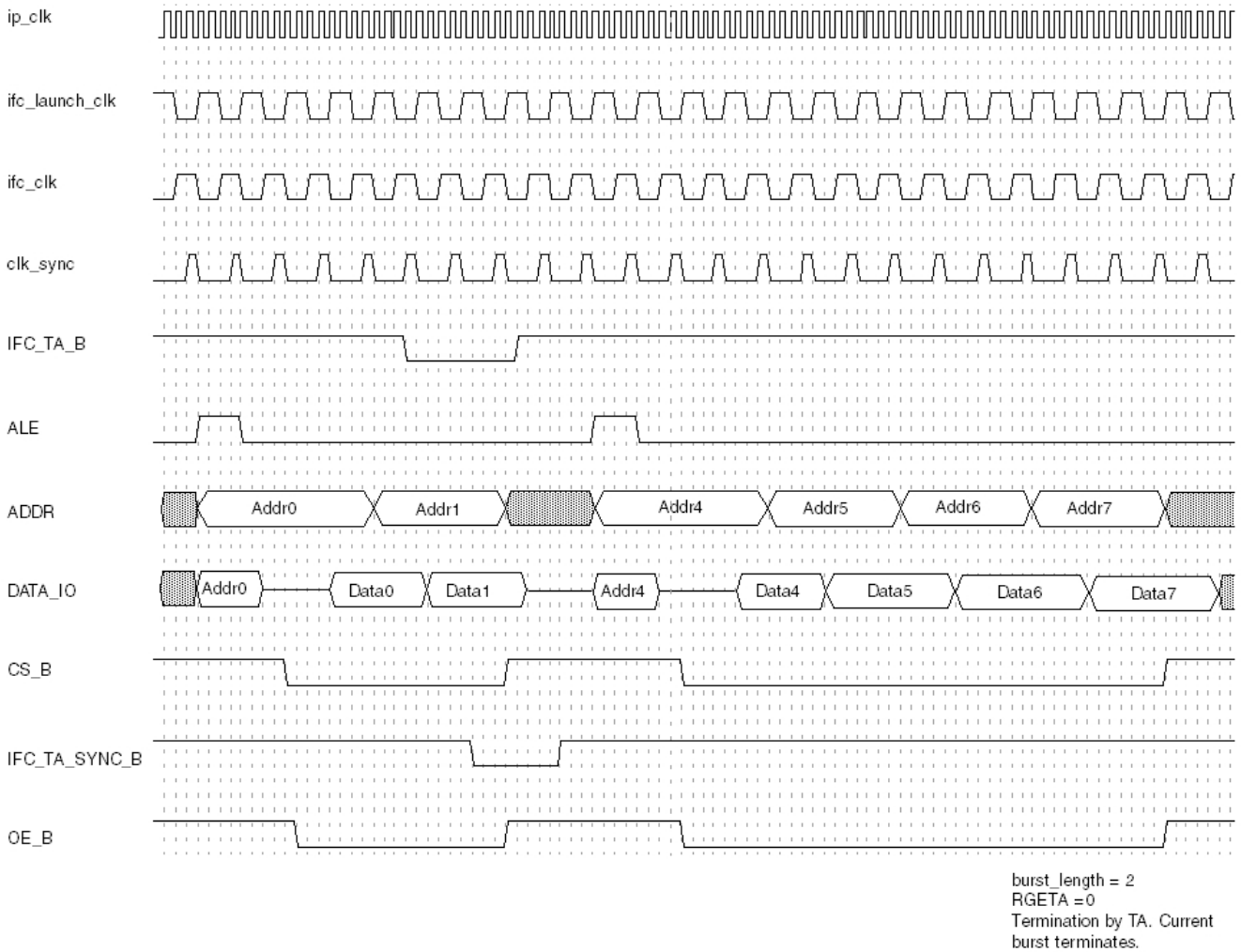


Figure 13-274. Normal GPCM read operation - transaction aborted

13.4.4.1.2.2 Normal GPCM external termination-based read operation

In cases where the external device gives access termination signal through IFCTA_B, current access is terminated by deasserting the signals on the next ifc_clk.

Read access are terminated by deasserting OE_B on the next rising edge of ifc_launch_clk, and read data are sampled based on this signal. Write access is terminated by deasserting WE_B and CS_B. Deassertion of the WE signal is aligned to the rising edge of ifc_launch_clk.

In this mode, the IFCTA_B signal are synchronized through the async FIFO along with the read data and corresponding parity. Note that because IFCTA_B is synchronized, bus termination occurs only after synchronization delay of IFCTA_B signal; so, in case of a read cycle, the device still must drive data as long as OE_B is asserted. Deassertion of OE_B and CS_B is aligned to the rising edge of ifc_launch_clk. IFCTA_B should be

asserted for at least one ifc_clk cycle, if it is synchronous (that is, meeting set up and hold time); otherwise, asynchronous IFCTA_B should be asserted for minimum two ifc_clk cycles, as this figure shows.

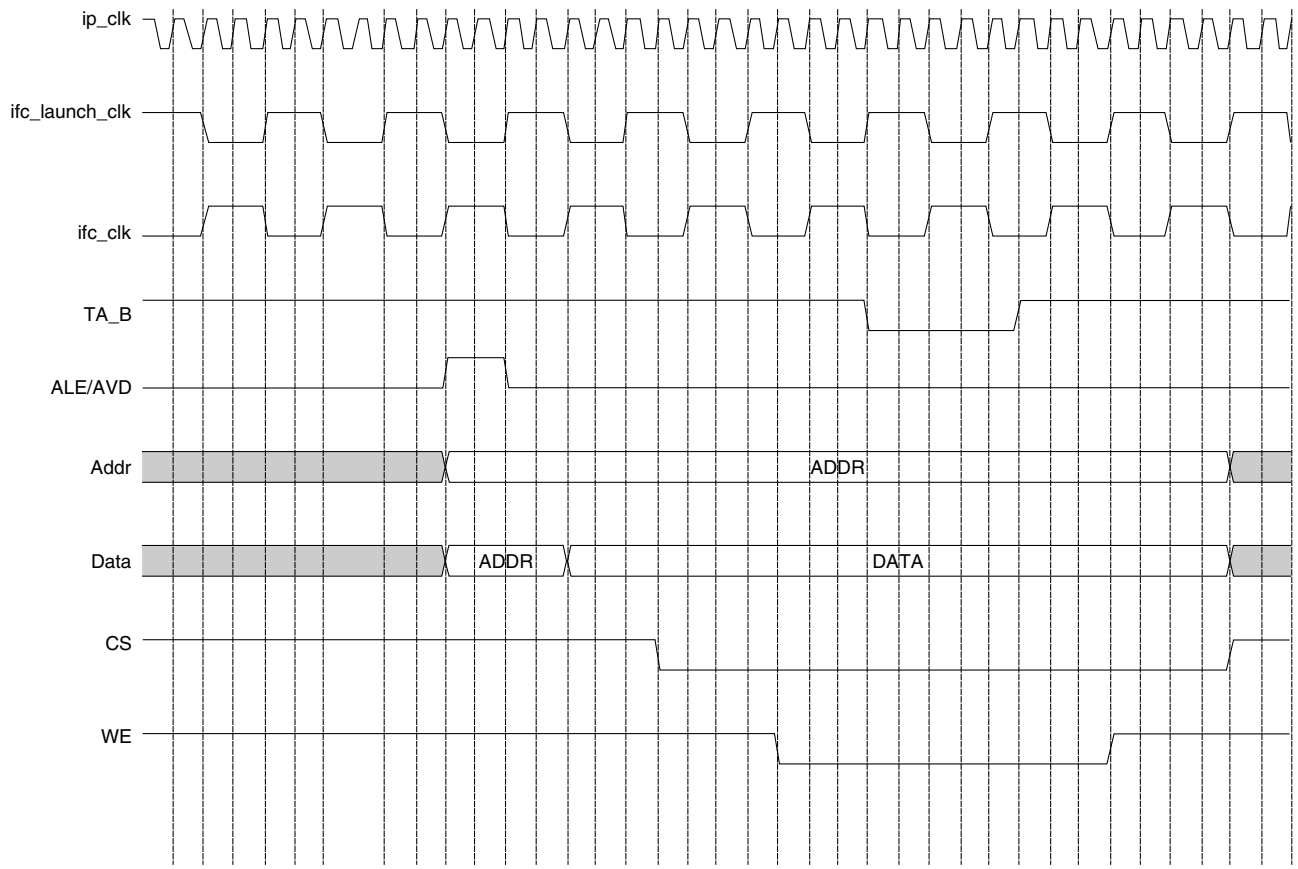


Figure 13-275. Normal GPCM access termination with IFCTA (write access)

13.4.4.2 Generic ASIC mode of operation

This figure explains the interface of the IFC to a device connected through a customized GPCM generic ASIC FCM.

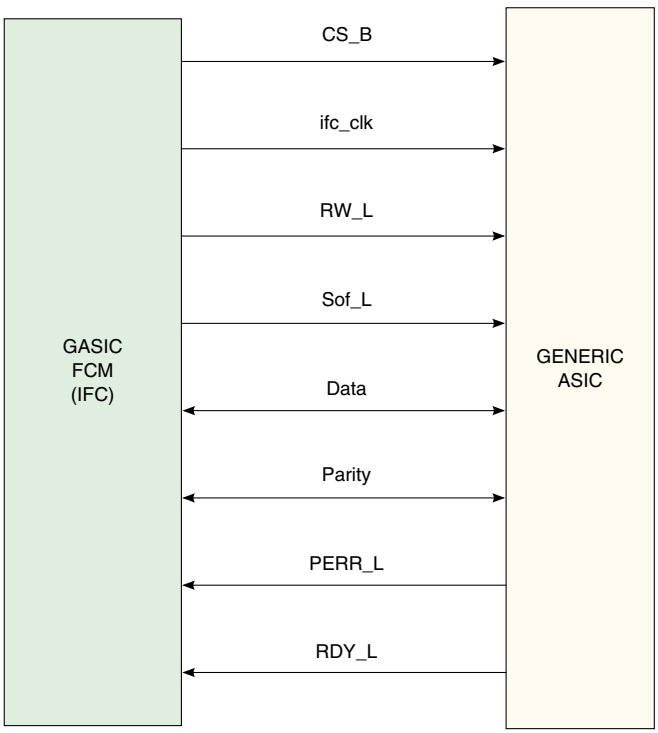


Figure 13-276. Generic ASIC interface

13.4.4.2.1 General ASIC program operation

Before initiating any general ASIC (GASIC) operation, the chip-select (CS_L) is asserted to select the appropriate device.

The assertion of start of frame (SOF_L) along with the low value on RW_L indicates the start of address phase for write transfer. In the case of a 16-bit device, the upper 16 bits of address is driven on the AD[0:15] first, followed by the lower 16 bits on the next clock. After the address phase, the write data sequences are driven on AD lines based on the port size. Along with the address and write data phases, per-byte parity is driven on the PAR lines. On the completion of write data phase, the host controller waits for the ready status (RDY_L) and the parity error sampling before deasserting the chip-select to complete the transfer. In the case where the device is not asserting RDY_L before timeout occurs, the host terminates the transfer by deasserting the CS_L.

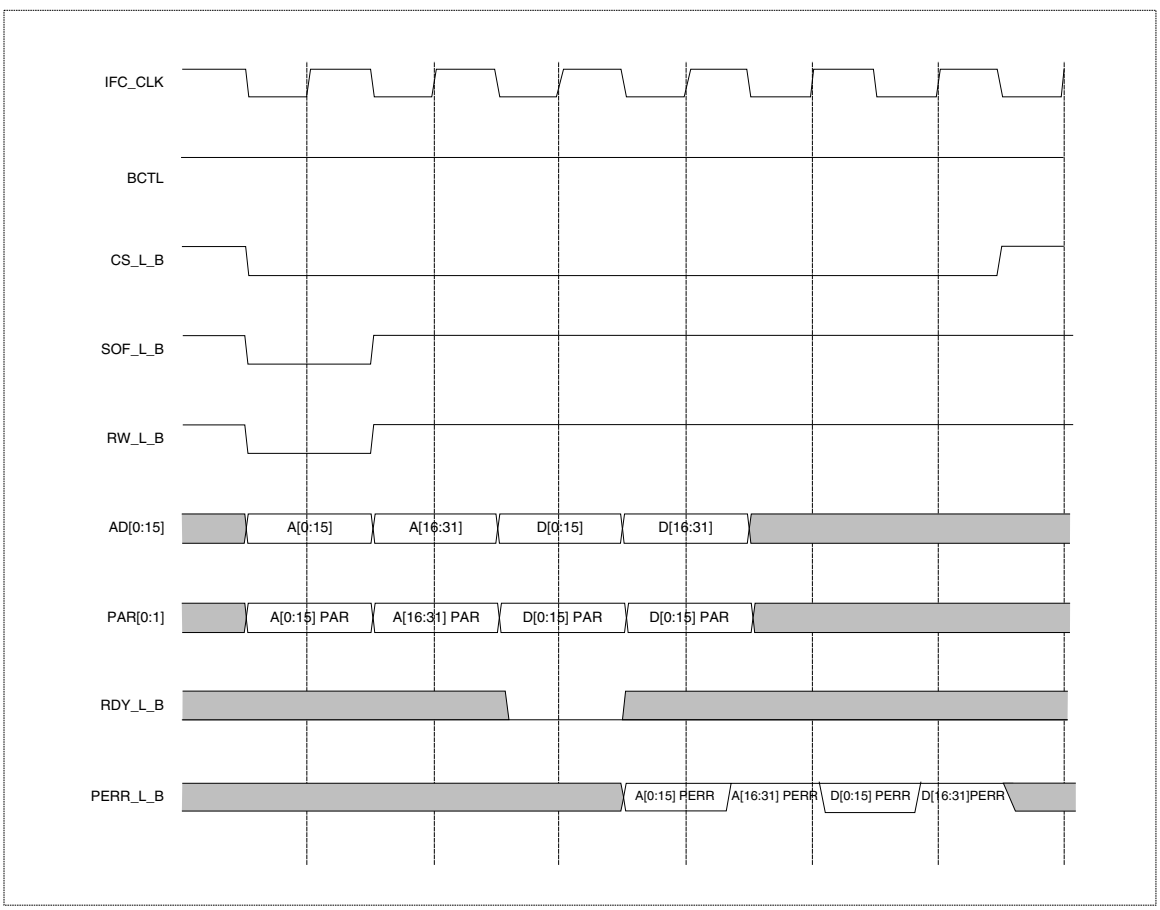


Figure 13-277. GASIC program operation with 16-bit device and zero-wait state

13.4.4.2.2 Generic ASIC read data sampling

After selecting the general ASIC (GASIC) device, the host asserts start of frame (SOF_L) and drive high value on RW_L to indicate the start of address phase for read transfer.

The host drives the AD lines and the corresponding PAR lines in this phase to send the 16 -bit of address to the device. After completing the address phase, the host stops driving the AD lines and waits for the assertion of ready status (RDY_L) from the device. The assertion of RDY_L indicates the start of read data phase. In the case the device has not asserted RDY_L before timeout occurs, the host terminates the transfer by deasserting the CS_L.

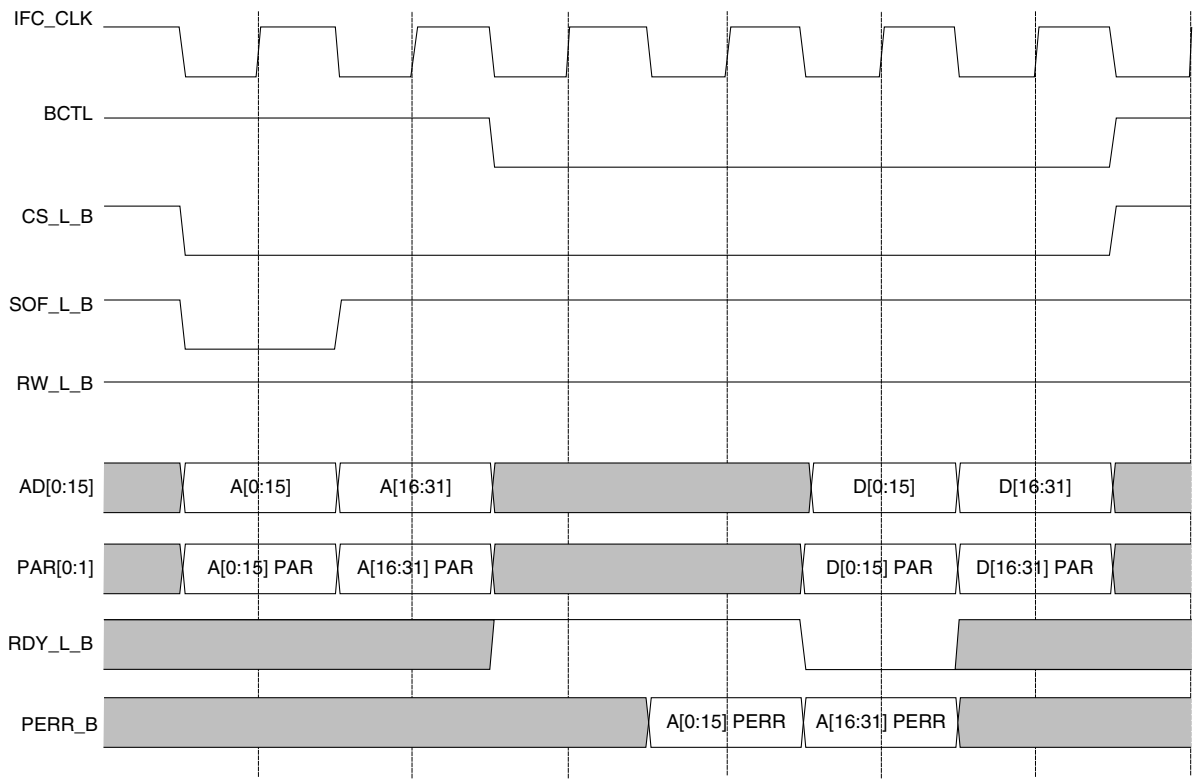


Figure 13-278. GASIC read operation with 16-bit device and 2-wait state

13.4.5 Clock generation module

The clock generation module generates a divided clock derived from the IFC module input clock depending on the programmed value in the CCR register.

Whenever there is a change in any clock division ratio, the clock is gated for the IFC module input clocks. The clock status is reported as unstable in the CSR register.

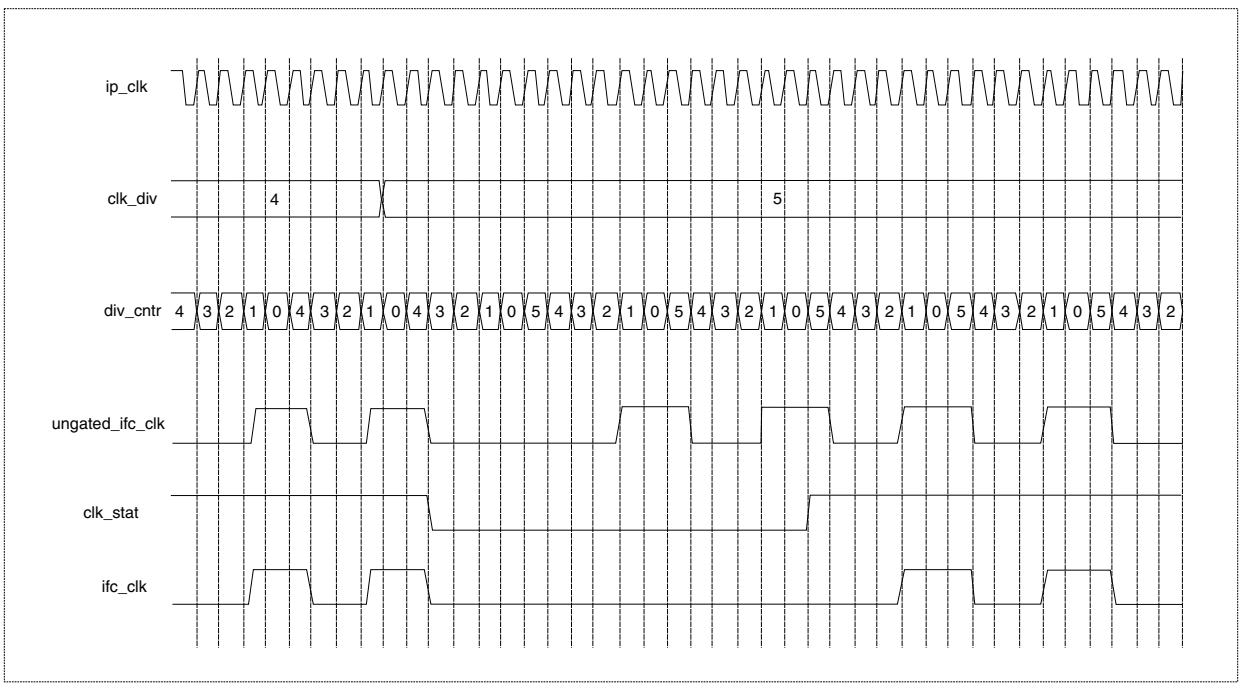


Figure 13-279. External clock generation

13.5 Initialization/Application information

The IFC can be used with separate address and data buses or with a multiplexed address/data bus.

This section provides guidelines for interfacing peripherals in the IFC's various modes.

13.5.1 IFC flash connections

13.5.1.1 NAND flash connections

The NAND FCM provides a glueless interface to 8- or 16-bit parallel-bus NAND flash EEPROM devices.

This figure shows a simple connection between an 8-bit port size NAND flash EEPROM and the IFC. In NAND FCM mode, commands, address bytes, and data are all transferred on AD[0:7].

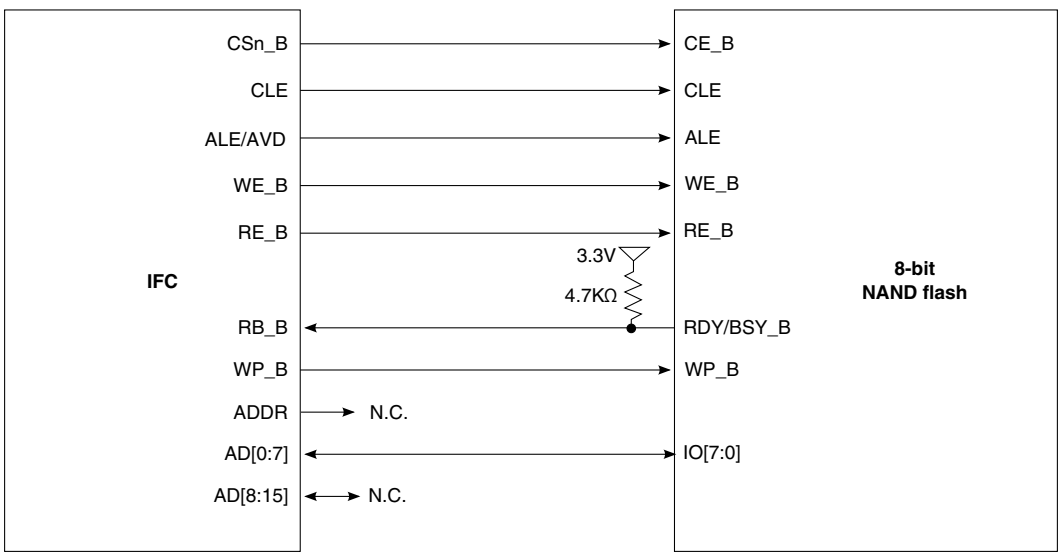


Figure 13-280. IFC to 8-bit asynchronous NAND device interface

This figure shows connection of a 16-bit NAND flash to IFC. Commands and address bytes appear on AD[8:15].

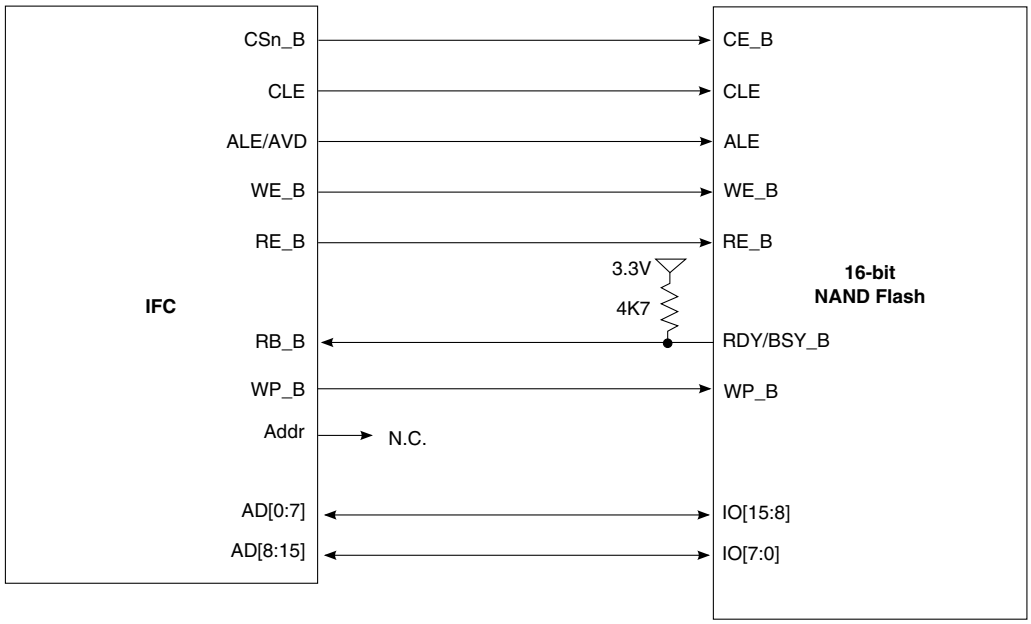


Figure 13-281. IFC to 16-bit asynchronous NAND flash device interface

13.5.1.2 NOR flash connections

For 16-bit devices, ADDR[30] is used and ADDR[31] is irrelevant; however, for 8-bit devices, ADDR[30:31] are necessary.

This figure shows the IFC-to-NOR device interface.

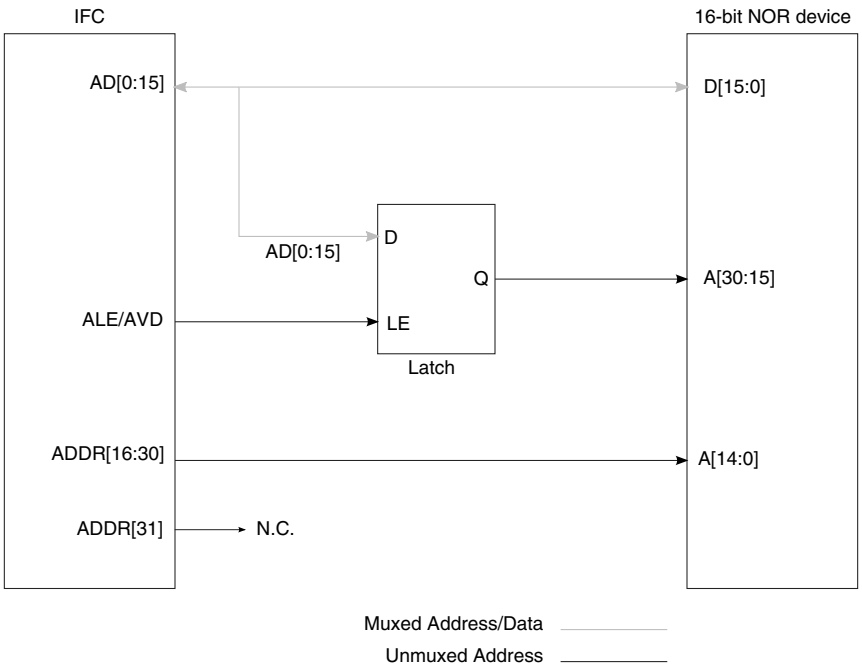


Figure 13-282. IFC to 16-bit NOR device interface

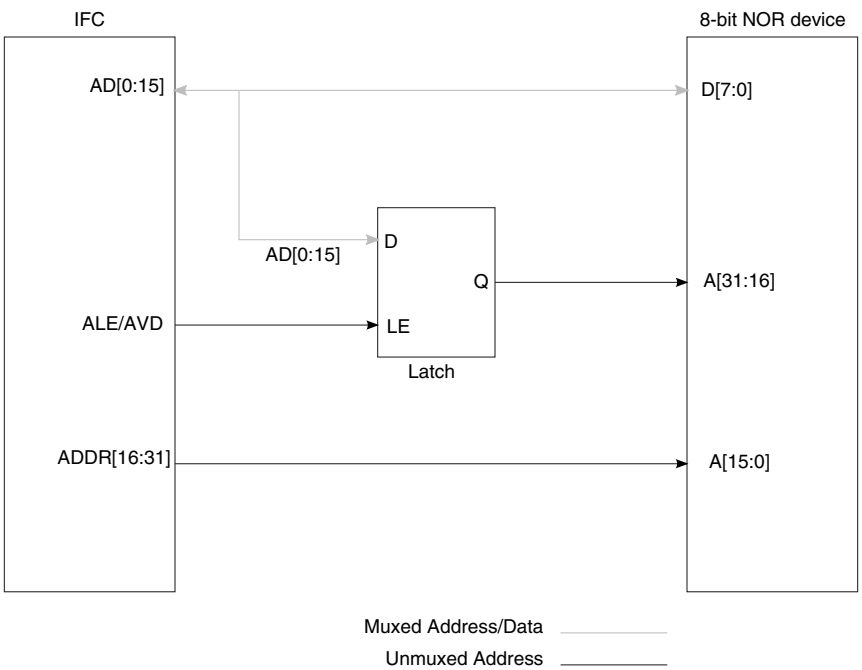


Figure 13-283. IFC to 8-bit NOR device interface

13.5.2 Bus turnaround

Because the IFC uses multiplexed address and data, special consideration is given to avoid bus contention at bus turnaround.

The following cases must be examined:

- Address phase after previous read
- Read-data phase after address phase
- Read-modify-write cycle for parity-protected memory banks

The bus does not change direction for the following cases, so no special attention is required:

- Continued burst after the first beat
- Write-data phase after address phase
- Address phase after previous write

13.5.2.1 Address phase after previous read

During a read cycle, the memory/peripheral drives the bus and the bus transceiver drives AD.

After the data has been sampled, the output drivers of the external device must be disabled, which can take some time.

After the previous cycle ends, BCTL goes high and changes the direction of the bus transceiver. The IFC then inserts a bus turnaround time (that is, TBCTL) to avoid contention. The external device has now already placed its data signals in high impedance, and no bus contention occurs.

13.5.2.2 Read-data phase after address phase

During the address phase, AD actively drives the address and BCTL is high, driving the bus transceivers in the same direction as during a write.

After the end of the address phase, BCTL goes low and changes the direction of the bus transceiver.

13.5.2.3 Read-modify-write cycle for parity protected memory banks

Principally, a read-modify-write cycle is a read cycle immediately followed by a write cycle. Because the write cycle has a new address phase in any case, this essentially is the same as an address phase after a previous read.

13.5.3 Interfacing to different port sizes

The IFC supports 8- and 16-bit data port sizes.

However, the IFC requires that the portion of the data bus used for a transfer to or from a particular port size be fixed. A 16-bit port must reside on AD[0:15], and an 8-bit port on AD[0:7]. This figure shows the device connections on the data bus.

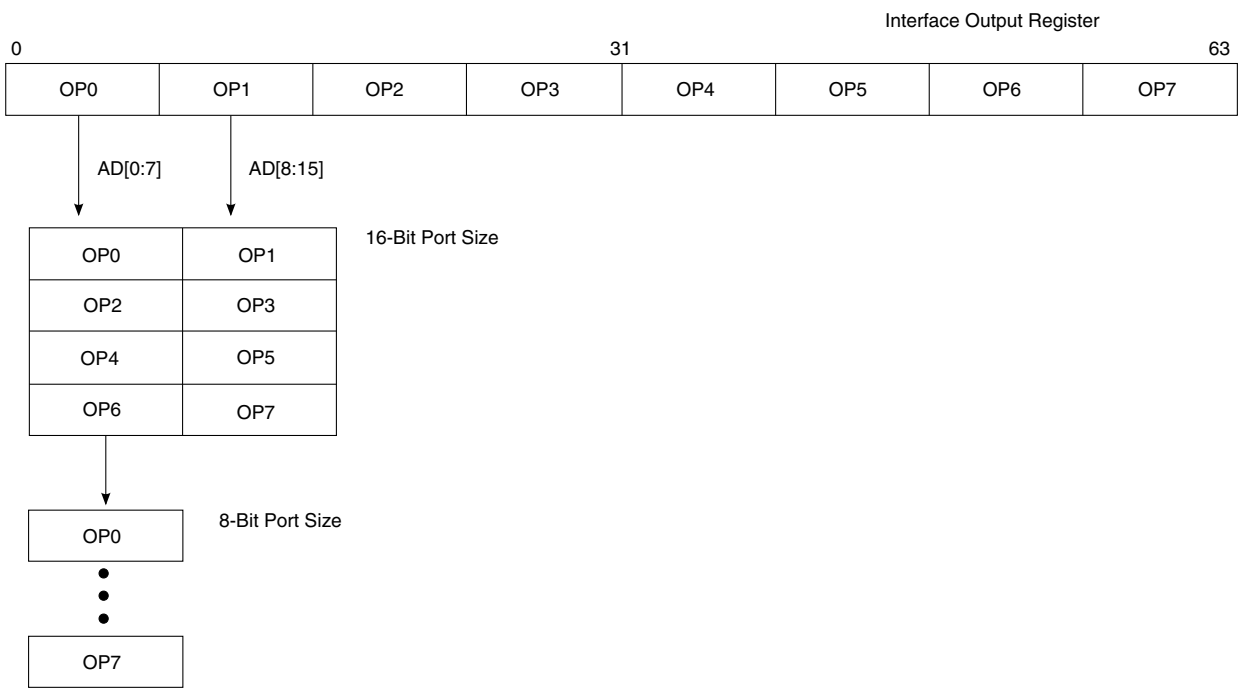


Figure 13-284. Interface to different port-size devices

13.5.4 Command sequence examples for NAND flash EEPROM

To program the IFC and FCM for executing NAND flash command sequences, obtain command codes and pause states from the relevant NAND flash device datasheet and programmed into FCM configuration registers.

This section describes some common sequences for multi-gigabit NAND flash EEPROMs; however, details should be verified against manufacturer's specific programming data.

Throughout these examples, it is assumed that one or more banks of the IFC have been configured under FCM control with base address, port size, ECC mode, and timing parameters configured in accordance with the device's data sheet .

13.5.4.1 NAND flash soft reset command sequence example

An example of configuring FCM to execute a soft reset command to a 2 KB-page NAND flash is shown in this table.

This sequence does not require the use of the shared FCM buffer RAM. At the conclusion of the sequence, IFC issues a command complete interrupt if interrupts are enabled.

.

Table 13-249. FCM register settings for soft reset

Register	Initial contents	Description
NAND_FCR0	FF00_0000h	CMD0 = FFh = reset command; other commands unused
ROW0	-	Unused
COL0	-	Unused
NAND_BC	-	Unused
NAND_FSR	-	Unused
NAND_FIR0, NAND_FIR1, NAND_FIR2	2400_0000h, 0000_0000h, 0000_0000h	OP0 = CMD0 = command 0; OP1-OP14 = NOOP

13.5.4.2 NAND flash read status command sequence example

An example of configuring FCM to execute a status read command to 2 KB-page NAND flash is shown in this table.

This sequence does not require the use of the shared FCM buffer RAM, but reads the NAND flash status into NAND_FSR.

At the conclusion of the sequence, the IFC issues a command complete interrupt if interrupts are enabled.

Table 13-250. FCM register settings for status read

Register	Initial contents	Description
NAND_FCR0	7000_0000h	CMD0 = 70h = read status command; other commands unused
ROW0	-	Unused
COL0	-	Unused
NAND_BC	-	Unused
NAND_FSR	-	Status returned in RS0
NAND_FIR0, NAND_FIR1, NAND_FIR2	25C0_0000h, 0000_0000h, 0000_0000h	OP0 = CMD0 = command 0; OP1 = RDSTAT = read status to NAND_FSR; OP2-OP14 = NOOP

13.5.4.3 NAND flash read identification command sequence example

An example of configuring FCM to execute a status ID command to 2 KB-page NAND flash is shown in this table.

This sequence uses the shared FCM buffer RAM to receive the bytes of ID during the sequence.

At the conclusion of the sequence, IFC issues a command complete interrupt if interrupts are enabled.

Table 13-251. FCM register settings for ID read

Register	Initial contents	Description
NAND_FCR0	9000_0000h	CMD0 = 90h Read ID command; other commands unused
ROW3	Row address (0000_0020h)	Locates the block and the page within that block to be accessed
COL0	0000_0000h	Locates the byte to be accessed within a given page MS = 0
NAND_BC	0000_0004h	BC = 4 to read the ONFI Signature (for ONFI devices)
NAND_FSR	-	Unused
NAND_FIR0, NAND_FIR1, NAND_FIR2	2608_4000h, 0000_0000h, 0000_0000h	OP0 = CMD0 = command 0; OP1 = UA = Send row address programmed in row address register 3. Use SRAM buffer 0 to store the read ID bytes; OP2 = RB = Read BC bytes of data into the SRAM buffer 0; OP3-OP14 = NOOP

The above sequence uses opcode UA to send the address stored in row address register 3. This is done so that the data transfer, that is, the read data is stored in buffer 0 of the internal SRAM. Thus, the UA could also be used for operations, such as read page parameter, get feature, set feature, read unique ID always using buffer 0 of the internal SRAM.

13.5.4.4 NAND flash page read command sequence example

An example of configuring FCM to execute a random page read command to 2-Kbytes page NAND flash is shown in the table below.

This sequence reads an entire page (main and spare region) into the shared FCM buffer RAM, checking ECC (if enabled) as it proceeds.

At the conclusion of the sequence, IFC will issue a command complete interrupt if interrupts are enabled. Once the sequence has completed, the shared buffer (buffer 1 for page index 5) and transfer error registers are valid.

Table 13-252. FCM register settings for page read

Register	Initial contents	Description
NAND_FCR0	0030_0000h	CMD0 = 00h = random read address entry; CMD1 = 30h = read page
ROW0	row address (for example, 0000_0005h locates page5 in block0)	Locates the block and the page within that block to be accessed
COL0	0000_0000h	Locates the byte to be accessed within a given page MS = 0
NAND_BC	0000_0000h	BC = 0 to read entire 2112-byte page
NAND_FSR	-	unused
NAND_FIR0, NAND_FIR1,NAND_FIR2	2411_4A68h, 0000_0000h, 0000_0000h	OP0 = CMD0 = command 0; OP1 = CA0 = column address; OP2 = RA0 = page address; OP3 = CMD1 = command 1; OP4 = RBCD = read BC bytes of data into FCM buffer; OP5 - OP14 = NOOP

13.5.4.5 NAND flash program command sequence example

An example of configuring FCM to execute a program command to 2 KB-page NAND flash is shown in this table.

This sequence writes an entire page (main and spare region) from the shared FCM buffer RAM, generating ECC (if enabled) as it proceeds.

The shared buffer (buffer 1 for page index 5) must be initialized by software prior to starting the sequence. At the conclusion of the sequence, the IFC issues a command complete interrupt if interrupts are enabled. The status of the programming operation is returned in NAND_FSR.

Note that operations specified by OP5 and OP6 (status read) should never be skipped while programming a NAND flash device, because it may happen that a new command is issued to the NAND flash device even when the device has not yet finished processing the previous request. This may result in unpredictable behavior.

Table 13-253. FCM register settings for page program

Register	Initial contents	Description
NAND_FCR0	8070_1000h	CMD0 = 80h = page address and data entry; CMD1 = 70h = read status CMD2 = 10h = program page;
ROW0	Row address (for example, 0000_0005h locates page5 in block0)	Locates the block and the page within that block to be accessed
COL0	0000_0000h	Locates the byte to be accessed within a given page MS = 0
NAND_BC	0000_0000h	BC = 0 to read entire 2112-byte page
NAND_FSR	-	Returns with AS0 holding program status
NAND_FIR0, NAND_FIR1, NAND_FIR2	2411_592Ch, 49C0_0000h, 0000_0000h	OP0 = CMD0 = command 0; OP1 = CA0 = column address; OP2 = RA0 = page address; OP3 = WBCD = write BC bytes of data from buffer; OP4 = CMD2 = command 2; OP5 = CW1 = wait on flash ready and issue command 1; OP6 = RDSTAT = read erase status into NAND_FSR; OP7-OP14 = NOOP

13.5.4.6 Read status command during busy period of program/erase operation

During program and erase operation, the device permits the user to read status during its busy period.

To achieve this, use the sequence of commands in this table.

Table 13-254. FCM register settings for read status during program busy period

Register	Initial contents	Description
NAND_FCR0	8070_1000h	CMD0 = 80h = page address and data entry; CMD1 = 0x70 = read status CMD2 = 10h = program page;

Table continues on the next page...

Table 13-254. FCM register settings for read status during program busy period (continued)

Register	Initial contents	Description
ROW0	row address (for example, 0000_0005h locates page5 in block0)	Locates the block and the page within that block to be accessed
COL0	0000_0000h	Locates the byte to be accessed within a given page MS = 0
NAND_BC	0000_0000h	BC = 0 to read entire 2112-byte page
NAND_FSR	-	Returns with AS0 holding program status
NAND_FIR0, NAND_FIR1, NAND_FIR2	2411_592Ch, 49C00_000h, 00000_000h	OP0 = CMD0 = command 0; OP1 = CA0 = column address; OP2 = RA0 = page address; OP3 = WBCD = write BC bytes of data from buffer; OP4 = CMD2 = command 2; OP5 = NWAIT = program NCFGR[NUM_WAIT] for tWB time. OP6 = CMD1 = command 1; OP7 = RDSTAT = read erase status into NAND_FSR; OP8–OP14 = NOOP

In this sequence, use opcode (NWAIT, CMD1) instead of CW1 to issue a read status during the device-busy phase. This is different from the sequence NAND flash program command sequence example, where a read status is issued after the device becomes ready.

13.5.4.7 Valid opcode transitions in the IFC

This table describes the valid opcode transitions.

Ensure that the FIR sequence is programmed using only valid opcode transitions.

Table 13-255. Supported opcode transitions (in FIR)

Current opcode	Next opcode supported *n*=0-7, *m*=0-3								
CMD*n*	CA*m*/ RA*m*	CMD*n*	UA	RBCD/ BTRD/ SBRD/ RDSTAT/ RB_B	CW*n*	WFR	NWAIT	NOOP	
CA*m*	CA*m*/ RA*m*	CMD*n*	UA	RDSTAT/ RB	CW*n*	WFR	NWAIT	NOOP	

Table continues on the next page...

Table 13-255. Supported opcode transitions (in FIR) (continued)

Current opcode	Next opcode supported *n*=0-7, *m*=0-3								
RA*m*	RA*m*	CMD*n*	UA	WBCD	RDSTAT/ RB	CW*n*	WFR	NWAIT	NO OP
UA*m*	RA*m*	CMD*n*	UA	WBCD	RDSTAT/ RB	CW*n*	WFR	NWAIT	NO OP
WBCD	CMD*n*	CW*n*	WFR	NWAIT	NOOP				
CW*n*	CA*m*/ RA*m*	CMD*n*	UA	RBCD/ BTRD/ SBRD/ RDSTAT/ RB_B	CW*n*	WFR	NWAIT	NOOP	
WFR	CMD*n*	RBCD/ SBRD	NWAIT	NOOP	-	-	-	-	
NWAIT	CMD*n*	CA*m*/ RA*m*	UA	WBCD	RDSTAT/ RB	CW*n*	WFR	NOOP	
RBCD/ BTRD/ SBRD/ RDSTAT/ RB	RDSTAT	CMD*n*	CW*n*	CA*m*/ RA*m*	NWAIT	NOOP	-	-	

Chapter 14
I2C Modules

14.1 I²C Overview

This chapter describes the dual inter-integrated circuit (I^2C) bus modules implemented on this device and covers the following topics:

- Introduction to I^2C
- I^2C Signal Descriptions
- I2C Controller Memory Map
- I^2C Functional Description
- I^2C Initialization/Application Information

14.2 Introduction to I²C

This section presents the following topics:

- Definition: I^2C Module
- Advantages of the I^2C Bus
- I^2C Module Block Diagram
- I^2C Features Summary
- I^2C Modes of Operation
- Definition: I^2C Conditions

14.2.1 Definition: I²C Module

The I^2C module is a functional unit that provides a two-wire-serial data (SDA) and serial clock (SCL)-bidirectional serial bus that provides a simple efficient method of data exchange between this device and other devices, such as microcontrollers, EEPROMs, real-time clock devices, A/D converters, and LCDs.

14.2.2 Advantages of the I²C Bus

The two-wire I²C bus minimizes interconnections between devices. The synchronous, multiple-master I²C bus allows the connection of additional devices to the bus for expansion and system development. The bus includes collision detection and arbitration that prevent data corruption if two or more masters attempt to control the bus simultaneously.

14.2.3 I²C Module Block Diagram

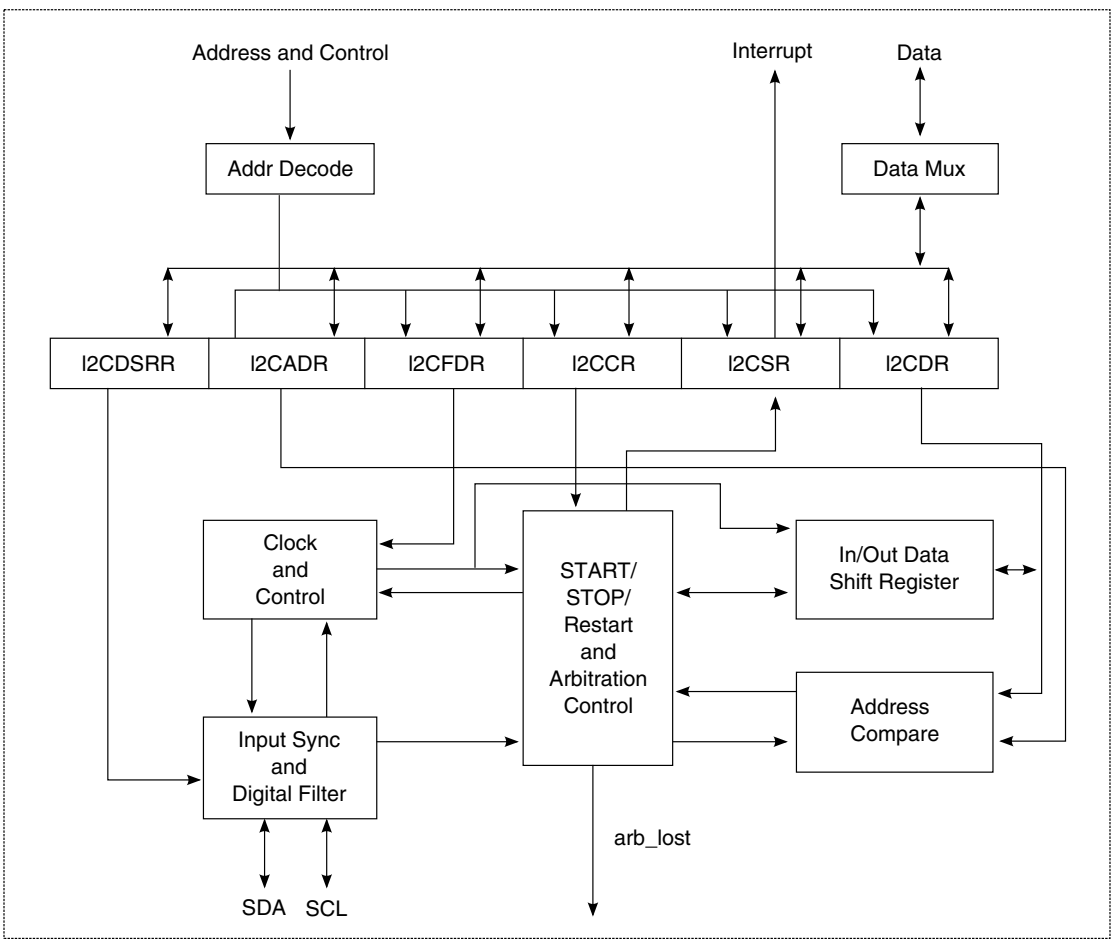


Figure 14-1. I²C Module Block Diagram

14.2.4 I²C Features Summary

The I²C module includes the following features:

- Acknowledge bit generation and detection
- Arbitration-lost interrupt with automatic mode switching from master to slave
- Bus busy detection
- Calling-address identification interrupt
- Multiple-master operation
- On-chip filtering for spikes on the bus
- Software-programmable clock frequency
- Software-selectable acknowledge bit
- START and STOP signal generation and detection
- Two-wire interface

14.2.5 I²C Modes of Operation

The I²C modules can operate in one of several modes, as shown in the following table.

Table 14-1. I²C Module Modes of Operation

Mode	Description	Important Notes
Master mode	The I²C module is the driver of the SDA line.	• Do not use the I²C module's slave address as a calling address. • The I²C module cannot be a master and a slave simultaneously.
Slave mode	The I²C module is not the driver of the SDA line.	• Enable the I²C module before a START condition from a non-I²C master is detected. • By default the I²C module performs as a slave receiver.
Interrupt-driven byte-to-byte data transfer mode	When successful slave addressing is achieved (and the I²C module deasserts SCL), the data transfer can proceed on a byte-to-byte basis in the direction specified by the R/W bit sent by the calling master.	Follow each byte of data by an acknowledge bit, which is signaled from the receiving device.

14.2.6 Definition: I²C Conditions

Table 14-2. I²C 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.
Repeated START	A START condition that is generated without a STOP condition to terminate the previous transfer.
STOP	A condition generated by the master to terminate a transfer and free the bus.

14.3 I²C Signal Descriptions

This section presents the following topics:

14.3.1 Signal Overview

The I²C 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 chip data sheet .

14.3.2 I²C Detailed Signal Descriptions

The SDA and SCL signals are described in the following table.

Table 14-3. I²C Module-Detailed Signal Descriptions

Signal	I/O	Description	
IIC*n*_SCL	I/O	Serial Clock. Performs as an input when the device is programmed as an I²C slave. SCL also performs as an output when the device is programmed as an I²C master. **Idle State**: High	
	O	When the I²C module is a master, it drives SCL along with SDA when transmitting. As a slave, the I²C module drives SCL low for data pacing. As output for the bidirectional serial clock, SCL operates as described below.	
		State Meaning	Asserted/Negated - SCL is driven along with SDA as the clock for the data.
	I	When the I²C module is idle or is acting as a slave, SCL is an input by default. The I²C module uses SCL to synchronize incoming data on SDA.The bus is assumed to be busy when SCL is detected low. As input for the bidirectional serial clock, SCL operates as described below.	

Table continues on the next page...

Table 14-3. I²C Module-Detailed Signal Descriptions (continued)

Signal	I/O	Description	
		State Meaning	Asserted/Negated - The I²C module uses the SCL signal to synchronize incoming data on SDA. The bus is assumed to be busy when SCL is detected low.
IIC*n*_SDA	I/O	Serial Data. Performs as an input when the device is in a receiving mode. SDA also performs as an output signal when the device is transmitting (either as an I²C master or slave). **Idle State**: High	
	O	When the I²C module is writing as a master or slave, it drives data on SDA synchronous to SCL. As output for the bidirectional serial data, SDA operates as described below.	
		State Meaning	Asserted/Negated - Data is driven.
	I	When the I²C module is idle or is in a receiving mode, SDA is an input by default. The unit receives data from other I²C devices on SDA. The bus is assumed to be busy when SDA is detected low. As input for the bidirectional serial data, SDA operates as described below.	
		State Meaning	Asserted/Negated - SDA is used to receive data from other devices. The bus is assumed to be busy when SDA is detected low.

14.4 I2C Controller Memory Map

The I²C module registers and their offsets are described in this section, as well as register access and register figure conventions. This section also contains important notes about the location, byte order, and read/write accesses of registers.

- Register Location: Locate all I²C registers in a cache-inhibited page.
- Byte Order: The I²C registers are shown in big-endian format. For a system that is in little-endian mode, the software must swap the bytes appropriately. Byte swapping is necessary because the I²C registers are byte registers.
- Read/Write Accesses: To guarantee in-order execution, execute a synchronizing assembly instruction after each I²C register read/write access.
- Writes to Reserved Fields: Reserved bits must be written with the value that they returned when read. Program the register by reading its value, modifying the appropriate fields, and writing back the value.

This section is organized as follows:

- I2C frequency divider register (I2C_I2CFDR)
- I2C control register (I2C_I2CCR)
- I2C status register (I2C_I2CSR)
- I2C data register (I2C_I2CDR)
- I2C digital filter sampling rate register (I2C_I2CDFSRR)

The following table lists the I^2C registers in offset order and provides links to the complete register descriptions.

I2C memory map

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
11_8000	I2C address register (I2C1_I2CADR)	8	R/W	00h	14.4.1/767
11_8004	I2C frequency divider register (I2C1_I2CFDR)	8	R/W	00h	14.4.2/767
11_8008	I2C control register (I2C1_I2CCR)	8	R/W	00h	14.4.3/769
11_800C	I2C status register (I2C1_I2CSR)	8	R/W	81h	14.4.4/771
11_8010	I2C data register (I2C1_I2CDR)	8	R/W	00h	14.4.5/772
11_8014	I2C digital filter sampling rate register (I2C1_I2CDFSRR)	8	R/W	20h	14.4.6/773
11_8100	I2C address register (I2C2_I2CADR)	8	R/W	00h	14.4.1/767
11_8104	I2C frequency divider register (I2C2_I2CFDR)	8	R/W	00h	14.4.2/767
11_8108	I2C control register (I2C2_I2CCR)	8	R/W	00h	14.4.3/769
11_810C	I2C status register (I2C2_I2CSR)	8	R/W	81h	14.4.4/771
11_8110	I2C data register (I2C2_I2CDR)	8	R/W	00h	14.4.5/772
11_8114	I2C digital filter sampling rate register (I2C2_I2CDFSRR)	8	R/W	20h	14.4.6/773
11_9000	I2C address register (I2C3_I2CADR)	8	R/W	00h	14.4.1/767
11_9004	I2C frequency divider register (I2C3_I2CFDR)	8	R/W	00h	14.4.2/767
11_9008	I2C control register (I2C3_I2CCR)	8	R/W	00h	14.4.3/769
11_900C	I2C status register (I2C3_I2CSR)	8	R/W	81h	14.4.4/771
11_9010	I2C data register (I2C3_I2CDR)	8	R/W	00h	14.4.5/772
11_9014	I2C digital filter sampling rate register (I2C3_I2CDFSRR)	8	R/W	20h	14.4.6/773
11_9100	I2C address register (I2C4_I2CADR)	8	R/W	00h	14.4.1/767
11_9104	I2C frequency divider register (I2C4_I2CFDR)	8	R/W	00h	14.4.2/767
11_9108	I2C control register (I2C4_I2CCR)	8	R/W	00h	14.4.3/769
11_910C	I2C status register (I2C4_I2CSR)	8	R/W	81h	14.4.4/771
11_9110	I2C data register (I2C4_I2CDR)	8	R/W	00h	14.4.5/772
11_9114	I2C digital filter sampling rate register (I2C4_I2CDFSRR)	8	R/W	20h	14.4.6/773

14.4.1 I2C address register (I2Cx_I2CADR)

The I^2C address register (I2CADR) specifies the address to which the I^2C module responds if the I^2C is addressed as a slave. This is not the address sent on the bus during the address-calling cycle when the I^2C module is in master mode.

Address: Base address + 0h offset

Bit	0	1	2	3	4	5	6	7
Read Write	ADDR							Reserved
Reset	0	0	0	0	0	0	0	0

I2Cx_I2CADR field descriptions

Field	Description
0–6 ADDR	Slave address. The ADDR field specifies the slave address to which the I^2C module responds if it is addressed as a slave. If the I^2C is in slave mode and a transaction's calling address matches I2CADR[ADDR], the module will set I2CSR[MIF], signaling a pending interrupt. (For more information about I2CSR[MIF], see I2C status register (I2C_I2CSR).)
7 -	This field is reserved. Reserved

14.4.2 I2C frequency divider register (I2Cx_I2CFDR)

The I^2C frequency divider register (I2CFDR), shown in the following figure, specifies the ratio used to prescale the clock for selecting a specific bit rate.

For additional guidance about the proper use of I2CFDR and I2CDFSRR on Power Architecture™ integrated host/communications processors, refer to the application note AN2919, *Determining the I^2C Frequency Divider Ratio for SCL.*

NOTE

Writing to the Reserved Field: If you write to the reserved field, always write back the field's original value, as described in the note in the Programmable Registers section.

The I2CFDR fields and FDR field settings for clock divider values are listed in the following table.

Address: Base address + 4h offset

Bit	0	1	2	3	4	5	6	7
Read Write	Reserved		FDR					
Reset	0	0	0	0	0	0	0	0

I2Cx_I2CFDR field descriptions

Field	Description
0–1 -	This field is reserved. Reserved
2–7 FDR	Frequency divider ratio. Specifies the ratio used to prescale the clock for bit rate selection. The serial bit clock frequency of the SCL is equal to the platform clock divided by the designated divider. You can change the frequency divider value at any point in a program. The serial bit clock frequency divider selections are described in the following list: **FDR Divider (Decimal)** 0x00 384 0x01 416 0x02 480 0x03 576 0x04 640 0x05 704 0x06 832 0x07 1024 0x08 1152 0x09 1280 0x0A 1536 0x0B 1920 0x0C 2304 0x0D 2560 0x0E 3072 0x0F 3840 0x10 4608 0x11 5120 0x12 6144 0x13 7680 0x14 9216 0x15 10240 0x16 12288 0x17 15360 0x18 18432 0x19 20480 0x1A 24576 0x1B 30720 0x1C 36864 0x1D 40960 0x1E 49152 0x1F 61440 0x20 256 0x21 288 0x22 320 0x23 352 0x24 384 0x25 448 0x26 512

Table continues on the next page...

I2Cx_I2CFDR field descriptions (continued)

Field	Description
	0x27 576 0x28 640 0x29 768 0x2A 896 0x2B 1024 0x2C 1280 0x2D 1536 0x2E 1792 0x2F 2048 0x30 2560 0x31 3072 0x32 3584 0x33 4096 0x34 5120 0x35 6144 0x36 7168 0x37 8192 0x38 10240 0x39 12288 0x3A 14336 0x3B 16384 0x3C 20480 0x3D 24576 0x3E 28672 0x3F 32768

14.4.3 I2C control register (I2Cx_I2CCR)

The I^{2}C control register (I2CCR) contains fields for controlling several actions, modes, messages, conditions, and capabilities.

NOTE

Writing to the Reserved Field: If you write to the reserved field, always write back the field's original value, as described in the note in the Programmable Registers section.

.

Address: Base address + 8h offset

Bit	0	1	2	3	4	5	6	7
Read	MEN	MIEN	MSTA	MTX	TXAK		Reserved	BCST
Write						RSTA		
Reset	0	0	0	0	0	0	0	0

I2Cx_I2CCR field descriptions

Field	Description
0 MEN	Module enable. Controls the software reset of the I^2C module. 0 The module is reset and disabled. The module is held in reset, but the registers can still be accessed. 1 The I^2C module is enabled. MEN must be set before any other control register fields have any effect. All I^2C registers for slave receive or master START can be initialized before setting this field, however.
1 MIEN	Module interrupt enable. Enables the interrupt to be reported to the interrupt controller and/or (ultimately) the CPU. 0 Interrupt reporting from the I^2C module is disabled. If any pending interrupt conditions exist, they are not cleared. 1 Interrupt reporting from the I^2C module is enabled. If an interrupt condition is detected and I2CSR[MIF] is also set, the interrupt is reported.(For more information about I2CSR fields, see I2C status register (I2C_I2CSR).)
2 MSTA	Master/Slave Mode START. 0 If MSTA is changed from 1 to 0, a STOP condition is generated and the I^2C mode changes from master to slave. MSTA is cleared without generating a STOP condition when the master loses arbitration. 1 If MSTA is changed from 0 to 1, a START condition is generated on the bus and master mode is selected for the I^2C.
3 MTX	Transmit/Receive Mode Select. Specifies the direction of master and slave transfers. If the I^2C module is configured as a slave, the software should set MTX to match I2CSR[SRW]. If the I^2C module is in master mode, MTX should be set according to the type of transfer required. For address cycles, therefore, this field is always high (has a value of 1). MTX is cleared if the master loses arbitration. 0 Receive mode is selected. 1 Transmit mode is selected.
4 TXAK	Transfer Acknowledge. Specifies the value driven onto the SDA line during acknowledge cycles for both master and slave receivers. The TXAK value applies only if the I^2C module is configured as a receiver, not as a transmitter. The TXAK setting does not apply to address cycles. When the device is addressed as a slave, an acknowledge is always sent. 0 An acknowledge signal (low value on the SDA) is sent out to the bus at the ninth clock after receiving 1 byte of data. 1 No acknowledge signal (high value on the SDA) is sent.
5 RSTA	Repeated START. Specifies whether to generate a repeated START condition. Setting RSTA always generates a repeated START condition on the bus and provides the device with the current bus master. The RSTA field is not readable; an attempt to read RSTA returns a 0. 0 No START condition is generated. 1 A repeated START condition is generated.
6 -	This field is reserved. Reserved
7 BCST	Broadcast. 0 The broadcast accept capability is disabled. 1 The I^2C is enabled to accept broadcast messages at address 0.

14.4.4 I2C status register (I2Cx_I2CSR)

The I^2C status register (I2CSR) is read-only with the exception of the MIF and MAL fields, which can be cleared by software.

Address: Base address + Ch offset

Bit	0	1	2	3	4	5	6	7
Read	MCF	MAAS	MBB	MAL	BCSTM	SRW	MIF	RXAK
Write								
Reset	1	0	0	0	0	0	0	1

I2Cx_I2CSR field descriptions

Field	Description
0 MCF	Data Transfer. When one byte of data is transferred, MCF is cleared. MCF is set by the falling edge of the ninth clock of a byte transfer. 0 A byte transfer is in progress. MCF is cleared when I2CDR is read in receive mode or written in transmit mode. (For more information about I2CDR, see I2C data register (I2C_I2CDR).) 1 The byte transfer is completed.
1 MAAS	Addressed as a slave. MAAS is set if the module is acting as a slave and has detected that the I2CADR address matches with the transaction's calling address. The processor is interrupted if I2CCR[MIEN] is set. Next, the processor must check the SRW value and set I2CCR[MTX] accordingly. Any write to I2CCR automatically clears MAAS. For more information, see I2C address register (I2C_I2CADR) (I2CADR fields) and I2C control register (I2C_I2CCR) (I2CCR fields). 0 Not addressed as a slave. 1 Addressed as a slave.
2 MBB	Bus Busy. Indicates the status of the bus. When a START condition is detected, MBB is set. If a STOP condition is detected, it is cleared. 0 The I^2C bus is idle. 1 The I^2C bus is busy.
3 MAL	Arbitration Lost. MAL is automatically set (value of 1) if arbitration is lost. Note that the device does not automatically retry a failed transfer attempt. 0 Arbitration is not lost. Can only be cleared by software. 1 Arbitration is lost.
4 BCSTM	Broadcast Match. The broadcast address is always all zeros. BCSTM can be set only if I2CCR[BCST] is set to enable it. (For more information about I2CCR[BCST], see I2C control register (I2C_I2CCR).) 0 There has not been a broadcast match. 1 The calling address matches with the broadcast address instead of the programmed slave address. BCSTM is also set if the I^2C drives an address of all zeros and broadcast mode is enabled.
5 SRW	Slave Read/Write. If MAAS is set, SRW indicates the value of the R/W command bit of the calling address, which is sent from the master. By checking SRW, the processor can select slave transmit/receive mode according to the master's command.

Table continues on the next page...

I2Cx_I2CSR field descriptions (continued)

Field	Description
	0 Slave receive mode is selected, with the master writing to the slave. 1 Slave transmit mode is selected, with the master reading from the slave. SRW is valid only if both of the following conditions are met: • A complete transfer occurred and no other transfers have been initiated. • The I^2C module is configured as a slave and has an address match.
6 MIF	Module Interrupt. MIF is set if an interrupt is detected. An interrupt is reported if I2CCR[MIEN] is set. The interrupts for I^2C1 and I^2C2 are combined into one interrupt, which is sourced by the dual I^2C module 1; similarly, the interrupts for I^2C3 and I^2C4 are combined into one interrupt, which is sourced by the dual I^2C module 2.(For more information about I2CCR fields, see I2C control register (I2C_I2CCR). 0 No interrupt is detected. MIF can be cleared only by software. 1 An interrupt is detected. MIF is set when one of the following events occurs: • One byte of data is transferred (set at the falling edge of the ninth clock). • The I2CADR value matches with the calling address in Slave Receive mode. • Arbitration is lost.
7 RXAK	Received Acknowledge. The value of SDA during the reception of a bus cycle's acknowledge bit. If RXAK = 0, it indicates that an acknowledge signal has been received after the 8 bits of data have been transmitted on the bus. If RXAK = 1, it means no acknowledge signal has been detected at the ninth clock. 0 An acknowledge signal has been received. 1 No acknowledge signal has been received.

14.4.5 I2C data register (I2Cx_I2CDR)

I2CDR specifies the calling address and data to be transmitted (if the I^2C is in master or slave transmit mode) or allows the I^2C module to receive the next byte of data on the I^2C module (if the I^2C is in master or slave receive mode).

Address: Base address + 10h offset

Bit	0	1	2	3	4	5	6	7
Read Write	DATA							
Reset	0	0	0	0	0	0	0	0

I2Cx_I2CDR field descriptions

Field	Description
0–7 DATA	Specifies data to be transmitted or allows receipt of the next data byte on the I^2C module, depending on the I^2C mode: • Transmit mode: Data transmission is initiated when data is written to I2CDR. For master transmit mode, the first byte of data written to I2CDR is used for the address transfer and is follows the format described in Transactions. When bytes are written to I2CDR in transmit mode, they cannot be verified by reading them back. • Receive mode: Reading I2CDR allows the I^2C module to receive the next byte of data on the I^2C interface (in addition to reading the contents of I2CDR). In all cases, the most significant bit is sent first.

I2Cx_I2CDR field descriptions (continued)

Field	Description
	I2CCR[MTX] must be set appropriately for the desired behavior. For example, if I2CCR[MTX] is set for transmit mode (1) instead of receive mode (0), reading I2CDR does not initiate receipt of the next data byte. (For more information about I2CCR[MTX], see I2C control register (I2C_I2CCR).) For both master receive and slave receive modes, the very first read is always a dummy read.

14.4.6 I2C digital filter sampling rate register (I2Cx_I2CDFSRR)

The digital filter sampling rate register (I2CDFSRR) specifies the sample rate for filtering out signal noise.

For additional guidance about the proper use of I2CFDR and I2CDFSRR on Power Architecture™ integrated host/communications processors, refer to the application note AN2919, *Determining the I^2C Frequency Divider Ratio for SCL.*

NOTE

Writing to the Reserved Field: If you write to the reserved field, always write back the field's original value, as described in the note in the Programmable Registers section.

Address: Base address + 14h offset

Bit	0	1	2	3	4	5	6	7
Read Write	Reserved		DFSR					
Reset	0	0	1	0	0	0	0	0

I2Cx_I2CDFSRR field descriptions

Field	Description
0–1 -	This field is reserved. Reserved
2–7 DFSR	Digital Filter Sampling Rate. Specifies the sample rate for the program to use in filtering out signal noise. This rate is used to prescale the frequency the digital filter uses to take samples from the I^2C bus. The resulting sampling rate is calculated by dividing the platform frequency by the non-zero value of the DFSR. If I2CDFSRR = 0, the I^2C bus sample points to the reset divisor.

14.5 I^2C Functional Description

This section presents the following topics:

- Notes About Module Operation
- Transactions
- Protocol Implementation Details

- Bus Arbitration
- Clock Behavior
- Filtering of SCL and SDA Lines

14.5.1 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.

14.5.2 Transactions

This section covers the following topics:

- Protocol Overview
- Definitions
- I^2C Calling Address Requirements
- High-Level Protocol Steps
- START Condition
- Slave Address Transmission
- General Call (Broadcast) Addressing
- Data Transmission
- STOP Condition
- Repeated START Condition

14.5.2.1 Protocol Overview

The following figure shows the behavior of SCL and SDA during a typical I^2C transaction.

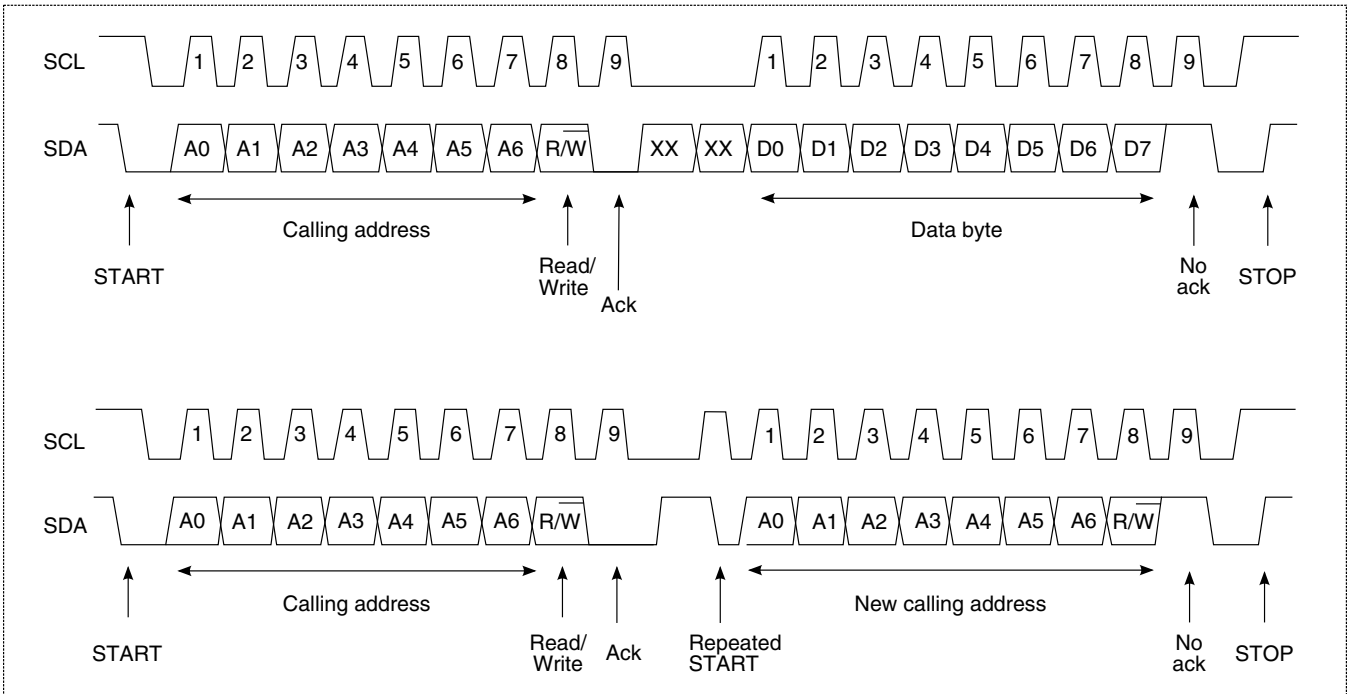


Figure 14-32. I²C Transaction Protocol

14.5.2.2 Definitions

This section defines several important terms presented in Figure 14-32.

Table 14-34. I²C Definitions

Term	Definition
START	A START condition, as defined in Definition: I²C Conditions
STOP	A STOP condition, as defined in 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 I²C Calling Address Requirements.
Read/write (R/W_B)	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")
Acknowledge	A bit that specifies the acknowledgement of a calling address, indicated by pulling SDA low.

14.5.2.3 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.

14.5.2.4 High-Level Protocol Steps

The I^2C protocol conceptually supports two types of transfers, which are illustrated in Figure 14-32. The significant steps in these transfers are presented in the following table. Details of each of these steps are presented in subsequent sections.

Table 14-35. 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. Data transfer	3. Data transfer
4. STOP condition	4. Repeated START condition
5. (repeat Steps 1-4)	5. (repeat Steps 2-4 as needed)
	6. STOP condition.
	7. (repeat Steps 1-7)

14.5.2.5 START Condition

A master on the I^2C bus initiates a data transfer by sending a START condition on the I^2C bus when the bus is not engaged (both SDA and SCL are high).

On this device, the START condition is sent by setting I2CCR[MSTA]. (For more information about I2CCR[MSTA], see I2C control register (I2C_I2CCR).)

14.5.2.6 Slave Address Transmission

The master transmits the slave address immediately after the START condition (see START Condition). The process of slave address transmission is presented in Table 14-36.

Table 14-36. Slave Address Transmission Process

Step	Action
1	The master transmits the seven-bit slave address.
2	The master transmits the R/W_B bit.

Table continues on the next page...

Table 14-36. Slave Address Transmission Process (continued)

Step	Action
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 generate a STOP condition or a repeated START condition. • The acknowledge bit is cleared: The master must wait for SCL to return to logic zero.

14.5.2.7 General Call (Broadcast) Addressing

The master may also initiate a general call (broadcast) command by transmitting a slave address of 0x00. In this case, the second byte of the broadcast message is the master address. This device will not check the R/W_B bit in a broadcast address. Because the second byte is automatically acknowledged by hardware, the receiver device software must verify that the broadcast message is intended for itself by reading the second byte of the message. If the master address is for another receiver device and the third byte is a write command, software can ignore the third byte during the broadcast. If the master address is for another receiver device and the third byte is a read command, software must write 0xFF to I2CDR with I2CCR[TXAK] = 1, so that it does not interfere with the data written from the addressed device.

This device responds to a general call (broadcast) command if I2CCR[BCST] is set.

14.5.2.8 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_B 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.

14.5.2.9 STOP Condition

A master on the I^2C bus can terminate a data transfer by sending a STOP condition. It can do so even if the slave has sent an acknowledge bit. In this case, the slave must release the I^2C bus.

On this device, the STOP condition is sent by clearing I2CCR[MSTA]. (For more information about I2CCR[MSTA], see I2C control register (I2C_I2CCR).)

A master is not required to send a STOP condition at the end of every transfer. For more information, see Repeated START Condition.

14.5.2.10 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 14-32.

14.5.3 Protocol Implementation Details

This section provides details of how aspects of the I^2C protocol are implemented in this device. The following sections are included:

- Transaction Monitoring
- Control Transfer

14.5.3.1 Transaction Monitoring

The different conditions of the I^2C data transactions (see Transactions) are monitored as follows:

- START conditions are detected when SDA falls while SCL is high.
- STOP conditions are detected when SDA rises while SCL is high.
- Data transfers in progress are canceled when a STOP condition is detected or if there is a slave address mismatch. Cancellation of data transactions resets the clock module.
- The bus is detected to be busy upon the detection of a START condition, and idle upon the detection of a STOP condition.

14.5.3.2 Control Transfer

The I^2C module contains logic that controls the output to the serial data (SDA) and serial clock (SCL) lines of the I^2C. The SCL output is driven low as determined by the internal clock generated in the clock module. The SDA output can only change at the midpoint of a low cycle of the SCL, unless it is performing a START, STOP, or repeated START condition. Otherwise, the SDA output is held constant.

Table 14-37 presents the behavior of the SCL and SDA signals under the various protocol conditions.

Table 14-37. SDA and SCL Signal Behavior

Mode	Conditions When SDA Is Driven Low	Conditions When SCL Corresponds to the Internal SCL Signal
Master	Data bit (transmission)	Bus owner
	Acknowledge bit (receive)	Lost arbitration
	START condition	START condition
	STOP condition	STOP condition
	Repeated START condition	Repeated START condition begin
		Repeated START condition end
Slave	Acknowledging address match	Address cycle
	Data bit (transmit)	Transmit cycle
	Acknowledge bit (receive)	Acknowledge cycle

14.5.4 Bus Arbitration

This section covers the following topics related to bus arbitration:

- Bus Arbitration Overview
- Loss of Arbitration
- Module Startup During a Data Transfer

14.5.4.1 Bus Arbitration Overview

If two or more masters simultaneously try to control the bus, each master's clock synchronization procedure (including the I^2C module) determines the bus clock. The low part of the period is equal to the longest clock low period and the high part of the period is equal to the shortest one among the masters.

14.5.4.2 Loss of Arbitration

A bus master loses arbitration if it transmits a logic 1 on SDA while another master transmits a logic 0. In this case, the losing master performs the following tasks:

1. Switches to slave-receive mode.
2. Stops driving the SDA line without generating a STOP condition.
3. Sets I2CSR[MAL] to indicate the loss of arbitration.
4. Services the transaction if it is directed to itself.

14.5.4.3 Module Startup During a Data Transfer

Table 14-38 presents the behavior of the I^2C module if it is enabled in the middle of an ongoing byte transfer.

Table 14-38. Module Behavior at Startup During a Data Transfer

Mode	Behavior
Slave	The I^2C module ignores the current transfer on the bus and starts operating whenever a subsequent START condition is detected.
Master	The I^2C module cannot tell whether the bus is busy; therefore, if a START condition is initiated, the current bus cycle can be corrupted. This ultimately results in the current bus master of the I^2C bus losing arbitration, after which bus operations return to normal.

14.5.5 Clock Behavior

This section covers the following topics of SCL synchronization:

- SCL Synchronization

- Clock Stretching
- Handshaking

14.5.5.1 SCL 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. 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.

14.5.5.2 Clock Stretching

Slaves can use the clock synchronization mechanism to slow down the transfer bit rate. After the master has driven the SCL line 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.

14.5.5.3 Handshaking

The clock synchronization mechanism can be used as a handshake in data transfer. Slave devices can hold SCL low after completion of a 1-byte transfer (9 bits). In such cases, the mechanism halts the bus clock and forces the master clock into wait states until the slave releases the SCL line.

14.5.6 Filtering of SCL and SDA Lines

This section covers the following topics:

- Filtering of SCL and SDA Lines-Overview
- Sample Rate Control

14.5.6.1 Filtering of SCL and SDA Lines-Overview

The SCL and SDA inputs are filtered to eliminate noise. Three consecutive samples of the SCL and SDA lines are compared to a pre-determined sampling rate. If they are all high, the output of the filter is high. If they are all low, the output is low. If they are any combination of highs and lows, the output is whatever the value of the line was in the previous clock cycle.

14.5.6.2 Sample Rate Control

The sampling rate is controlled by the value stored in I2CDFSRR (as described in I2C digital filter sampling rate register (I2C_I2CDFSRR)). The duration of the sampling cycle is controlled by a down counter. This allows a software write to I2CDFSRR to control the filtered sampling rate.

14.6 I^{2}C Initialization/Application Information

This section discusses the following topics related to I^{2}C initialization and application:

- 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

14.6.1 Recommended Interrupt Service Flow

Figure 14-33 shows a flowchart for the recommended I^{2}C interrupt service routine. Deviation from the flowchart may result in unpredictable I^{2}C bus behavior. Although the flowchart does not explicitly show the synchronization after every I2C register access, it is recommended that a synchronizing instruction follow each I^{2}C register read or write to guarantee in-order instruction execution.

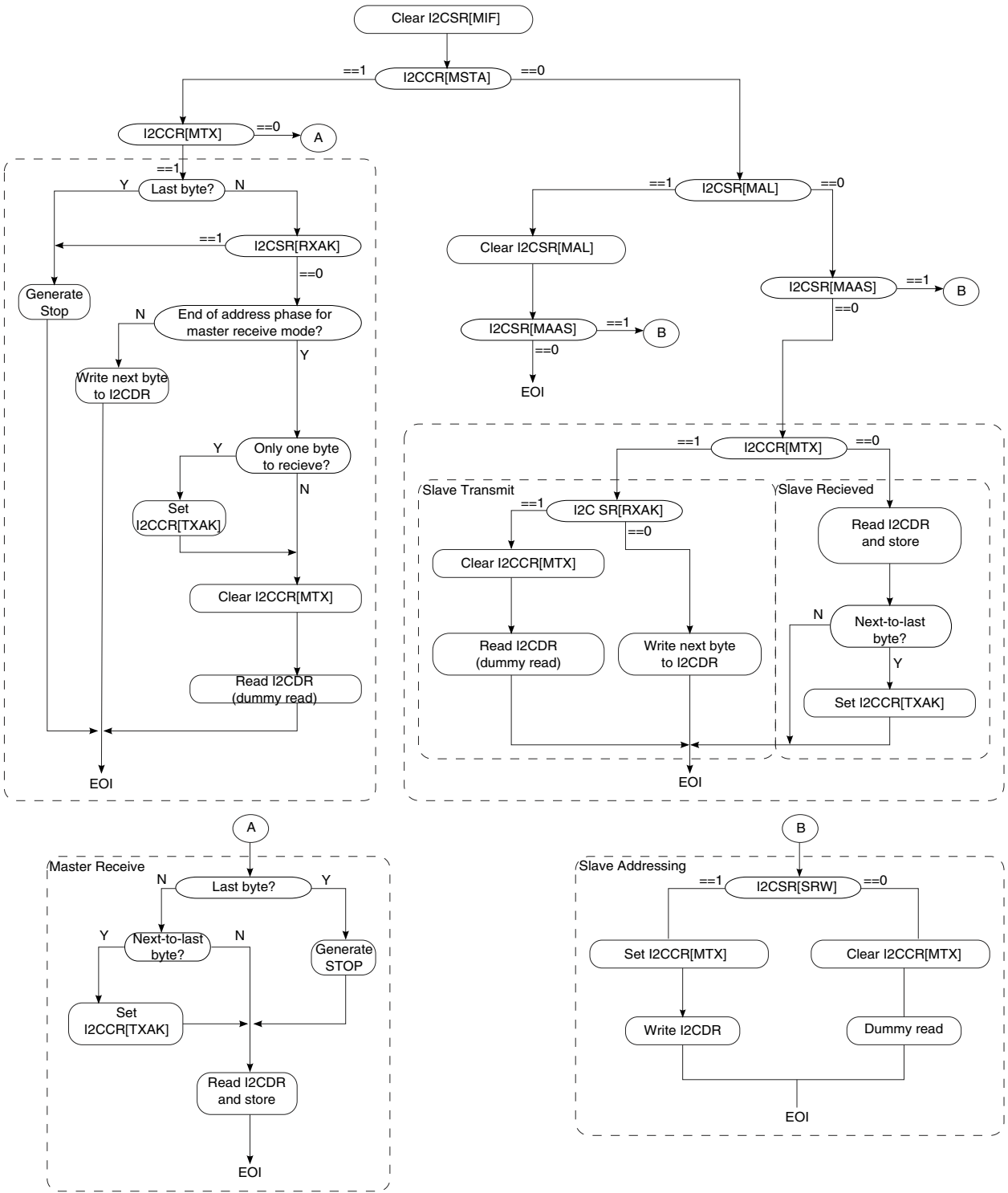


Figure 14-33. Recommended I²C Interrupt Service Routine Flowchart

14.6.2 General Programming Guidelines (for Both Master and Slave Mode)

This section provides programming guidelines recommended for the I^2C module in both master and slave mode. It describes the following procedures and processes:

- Initializing the Module
- Software Response After a Transfer
- Generating SCL when SDA is Low

NOTE

Illegal or irregular bus activity: The I^2C module does not guarantee its recovery from all illegal I^2C bus activity. Additionally, a malfunctioning device may hold the bus captive. A good programming practice is for software to rely on a watchdog timer to help recover from I^2C bus hangs. The recovery routine should also handle the case when the status bits returned after an interrupt are not consistent with what was expected due to illegal I^2C bus protocol behavior.

14.6.2.1 Initializing the Module

The following sequence initializes the I^2C unit:

1. Program I2CFDR[FDR] with the appropriate ratio for the desired SCL frequency. (For more information about the I2CFDR[FDR], see I2C frequency divider register (I2C_I2CFDR).)
2. Update I2CADR to define the slave address for this device. (For more information about I2CADR fields, see I2C address register (I2C_I2CADR).)
3. Modify I2CCR to select master/slave mode, transmit/receive mode, and interrupt-enable or disable; set I2CCR[MEN] to enable the I^2C module. (For more information about I2CCR fields, see I2C control register (I2C_I2CCR).)

14.6.2.2 Software Response After a Transfer

Transmission or reception of a byte automatically sets the data transferring bit (I2CSR[MCF]), which indicates that one byte has been transferred. If the interrupt function is enabled during the initialization sequence (I2CCR[MIEN] is set), the I^2C interrupt bit (I2CSR[MIF]) is also set and an interrupt is generated to the PIC. In the interrupt handler, software must take the following steps:

1. Clear I2CSR[MIF].
2. Read the contents of the I^2C data register (I2CDR) in receive mode or write to I2CDR in transmit mode. Note that this access to I2CDR causes I2CSR[MCF] to be cleared automatically. See Recommended Interrupt Service Flow.

Note the programming guidelines for the following conditions:

- An interrupt at the end of the address cycle-When an interrupt occurs at the end of the address cycle, the master remains in transmit mode. If master receive mode is required, I2CCR[MTX] must be toggled at this stage. See Figure 14-33.
- Monitoring I2CSR[MIF] when the interrupt function is disabled-If the interrupt function is disabled, software can service the I2CDR in the main program by monitoring I2CSR[MIF]. In this case, I2CSR[MIF] must be polled rather than I2CSR[MCF] because MCF behaves differently when arbitration is lost. Note that interrupt or other bus conditions may be detected before the I^2C signals have time to settle. Thus, when polling I2CSR[MIF] (or any other I2CSR bits), software delays may be needed in order to give the I^2C signals sufficient time to settle.
- Slave-mode addressing-During slave-mode addressing, when I2CSR[MAAS] is set, I2CSR[SRW] should be read to determine the direction of the subsequent transfer, and I2CCR[MTX] should be programmed accordingly.
- Slave-mode data transfers-For slave-mode data transfers (MAAS is cleared), I2CSR[SRW] is not valid, and I2CCR[MTX] must be read to determine the direction of the current transfer. See Figure 14-33 for more details.

14.6.2.3 Generating SCL when SDA is Low

When a system reset does not cause all I^2C devices to be reset, it is sometimes necessary to force the I^2C module to become the I^2C bus master out of reset and drive SCL (even though SDA may already be driven, which indicates the bus is busy). Thus, SDA can be driven low by another I^2C device while this I^2C module is coming out of reset and will stay low indefinitely.

The following procedure can be used to force this I^2C module to generate SCL so that the device driving SDA can finish its transaction:

1. Disable the I^2C module (I2CCR[MEN] = 0) and change to master mode (I2CCR[MSTA] = 1) by programming the value 0x20 into I2CCR.
2. Re-enable the I^2C module (I2CCR[MEN] = 1) by programming the value 0xA0 into I2CCR.
3. Read I2CDR.
4. Return the I^2C module to slave mode (I2CCR[MSTA] = 0) by programming the value 0x80 into I2CCR.

14.6.3 Programming Guidelines Specific to Master Mode

This section includes the following programming guidelines specific to master mode:

- Generating START
- Generating STOP
- Generating Repeated START
- Loss of Arbitration and Forcing Slave Mode

14.6.3.1 Generating START

After initialization, the following sequence can be used to generate START:

1. If the device is connected to a multimaster I^2C system, test the state of I2CSR[MBB] to check whether the serial bus is available (I2CSR[MBB] = 0) before switching to master mode.
2. Select master mode (set I2CCR[MSTA]) to transmit serial data and select transmit mode (set I2CCR[MTX]) for the address cycle.
3. Write the slave address being called into I2CDR. The data written to I2CDR[0-6] comprises the slave calling address. I2CCR[MTX] indicates the direction of transfer (transmit/receive) required from the slave.

NOTE

The scenario above assumes that the I^2C interrupt bit (I2CSR[MIF]) is cleared. If MIF is set at any time, an I^2C interrupt is generated (if interrupt reporting is enabled with I2CCR[MIEN] =1).

NOTE

The interrupts for I^2C1 and I^2C2 are combined into a single interrupt to the PIC; similarly, interrupts for I^2C3 and I^2C4 are combined into a single interrupt to the PIC.

14.6.3.2 Generating STOP

A data transfer ends with a STOP condition generated by the master device.

A master transmitter generates a STOP condition after all the data has been transmitted. When the I^2C module is acting as the master transmitter, and all the data has been transmitted (that is, after last byte to be transmitted has been written to I2CDR), software clears I2CCR[MSTA] to generate a STOP condition.

When the I^2C module is acting as the master receiver, the master indicates the termination of the transfer by not acknowledging the final byte and by generating a STOP condition. For a multiple byte transfer before reading the next-to-last byte in I2CDR, software sets I2CCR[TXAK], then software reads the next-to-last byte in I2CDR, which causes the master receiver to not acknowledge the next transfer and to automatically generate a STOP at the conclusion of the next transfer. For single-byte transfers, at the conclusion of the address phase, software should set I2CCR[TXAK] and clear I2CCR[MTX], and then perform a dummy read to I2CDR. Prior to subsequent I^2C transactions, software should clear I2CCR[TXAK]. This can be performed when setting up the I2CCR for the next transfer.

14.6.3.3 Generating Repeated START

At the end of a data transfer, if the master wants to continue communicating on the bus, it can generate another START condition followed by another slave address without first generating a STOP condition. This is accomplished by setting I2CCR[RSTA].

14.6.3.4 Loss of Arbitration and Forcing Slave Mode

When a master loses arbitration, the following conditions all occur:

- I2CSR[MAL] is set.
- I2CCR[MSTA] is cleared (changing the master to slave mode).
- An interrupt occurs (if enabled) at the falling edge of the ninth clock of this transfer.

Thus, the slave interrupt service routine should first test I2CSR[MAL] and software should clear it if it is set.

14.6.4 Programming Guidelines Specific to Slave Mode

This section includes the following programming guidelines specific to slave mode:

- Slave Mode Interrupt Service Routine
- Acknowledge Receipt During Transmission

14.6.4.1 Slave Mode Interrupt Service Routine

In the slave interrupt service routine, the slave should be tested as follows to determine whether a calling of its own address has been received:

If I2CSR[MAAS] is set, software should set the transmit/receive mode select bit (I2CCR[MTX]) according to the R/W command bit (I2CSR[SRW]). Writing to I2CCR clears MAAS automatically. MAAS is read as set only in the interrupt handler at the end of that address cycle in which an address match occurred; interrupts resulting from subsequent data transfers clear MAAS.

A data transfer can then be initiated by writing to I2CDR for slave transmits or dummy reading from I2CDR in slave-receive mode. The slave drives SCL low between byte transfers. SCL is released when the I2CDR is accessed in the required mode. See Figure 14-33.

14.6.4.2 Acknowledge Receipt During Transmission

In the slave transmitter routine, the received acknowledge bit (I2CSR[RXAK]) must be checked before the next byte of data is sent:

- If I2CSR[RXAK] is cleared, the master has acknowledged the previous data transfer and the next byte of data may be transmitted.
- If I2CSR[RXAK] is set, the master is signaling an end-of-data by not acknowledging the previous data transfer. Software running on the slave transmitter must do the following:
  a. Clear I2CCR[MTX] to switch the slave from transmitter to receiver mode.
  b. Perform a dummy read of I2CDR, which releases SCL so the master can generate a STOP condition.

In the slave receiver routine, the receiver transmit acknowledge bit (I2CCR[TXAK]) must be set after the next-to-last byte of data from I2CDR is read.

See Recommended Interrupt Service Flow.

Chapter 15
Enhanced Serial Peripheral Interface

15.1 Introduction

The enhanced serial peripheral interface (eSPI) allows the device to exchange data with peripheral devices such as EEPROMs, real-time clocks, A/D converters, and ISDN devices.

The eSPI is a full-duplex, synchronous, character-oriented channel that supports a simple interface (receive, transmit, clock and chip selects). The eSPI consists of transmitter and receiver sections, an independent baud-rate generator, and a control unit. The transmitter and receiver sections use the same clock, which is derived from the eSPI baud rate generator in master mode. During an eSPI transfer, data is sent and received simultaneously.

The eSPI receiver and transmitter each have a FIFO of 32 bytes, as shown below. When the eSPI is disabled in the eSPI mode register (SPMODE[EN] = 0), it consumes little power.

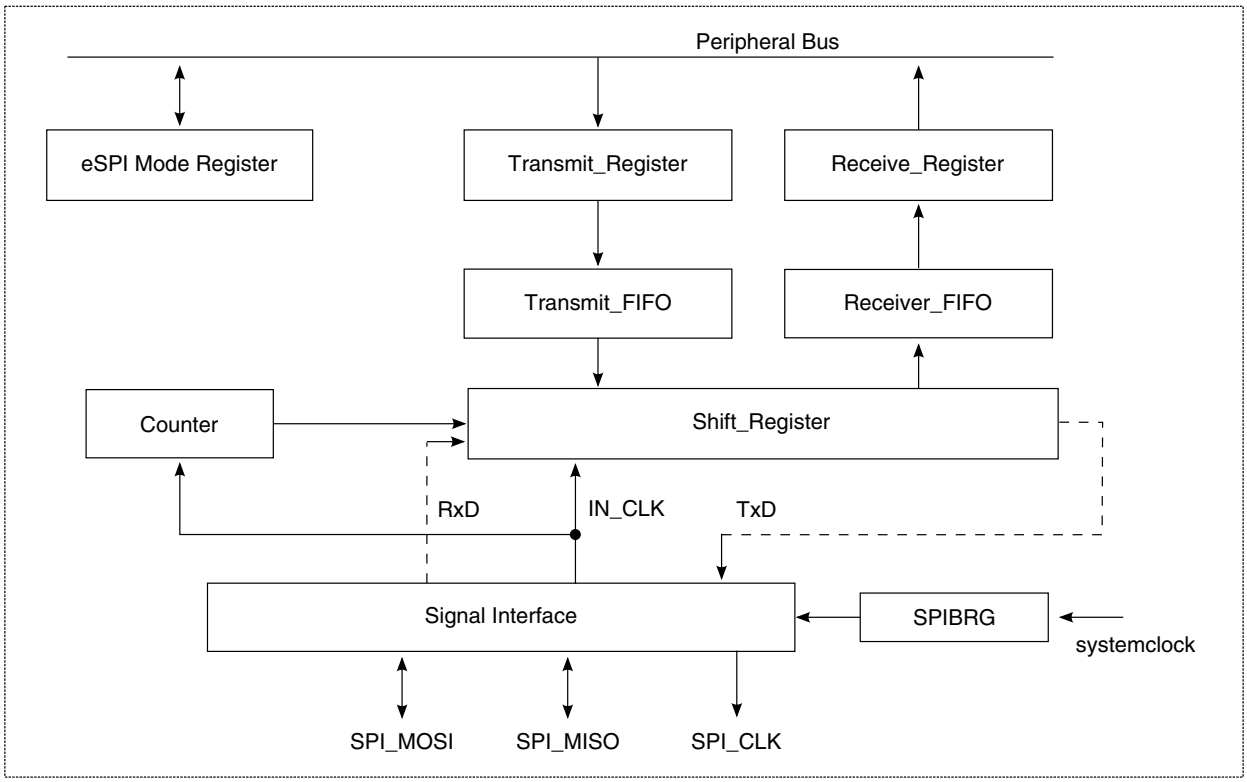


Figure 15-1. eSPI block diagram

15.1.1 Features

The major features of the eSPI are listed as follows:

- Interface contains SPI_MOSI, SPI_MISO, SPI_CLK, and 4 chip selects
- Supports eSPI master
- Supports RapidS^{TM} full clock cycle operation
- Full-duplex or half-duplex master operation
- Supports Winbond dual output read
- Command in transaction level-easier for accessing eSPI devices
- Works with a range from 4-bit to 16-bit data characters
- Supports back-to-back character transmission and reception
- Supports reverse data mode for 8/16 bits data characters
- Supports single-master environment
- Maximum clock rate possible is (platform clock rate/2)
- Independent programmable baud rate generator
- Programmable clock phase and polarity.
- Supports four different configurations per chip select
- Local loopback capability for testing

15.1.2 eSPI transmission and reception process

As the eSPI is a character-oriented communication unit, the core is responsible for packing and unpacking the receive and transmit frames.

A frame consists of all of the characters transmitted or received during a completed eSPI transmission session, from the first character written to the eSPI transmit FIFO access register (SPITF) register to the last character transmitted with the total number as indicated in the command written to the eSPI command register (ESPI_SPCOM) register.

The core receives data by reading the eSPI receive FIFO access register (SPIRF) when the RNE ("not empty") bit in the eSPI event register (SPIE) is set.

The core transmits data by writing it into the SPITF register. After the core writes the final character to SPITF it waits for DON bit in the SPIE register to be set indicating frame was fully transmitted. It might then write a new command to SPCOM register.

The eSPI sets the TNF ("not full") bit in SPIE register whenever its transmit FIFO is not full.

The eSPI core handshake protocol can be implemented by using a polling or interrupt mechanism. When using a polling mechanism, the core reads the SPIE in a predefined frequency and acts according to the value of the SPIE bits. The polling frequency depends on the eSPI serial channel frequency. When using the interrupt mechanism, setting either the TNF (not full) or RNE (not empty) bits of the SPIE causes an interrupt to the core. The core then reads the SPIE and acts appropriately.

15.1.3 Modes of operation

The eSPI can be programmed to work in a single master environment. This section describes eSPI master operation in a single-master configuration.

In master mode, the eSPI sends a message to the slave peripheral, which sends back a simultaneous reply. A single master with multiple slaves uses up to four chip select signals to selectively enable slaves, as shown below.

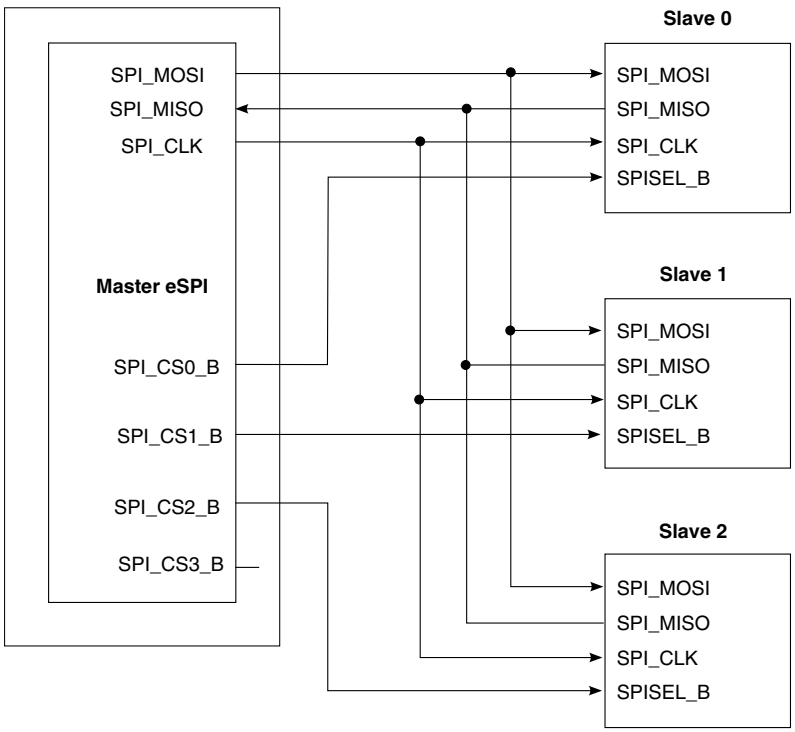


Figure 15-2. Single-master/multi-slave configuration

To start exchanging data, the core writes the data to be sent into the SPITF register. The eSPI then generates programmable clock pulses on SPI_CLK for each character. It shifts Tx data out on the "eSPI master-out slave-in" (SPI_MOSI) and Rx data in on the eSPI "master-in slave-out" (SPI_MISO) simultaneously. During the transmission process the core is responsible for supplying the data whenever the eSPI requests it to ensure smooth operation.

The maximum sustained data rate that the eSPI supports depends on the software latency. However, the eSPI can transfer a single character at very high rates— a maximum (up to system clock / 2) specified by the device data sheet (where system clock is defined as platform clock divided by 2). Gaps might be inserted between multiple frames.

15.2 External signal descriptions

The eSPI interface consists of transmit, receive, clock and chip selects.

The following table provides an overview of eSPI signal properties.

Table 15-1. Signal properties

Name	Function
SPI_MISO	Master input slave output
SPI_MOSI	Master output slave input or second master input slave output for Winbond dual output read.
SPI_CLK	Output serial clock connected to the other SPI_CLK.
SPI_CS_B [0:3]	eSPI slave select outputs

15.2.1 eSPI detailed signal descriptions

Table 15-2. ESPI detailed signal descriptions

Signal	I/O	Description	
SPI_MISO	I	Master input slave output	
		State meaning	Asserted-The data that has been received from the eSPI is high Negated-The data that has been received from the eSPI is low
		Timing	Assertion-According to the SPI_CLK assertion/negation/in the middle of phase (depends on the SPMODEx configuration register) Negation-According to the SPI_CLK assertion/negation/in the middle of phase (depends on the SPMODEx configuration register)
SPI_MOSI	I/O	Master output slave input or second master input slave output for Winbond dual output read	
		State meaning	Asserted-The data that has been transmitted from/to the eSPI is high Negated-The data that has been transmitted from/to the eSPI is low
		Timing	Assertion-According to the SPI_CLK assertion/negation/in the middle of phase (depends on the SPMODEx configuration register). Negation-According to the SPI_CLK assertion/negation/in the middle of phase (depends on the SPMODEx configuration register).
SPI_CLK	O	Serial clock out	
		State meaning	Assertion/negation-According to SPMODEx[PM,DIV16] register rate configuration
		Timing	Assertion/negation-During frame reception/transmission
SPI_CS_B [0:3]	O	eSPI slave select outputs	
		State meaning	Asserted- Slave 0, 1, 2, 3 is selected and master controls transmission/reception Negated-idle state
		Timing	Assertion-A predefined time before frame starts, during frame transmission/reception, a predefined time after frame ends Negation-When master is idle or controls another slave

The eSPI can be configured as a master in single master environment. The master eSPI generates the transfer clock SPI_CLK using the eSPI baud rate generator (BRG). The eSPI BRG takes as its input the platform clock divided by two.

SPI_CLK is a gated clock, active only during data transfers. Four combinations of SPI_CLK phase and polarity can be configured with the clock invert SPMODEx[CIx] and clock phase SPMODEx[CPx] register bits.

The eSPI master-in slave-out SPI_MISO signal acts as an input for master devices and as an output for slave devices. Conversely, the master-out slave-in SPI_MOSI signal is an output for master devices and an input for slave devices. However, it also acts as a second input for master devices and as a second output for slave devices when using Winbond dual output read.

SPI_CLK is the clock output signal that shifts received data in from SPI_MISO and transmitted data out to SPI_MOSI. eSPI masters must output a slave select signal to enable eSPI slave devices.

15.3 Enhanced serial peripheral interface (eSPI) memory map

The table below shows the memory mapped registers of the eSPI and their offsets. It lists the offset, name, and a cross-reference to the complete description of each register. Note that the full register address comprises CCSRBAR together with the SPI block base address and offset listed in the table below.

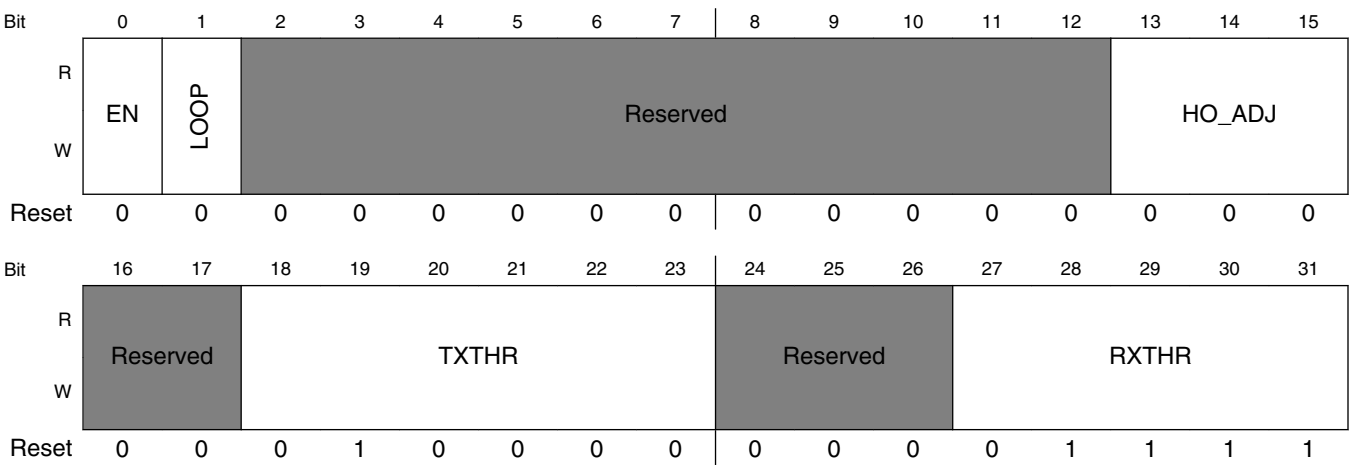
ESPI memory map

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
11_0000	eSPI mode register (ESPI_SPMODE)	32	R/W	0000_100Fh	15.3.1/795
11_0004	eSPI event register (ESPI_SPIE)	32	R/W	0020_0000h	15.3.2/797
11_0008	eSPI mask register (ESPI_SPIM)	32	R/W	0000_0000h	15.3.3/799
11_000C	eSPI command register (ESPI_SPCOM)	32	W	0000_0000h	15.3.4/801
11_0010	eSPI transmit FIFO access register (ESPI_SPITF)	32	W	0000_0000h	15.3.5/803
11_0014	eSPI receive FIFO access register (ESPI_SPIRF)	32	R	0000_0000h	15.3.6/805
11_0020	eSPI CS0 mode register (ESPI_SPMODE0)	32	R/W	0010_0000h	15.3.7/806
11_0024	eSPI CS1 mode register (ESPI_SPMODE1)	32	R/W	0010_0000h	15.3.8/808
11_0028	eSPI CS2 mode register (ESPI_SPMODE2)	32	R/W	0010_0000h	15.3.9/809
11_002C	eSPI CS3 mode register (ESPI_SPMODE3)	32	R/W	0010_0000h	15.3.10/ 811

15.3.1 eSPI mode register (ESPI_SPMODE)

The eSPI mode register (SPMODE) controls eSPI general operation mode.

Address: 11_0000h base + 0h offset = 11_0000h



ESPI_SPMODE field descriptions

Field	Description
0 EN	Enable eSPI. Any bits in SPMODE must not change when EN is set. 0 The eSPI is disabled. The eSPI is in a idle state and consumes minimal power. The eSPI BRG is not functioning and the input clock is disabled. 1 The eSPI is enabled.
1 LOOP	Loop mode. Enables local loopback operation. 0 Normal operation. 1 Loopback mode. Used to test the eSPI controller internal functionality, the transmitter output is internally connected to the receiver input. The receiver and transmitter operate normally, except that received data is ignored.
2–12 -	This field is reserved. Reserved
13–15 HO_ADJ	Data output hold adjustment. This field can be used for RapidS. 000 Output data is delayed by an extra 3 platform clock cycles 001 Output data is delayed by an extra 1 platform clock cycles 010 Output data is delayed by an extra 2 platform clock cycles 011 Output data is delayed by an extra 3 platform clock cycles 100 Output data is delayed by an extra 4 platform clock cycles 101 Output data is delayed by an extra 5 platform clock cycles 110 Output data is delayed by an extra 6 platform clock cycles 111 Output data is delayed by an extra 7 platform clock cycles

Table continues on the next page...

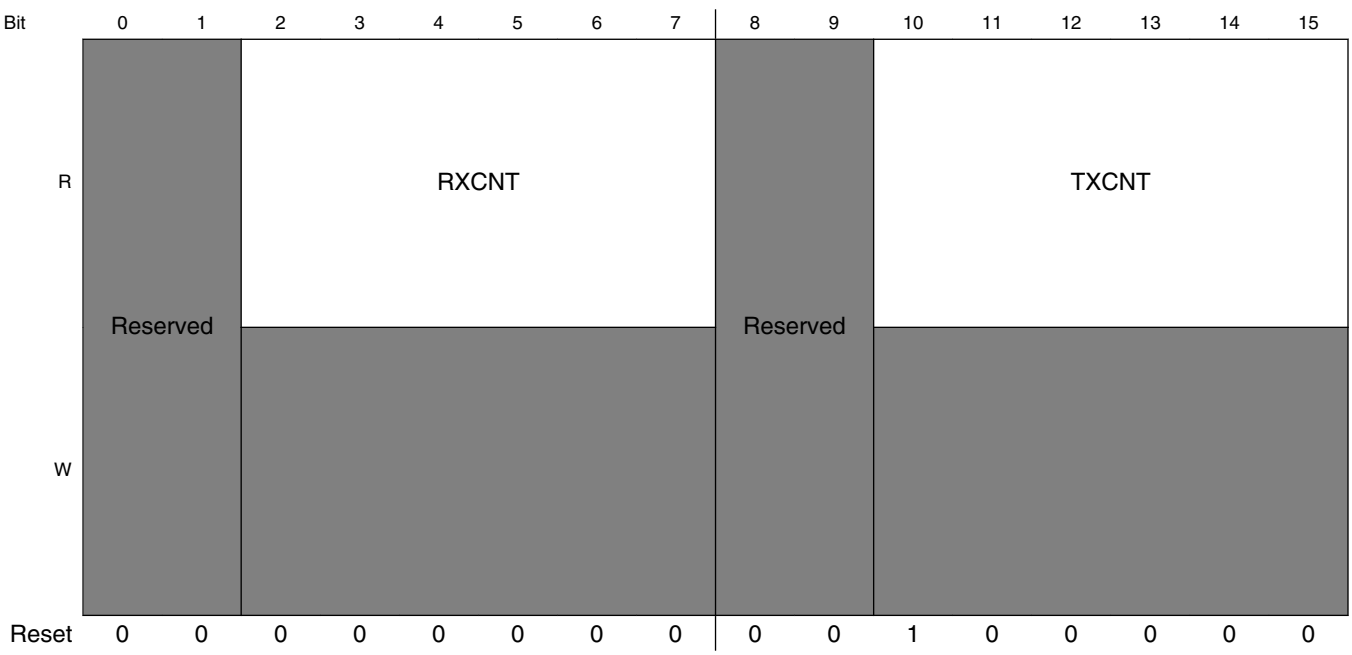
ESPI_SPMODE field descriptions (continued)

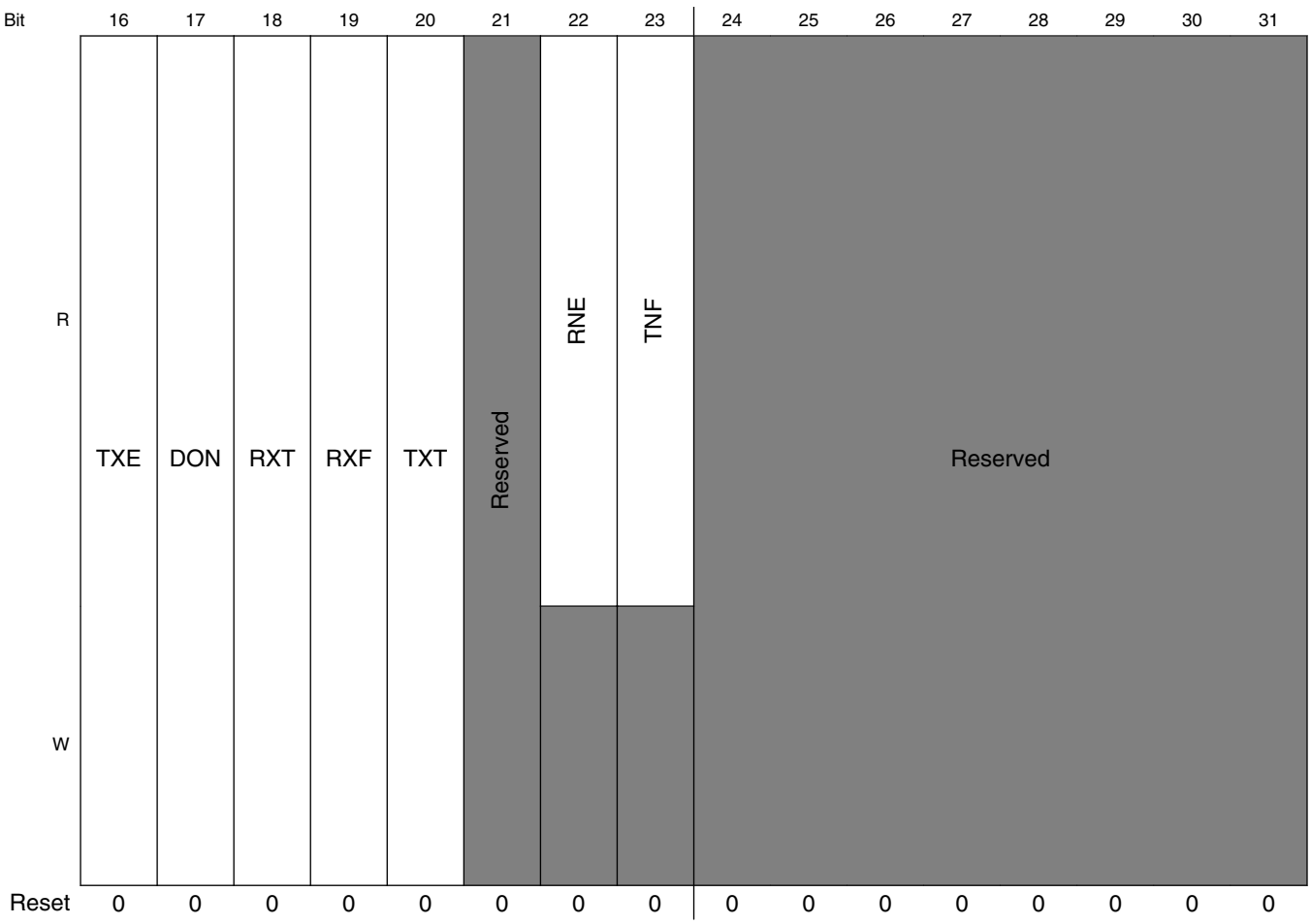
Field	Description
16–17 -	This field is reserved. Reserved
18–23 TXTHR	Tx FIFO threshold-if Tx FIFO has less than TXTHR bytes, an interrupt can be issued to the core. Valid values: 1-32
24–26 -	This field is reserved. Reserved
27–31 RXTHR	Rx FIFO threshold-if Rx FIFO has more than RXTHR bytes, an interrupt can be issued to the core. Valid values: 0-31

15.3.2 eSPI event register (ESPI_SPIE)

The eSPI event register (SPIE) generates interrupts and reports events recognized by the eSPI. When an event is recognized, the eSPI sets the corresponding SPIE bit. Clear SPIE bits by writing a 1-writing 0 has no effect. Setting a bit in the eSPI mask register (SPIM) enables interrupt and clearing a bit masks the corresponding interrupt. Unmasked SPIE bits must be cleared before the core clears internal interrupt requests. Bits RNE and TNF are status bits. Fields RXCNT and TXCNT hold Rx and Tx FIFOS' statuses. They are not cleared as a result of writing to SPIE.

Address: 11_0000h base + 4h offset = 11_0004h





ESPI_SPIE field descriptions

Field	Description
0–1 -	This field is reserved. Reserved, should be cleared.
2–7 RXCNT	The current number of full Rx FIFO bytes **NOTE:** For character lengths of 9 to 16 bits-each character occupies 2 bytes in Rx/Tx FIFO.
8–9 -	This field is reserved. Reserved, should be cleared.
10–15 TXCNT	The current number of free Tx FIFO bytes **NOTE:** For character lengths of 9 to 16 bits-each character occupies 2 bytes in Rx/Tx FIFO
16 TXE	Tx FIFO is empty
17 DON	Last character was transmitted. The last character was transmitted and a new command can be written for the next frame
18 RXT	Rx FIFO has more than RXTHR bytes, that is, at least RXTHR + 1 bytes
19 RXF	Rx FIFO is full

Table continues on the next page...

ESPI_SPIE field descriptions (continued)

Field	Description
20 TXT	Tx FIFO has less than TXTHR bytes, that is, at most TXTHR - 1 bytes
21 -	This field is reserved. Reserved, should be cleared.
22 RNE	Not empty. Indicates that the Rx FIFO register contains a received character. 0 The Rx FIFO is empty 1 The Rx FIFO has a received character. The core can read the content of Rx FIFO through SPIRF.
23 TNF	Tx FIFO not full. 0 The transmitter FIFO is full. 1 The transmitter FIFO is not full.
24–31 -	This field is reserved. Reserved, should be cleared.

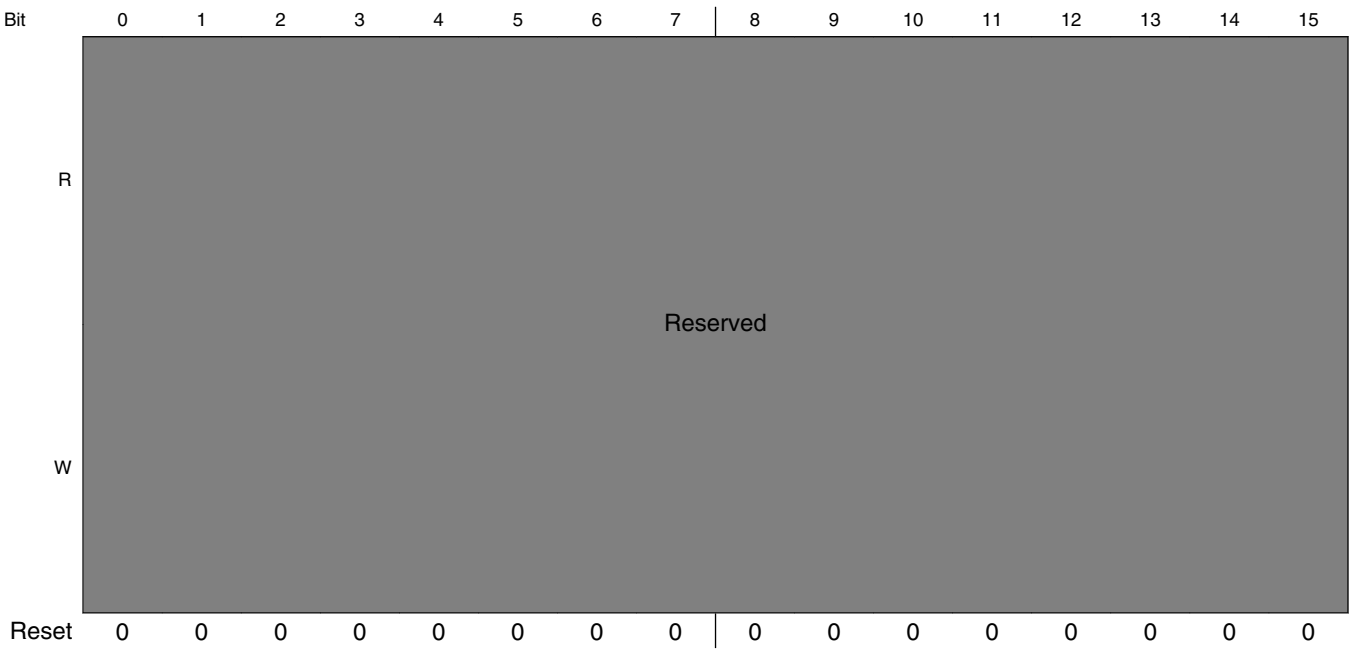
15.3.3 eSPI mask register (ESPI_SPIM)

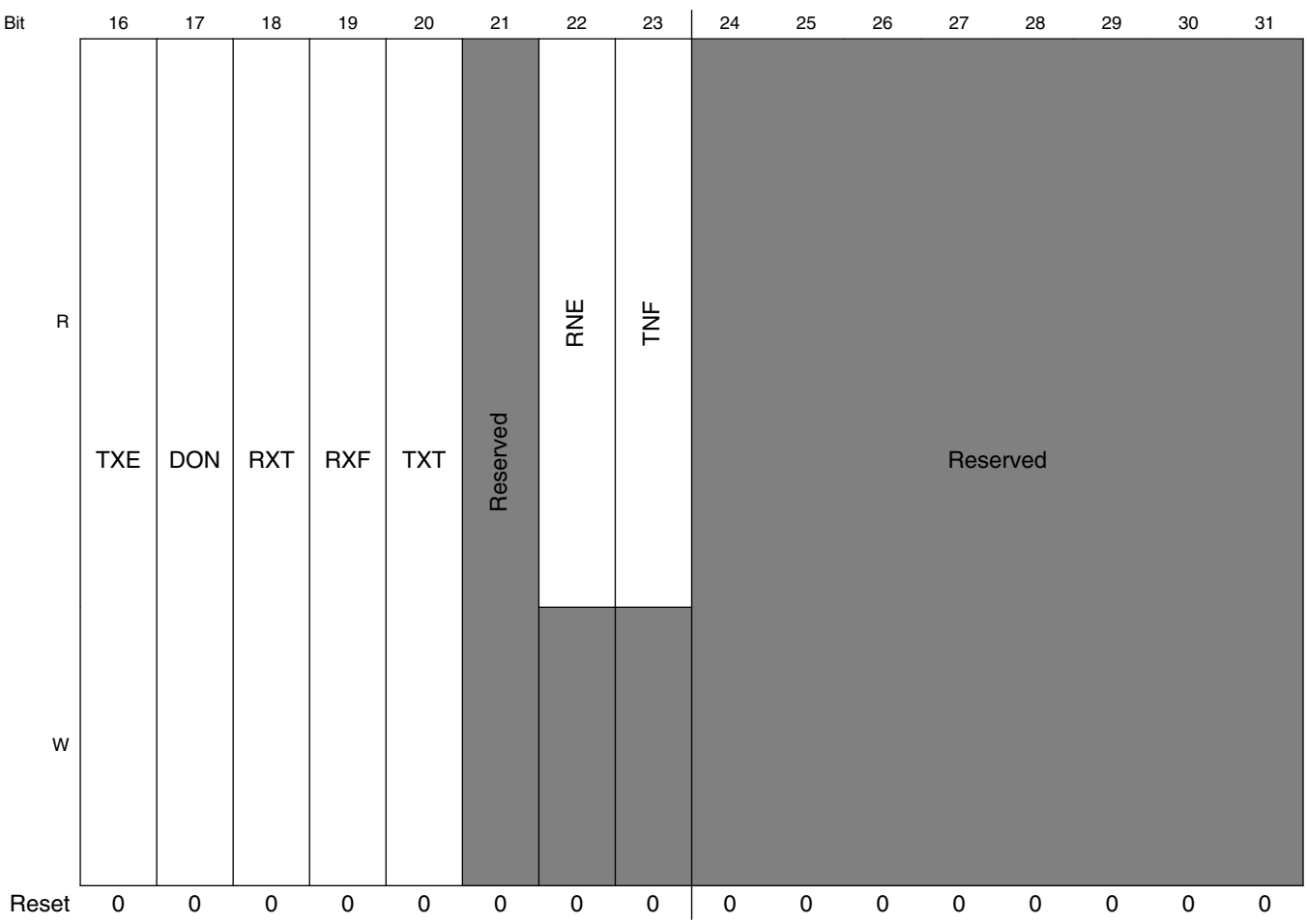
The eSPI mask register (SPIM) enables/masks interrupts for events recognized by the eSPI. When an event is recognized, the eSPI sets the corresponding SPIE bit. Setting a bit in the eSPI mask register (SPIM) enables and clearing a bit masks the corresponding interrupt. Unmasked SPIE bits must be cleared before the core clears internal interrupt requests.

Bits RNE and TNF in SPIM are status bits. They are not cleared as a result of writing to SPIM.

Enhanced serial peripheral interface (eSPI) memory map

Address: 11_0000h base + 8h offset = 11_0008h





ESPI_SPIM field descriptions

Field	Description
0–15 -	This field is reserved. Reserved, should be cleared.
16 TXE	Tx FIFO empty interrupt mask 0 TXE event will not cause eSPI Interrupt 1 TXE event will cause eSPI Interrupt
17 DON	Last character transmitted mask 0 DON event will not cause eSPI Interrupt 1 DON event will cause eSPI Interrupt
18 RXT	Rx threshold interrupt mask 0 RXT event will not cause eSPI Interrupt 1 RXT event will cause eSPI Interrupt
19 RXF	Rx FIFO full interrupt mask 0 RXF event will not cause eSPI Interrupt 1 RXF event will cause eSPI Interrupt
20 TXT	Tx threshold interrupt mask 0 TXT event will not cause eSPI Interrupt 1 TXT event will cause eSPI Interrupt
21 -	This field is reserved. Reserved, should be cleared.
22 RNE	Rx not empty interrupt mask 0 Not Empty event will not cause eSPI Interrupt 1 Not Empty event will cause eSPI Interrupt
23 TNF	Tx not full interrupt mask 0 Not full event will not cause eSPI Interrupt 1 Not full event will cause eSPI Interrupt
24–31 -	This field is reserved. Reserved, should be cleared.

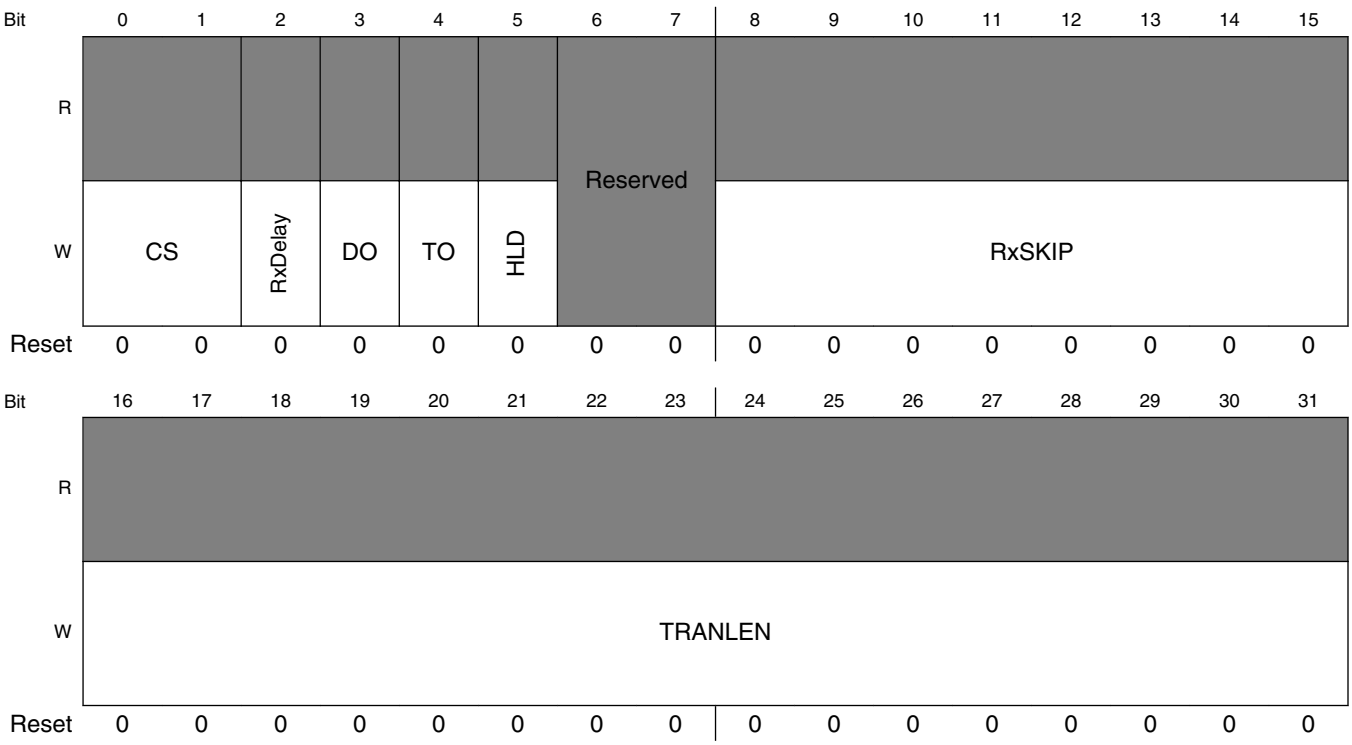
15.3.4 eSPI command register (ESPI_SPCOM)

The eSPI command register (SPCOM) is used by the host to supply information on the new frame.

After SPCOM has been written to initiate the first transaction after startup, commands can be executed only after SPIE[DON] is set. Otherwise they are ignored.

A transaction can be full duplex (regular eSPI) or half duplex. Half duplex can be used for example for write accesses to a flash (only transmit) or for a read access from a flash (first part is transmit without receive, while the second part is receive without transmit).

Address: 11_0000h base + Ch offset = 11_000Ch



ESPI_SPCOM field descriptions

Field	Description
0–1 CS	Chip select-chip select for which transaction is destined 00 SPI_CS0_B 01 SPI_CS1_B 10 SPI_CS2_B 11 SPI_CS3_B
2 RxDelay	RxDelay 0 Normal eSPI operation 1 Rx data should be sampled a bit later than regular eSPI (used for full cycle operation such as with Atmel RapidS devices)
3 DO	This mode is useful only for character lengths of 4,6,8. DO and RapidS should not be set simultaneously. 0 Normal eSPI operation 1 Winbond dual output read-when eSPI master reads data 2 data bits are available (on MISO and MOSI)
4 TO	Transmit only 1 No reception is done for the frame (useful for write transactions) 0 Normal operation
5 HLD	HLD 0 Normal operation 1 Mask first generated SPI_CLK. Should be used only for RapidS mode0

Table continues on the next page...

ESPI_SPCOM field descriptions (continued)

Field	Description
6–7 -	This field is reserved. Reserved, should be cleared.
8–15 RxSKIP	If (RXSKIP ≠ 0)-Number of characters skipped for reception from frame start. Non-zero values of RxSKIP force the eSPI to half-duplex mode, and therefore this causes TRANLEN-RxSKIP characters to be skipped for transmission. RXSKIP is useful for reads of SPI Flash memories where the first valid read data is received several characters after the transmission begins (after the eSPI has transmitted an instruction opcode and address). **NOTE:** If TO = 1, RxSKIP must be set to 0. **NOTE:** If RXSKIP = 0 and TO = 0, the eSPI changes to full duplex mode.
16–31 TRANLEN	Transaction length - (number of characters in the frame - 1)

15.3.5 eSPI transmit FIFO access register (ESPI_SPITF)

The 32-bit write-only eSPI transmit FIFO access register (SPITF) holds the characters to be written to the transmit FIFO. The number of bits in each character is specified by SPMODEx[LENx]. Each time SPIE[TNF] is set, the core can write more data to the SPITF register, if there is no error indication in the SPIE.

For character lengths of 4 to 8 bits, SPITF contains up to 4 characters (unless end of frame). The lsbs are in bits 7, 15, 23, and 31 of SPITF.

For character lengths of 9 to 16 bits, SPITF contains up to 2 characters (unless end of frame). For 16 bits with SPMODEx[REVx] = 1, the lsb is in bits 15 and 31 of SPITF. For other options, lsbs are in bits 7 and 23 while msbs are in bits (23-LENx) and (39-LENx) of SPITF.

For example: REV = 0, LEN = 10 (0xA), SPITF[0-15] = 0xFB05-bitstream is: (lsb first) 11011111101 (msb last).

NOTE

The user must write N bytes of SPITF ($1 \leq N \leq 4$) that do not exceed the number of free bytes in the transmit FIFO. It is valid for the user to write only 1 or 2 bytes of SPITF (at offset 0x010) if the user wishes to write fewer characters than the maximum supported by SPITF for the particular character length in use.

The following figures show examples of the contents of SPITF with various parameters set.

SPITF Example - SPMODEx[REVx]=0, SPMODEx[LENx]=3, LSB Sent First:

	0–3	4–5	6–7	8–11	12–13	14–15	16–19	20–21	22–23	24–27	28–29	30–31
R												
W	-	MSB 0	Data 0	LSB0	-	MSB 1	Data 1	LSB1	-	MSB 2	Data 2	LSB2	-	MSB 3	Data 3	LSB3

SPITF Example - SPMODEx[REVx]=x, SPMODEx[LENx]=7:

	0	1–6	7	8	9–14	15	16	17–22	23	24–30	31
R											
W	MSB0	Data 0	LSB 0	MSB1	Data 1	LSB 1	MSB2	Data 2	LSB 2	MSB3	Data 3	LSB 3

SPITF Example - SPMODEx[REVx]=0, SPMODEx[LENx]=10, LSB Sent First:

	0–6	7	8–12	13	14–15	16–22	23	24–28	29	30–31
R										
W	Data 0 LS Byte	LSB0	-	MSB 0	Data 0 MS Byte	Data 1 LS Byte	LSB1	-	MSB 1	Data 1 MS Byte

SPITF Example - SPMODEx[REVx]=1, SPMODEx[LENx]=15, MSB Sent First:

	0	1–7	8–14	15	16	17–23	24–30	31
R								
W	MSB0	Data 0 MS Byte	Data 0 LS Byte	LS B0	MSB1	Data 1 MS Byte	Data 1 LS Byte	LS B1

Address: 11_0000h base + 10h offset = 11_0010h

Bit	0–15	16–31
R		
W	DATA	
Reset	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

ESPI_SPITF field descriptions

Field	Description
0–31 DATA	Varies as parameters set

15.3.6 eSPI receive FIFO access register (ESPI_SPIRF)

The 32-bit read-only eSPI receive data register (SPIRF) is used to hold characters read from the receive FIFO. Each time SPIE[RNE] is set, the core can read the SPIRF register.

For character lengths of 4 to 8 bits, SPIRF contains up to 4 characters. The msbs are in bits 0, 8, 16, and 24. For character lengths of 9 to 16 bits, SPIRF contains up to 2 characters. The msbs are in bits 0 and 16. SPMODEx[REVx] does not affect the msb or lsb bit positions when reading the SPIRF register.

The user must read N bytes of SPIRF ($1 \le N \le 4$) that do not exceed the amount of data in the receive FIFO. The user can read less bytes than the amount of data in the receive FIFO. For example, a 1-byte read of SPIRF when configured for 8-bit characters with 4 characters of data in the receive FIFO results in the 3 unread characters shuffling down to the lower 24 bits of SPIRF in preparation for the following SPIRF read.

SPIRF Example - SPMODEx[LENx]=3

	0	1 2	3	4 7	8	9 10	11	12 15	16	17 18	19	20 23	24	25 26	27	28 31
R	MSB0	Data 0	LS B0	-	MSB1	Data 1	LSB 1	-	MSB2	Data 2	LSB 2	-	MSB3	Data 3	LSB 3	-
W																

SPIRF Example - SPMODEx[LENx]=10:

	0	1 7	8 9	10	11 15	16	17 23	24 25	26	27 31
R	MSB 0	Data 0 MS Byte	Data 0 LS Byte	LSB0	-	MSB 1	Data 1 MS Byte	Data 1 LS Byte	LSB1	-
W										

SPIRF Example - SPMODEx[LENx]=15:

	0	1 7	8 14	15	16	17 23	24 30	31
R	MSB0	Data 0 MS Byte	Data 0 LS Byte	LSB 0	MSB1	Data 1 MS Byte	Data 1 LS Byte	LSB 1
W								

Address: 11_0000h base + 14h offset = 11_0014h

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	DATA																															
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

ESPI_SPIRF field descriptions

Field	Description
0–31 DATA	Varies as parameters set

15.3.7 eSPI CS0 mode register (ESPI_SPMODE0)

The eSPI CS0 mode register (SPMODE0) controls eSPI master operation with chip select 0.

Address: 11_0000h base + 20h offset = 11_0020h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CI0	CP0	REV0	DIV160	PM0				ODD0	Reserved		POL0	LEN0			
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CS0BEF				CS0AFT				CS0CG					Reserved		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ESPI_SPMODE0 field descriptions

Field	Description
0 CI0	Clock invert. Inverts eSPI clock polarity. See Figure 15-13 and Figure 15-14 for more information 0 The inactive state of SPI_CLK is low. 1 The inactive state of SPI_CLK is high.
1 CP0	Clock phase. Selects the transfer format. See Figure 15-13 and Figure 15-14 for more information. 0 SPI_CLK starts toggling at the middle of the data transfer. 1 SPI_CLK starts toggling at the beginning of the data transfer.
2 REV0	Reverse data mode. Determines the receive and transmit character bit order. 0 lsb of the character sent and received first 1 msb of the character sent and received first-for 8/16 bits data character only
3 DIV160	Divide by 16. Selects the clock source for the eSPI baud rate generator (eSPI BRG) when configured as an eSPI master. **NOTE:** System clock as used here is defined to be platform clock divided by 2.

Table continues on the next page...

ESPI_SPMODE0 field descriptions (continued)

Field	Description
	0 System clock is the input to the eSPI BRG. 1 System clock/16 is the input to the eSPI BRG.
4–7 PM0	Prescale modulus select. Specifies the divide ratio of the prescale divider in the eSPI clock generator. The eSPI baud rate generator clock source (either system clock or system clock divided by 16, depending on DIV16 bit) is divided by 2 x ([PM] + 1), a range from 2 to 32. For example, if the prescale modulus is set to PM=0x0011 and DIV16 is set, the SPI_CLK/system clock rate will be 16 x (2 x (0x0011 + 1)) = 128 **NOTE:** System clock as used here is defined to be platform clock divided by 2
8 ODD0	0 Even division - 2 x (PM + 1) x (15 x DIV16 + 1) - 50% duty cycle 1 Odd division - (2 x PM + 1) x (15 x DIV16 + 1) (except for PM = 0 where it divides by 2 x (7 x DIV16 + 1)); duty cycle is (PM + 1) ÷ (2 x PM + 1) for DIV16 = 0; duty cycle is 50% for DIV16 = 1.
9–10 -	This field is reserved. Reserved
11 POL0	CS0 Polarity. 1 Asserted Low, Negated High 0 Asserted High, Negated Low.
12–15 LEN0	Character length in bits per character. Supports a range from 1-bit to 16-bit data characters.
16–19 CS0BEF	CS assertion time in bits before frame start (that is, before clock toggles) Example: CS0BEF = 0010 inserts 2 bits time gap between CS0 assertion to clock toggle
20–23 CS0AFT	CS assertion time in bits after frame end (that is, after clock finishes toggling) Example: CS0AFT = 0010 inserts 2 bits time gap between clock stop to CS0 negation
24–28 CS0CG	Clock gap insert gaps between transmitted frames according to this size (during this time, chip select is negated). Chip select is negated minimum time of 1 bit time. Example: CS0CG = 00101 inserts 5 + 1 = 6 bits time gap between every two consecutive frames
29–31 -	This field is reserved. Reserved

15.3.8 eSPI CS1 mode register (ESPI_SPMODE1)

The eSPI CS1 mode register (SPMODE1) controls eSPI master operation with chip select 1.

Address: 11_0000h base + 24h offset = 11_0024h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	CI1	CP1	REV1	DIV161	PM1				ODD1	Reserved		POL1	LEN1			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CS1BEF				CS1AFT				CS1CG					Reserved		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ESPI_SPMODE1 field descriptions

Field	Description
0 CI1	Clock invert. Inverts eSPI clock polarity. See Figure 15-13 and Figure 15-14 for more information 0 The inactive state of SPI_CLK is low. 1 The inactive state of SPI_CLK is high.
1 CP1	Clock phase. Selects the transfer format. See Figure 15-13 and Figure 15-14 for more information. 0 SPI_CLK starts toggling at the middle of the data transfer. 1 SPI_CLK starts toggling at the beginning of the data transfer.
2 REV1	Reverse data mode. Determines the receive and transmit character bit order. 0 lsb of the character sent and received first 1 msb of the character sent and received first-for 8/16 bits data character only
3 DIV161	Divide by 16. Selects the clock source for the eSPI baud rate generator (eSPI BRG) when configured as an eSPI master. **NOTE:** System clock as used here is defined to be platform clock divided by 2. 0 System clock is the input to the eSPI BRG. 1 System clock/16 is the input to the eSPI BRG.
4–7 PM1	Prescale modulus select. Specifies the divide ratio of the prescale divider in the eSPI clock generator. The eSPI baud rate generator clock source (either system clock or system clock divided by 16, depending on DIV16 bit) is divided by 2 x ([PM] + 1), a range from 2 to 32. For example, if the prescale modulus is set to PM = 0x0011 and DIV16 is set, the SPI_CLK/system clock rate will be 16 x (2 x (0x0011 + 1)) = 128 **NOTE:** System clock as used here is defined to be platform clock divided by 2.

Table continues on the next page...

ESPI_SPMODE1 field descriptions (continued)

Field	Description
8 ODD1	0 Even division-2 x (PM + 1) x (15 x DIV16 + 1) - 50% duty cycle 1 Odd division-(2 x PM + 1) x (15 x DIV16 + 1) (except for PM = 0 where it divides by 2 x (7 x DIV16 + 1)); duty cycle is (PM + 1) ÷ (2 x PM + 1) for DIV16 = 0; duty cycle is 50% for DIV16 = 1.
9–10 -	This field is reserved. Reserved
11 POL1	CS1 Polarity. 1 Asserted Low, Negated High 0 Asserted High, Negated Low.
12–15 LEN1	Character length in bits per character. Supports a range from 1-bit to 16-bit data characters.
16–19 CS1BEF	CS assertion time in bits before frame start (that is, before clock toggles) Example: CS1BEF = 0010 inserts 2 bits time gap between CS1 assertion to clock toggle.
20–23 CS1AFT	CS assertion time in bits after frame end (that is, after clock finishes toggling) Example: CS1AFT = 0010 inserts 2 bits time gap between clock stop to CS1 negation.
24–28 CS1CG	Clock Gap insert gaps between transmitted frames according to this size (during this time, chip select is negated). Chip select is negated minimum time of 1 bit time. Example: CS1CG = 00101 inserts 5 + 1 = 6 bits time gap between every two consecutive frames.
29–31 -	This field is reserved. Reserved

15.3.9 eSPI CS2 mode register (ESPI_SPMODE2)

The eSPI CS2 mode register (SPMODE2) controls the eSPI master operation with chip select 2.

Address: 11_0000h base + 28h offset = 11_0028h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CI2	CP2	REV2	DIV162	PM2				ODD2	Reserved		POL2	LEN2			
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CS2BEF				CS2AFT				CS2CG					Reserved		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ESPI_SPMODE2 field descriptions

Field	Description
0 CI2	Clock invert. Inverts eSPI clock polarity. See Figure 15-13 and Figure 15-14 for more information 0 The inactive state of SPI_CLK is low. 1 The inactive state of SPI_CLK is high.
1 CP2	Clock phase. Selects the transfer format. See Figure 15-13 and Figure 15-14 for more information. 0 SPI_CLK starts toggling at the middle of the data transfer. 1 SPI_CLK starts toggling at the beginning of the data transfer.
2 REV2	Reverse data mode. Determines the receive and transmit character bit order. 0 lsb of the character sent and received first 1 msb of the character sent and received first-for 8/16 bits data character only
3 DIV162	Divide by 16. Selects the clock source for the eSPI baud rate generator (eSPI BRG) when configured as an eSPI master. **NOTE:** System clock as used here is defined to be platform clock divided by 2. 0 System clock is the input to the eSPI BRG. 1 System clock/16 is the input to the eSPI BRG.
4–7 PM2	Prescale modulus select. Specifies the divide ratio of the prescale divider in the eSPI clock generator. The eSPI baud rate generator clock source (either system clock or system clock divided by 16, depending on DIV16 bit) is divided by 2 x ([PM] + 1), a range from 2 to 32. For example, if the prescale modulus is set to PM = 0x0011 and DIV16 is set, the system clock/SPI_CLK ratio will be 16 x (2 x (0x0011 + 1)) = 128 **NOTE:** System clock as used here is defined to be platform clock divided by 2.
8 ODD2	0 Even division-2 x (PM + 1) x (15 x DIV16 + 1) - 50% duty cycle 1 Odd division-(2 x PM + 1) x (15 x DIV16 + 1) (except for PM = 0 where it divides by 2 x (7 x DIV16 + 1)); duty cycle is (PM + 1) ÷ (2 x PM + 1) for DIV16 = 0; duty cycle is 50% for DIV16 = 1
9–10 -	This field is reserved. Reserved
11 POL2	CS2 Polarity. 1 Asserted Low, Negated High 0 Asserted High, Negated Low.
12–15 LEN2	Character length in bits per character. Supports a range from 1-bit to 16-bit data characters.
16–19 CS2BEF	CS assertion time in bits before frame start (that is, before clock toggles) Example: CS2BEF = 0010 inserts 2 bits time gap between CS2 assertion to clock toggle
20–23 CS2AFT	CS assertion time in bits after frame end (that is, after clock finishes toggling) Example: CS2AFT = 0010 inserts 2 bits time gap between clock stop to CS2 negation
24–28 CS2CG	Clock Gap insert gaps between transmitted frames according to this size (during this time, chip select is negated). Chip select is negated minimum time of 1 bit time. Example: CS1CG = 00101 inserts 5 + 1 = 6 bits time gap between every two consecutive frames
29–31 -	This field is reserved. Reserved

15.3.10 eSPI CS3 mode register (ESPI_SPMODE3)

The eSPI CS3 mode register (SPMODE3) controls the eSPI master operation with chip select 3.

Address: 11_0000h base + 2Ch offset = 11_002Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CI3	CP3	REV3	DIV163	PM3				ODD3	Reserved		POL3	LEN3			
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CS3BEF				CS3AFT				CS3CG					Reserved		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ESPI_SPMODE3 field descriptions

Field	Description
0 CI3	Clock invert. Inverts eSPI clock polarity. See Figure 15-13 and Figure 15-14 for more information 0 The inactive state of SPI_CLK is low. 1 The inactive state of SPI_CLK is high.
1 CP3	Clock phase. Selects the transfer format. See Figure 15-13 and Figure 15-14 for more information. 0 SPI_CLK starts toggling at the middle of the data transfer. 1 SPI_CLK starts toggling at the beginning of the data transfer.
2 REV3	Reverse data mode. Determines the receive and transmit character bit order. 0 lsb of the character sent and received first 1 msb of the character sent and received first - for 8/16 bits data character only
3 DIV163	Divide by 16. Selects the clock source for the eSPI baud rate generator (eSPI BRG) when configured as an eSPI master. **NOTE:** System clock as used here is defined to be platform clock divided by 2 0 System clock is the input to the eSPI BRG. 1 System clock/16 is the input to the eSPI BRG.
4–7 PM3	Prescale modulus select. Specifies the divide ratio of the prescale divider in the eSPI clock generator. The eSPI baud rate generator clock source (either system clock or system clock divided by 16, depending on DIV16 bit) is divided by 2 x ([PM] + 1), a range from 2 to 32. For example, if the prescale modulus is set to PM = 0x0011 and DIV16 is set, the SPI_CLK/system clock rate will be 16 x (2 x (0x0011 + 1)) = 128 **NOTE:** System clock as used here is defined to be platform clock divided by 2

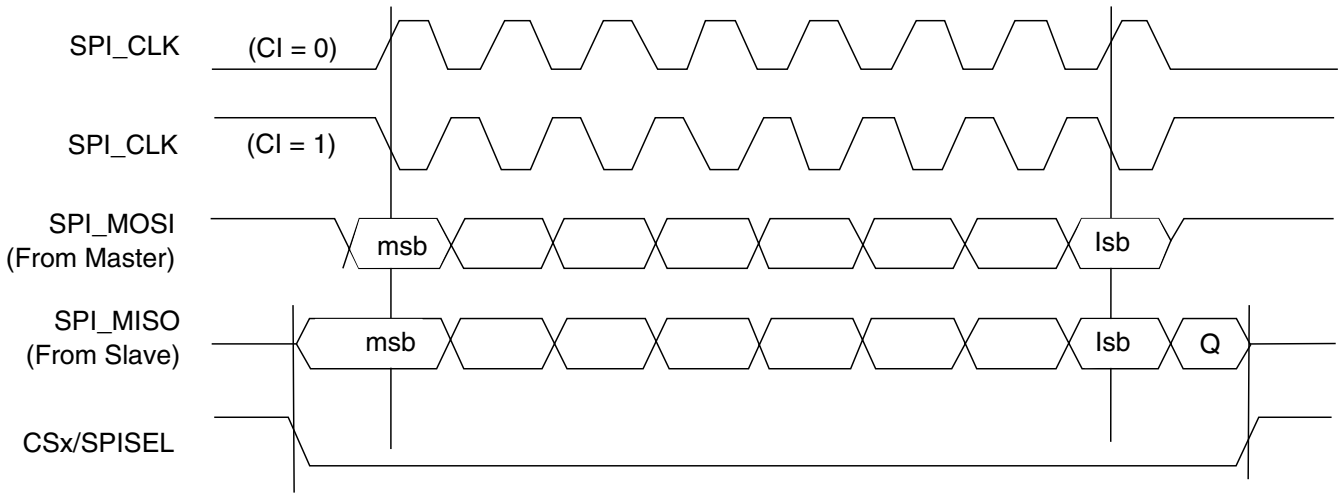
Table continues on the next page...

ESPI_SPMODE3 field descriptions (continued)

Field	Description
8 ODD3	0 Even division-2 x (PM + 1) x (15 x DIV16 + 1) - 50% duty cycle 1 Odd division-(2 x PM + 1) x (15 x DIV16 + 1) (except for PM = 0 where it divides by 2 x (7 x DIV16 + 1)); duty cycle is (PM + 1) ÷ (2 x PM + 1) for DIV16 = 0; duty cycle is 50% for DIV16 = 1
9–10 -	This field is reserved. Reserved
11 POL3	CS Polarity. 1 Asserted Low, Negated High 0 Asserted High, Negated Low.
12–15 LEN3	Character length in bits per character. Supports a range from 1-bit to 16-bit data characters.
16–19 CS3BEF	CS assertion time in bits before frame start (that is, before clock toggles) Example: CS3BEF = 0010 inserts 2 bits time gap between CS3 assertion to clock toggle
20–23 CS3AFT	CS assertion time in bits after frame end (that is, after clock finishes toggling) Example: CS3AFT = 0010 inserts 2 bits time gap between clock stop to CS3 negation
24–28 CS3CG	Clock Gap insert gaps between transmitted frames according to this size (during this time, chip select is negated). Chip select is negated minimum time of 1 bit time. Example: CS1CG = 00101 inserts 5 + 1 = 6 bits time gap between every two consecutive frames
29–31 -	This field is reserved. Reserved

15.4 eSPI transfer formats

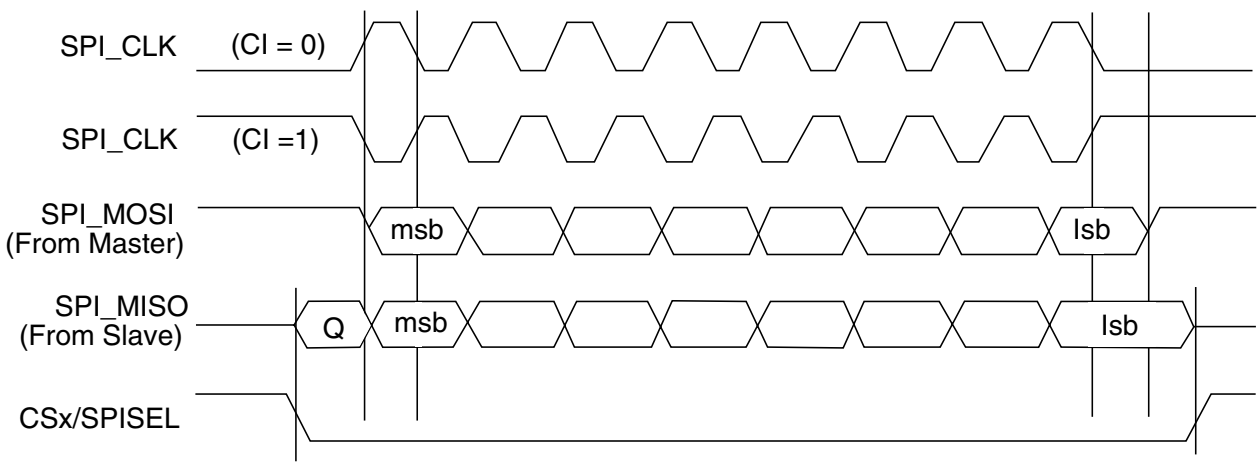
Figure below shows the eSPI transfer format in which SPI_CLK starts toggling in the middle of the transfer (SPMODEx[CPx] = 0).



NOTE: Q = Undefined Signal.

Figure 15-13. eSPI transfer format with SPMODEx[CPx] = 0

Figure below shows the eSPI transfer format in which SPI_CLK starts toggling at the beginning of the transfer (SPMODEx[CPx] = 1).



NOTE: Q = Undefined Signal.

Figure 15-14. eSPI transfer format with SPMODEx[CPx] = 1

15.5 CI and CP values for various eSPI devices

1. Regular devices-eSPI mode0-CI = CP = 0
2. Regular devices-eSPI mode3-CI = CP = 1

   For Winbond devices DO should also be set for dual output read command.
3. RapidS mode0-CI = 0, CP = 1, HLD = 1
4. RapidS mode3-CI = 1, CP = 0

15.6 eSPI programming examples

This section provides eSPI programming examples for 24-bit address memory and 16-bit address memory.

15.6.1 24-bit address example

The following sequence initializes the eSPI to read 36 bytes from 24-bit address memory, start address = 0x00_0040:

1. Configure a parallel I/O signal to operate as the eSPI CS1 output signal.
2. Write 0xFFFF_FFFF to SPIE to clear any previous events. Configure SPIM to enable all desired eSPI interrupts.
3. Configure SPMODE = 0x8000_100F to enable normal operation, eSPI enabled.
4. Configure SPMODE1 = 0x2417_1108-REV1 = 1, PM1 = 4 (divide eSPI input clock by 10), LEN1 = 7, POL1 = 1, CS1BEF = CS1AFT = CS1CG = 1.
5. Configure SPITF = 0x0300_0040-0x03 is read opcode while 0x00_0040 is the 24-bit start address.
6. Configure SPCOM = 0x0004_0027 so 4 bytes are skipped (1 for opcode and 3 for 24-bit address), TRANLEN = 36 + 4 - 1.

15.6.2 16-bit address example

The following sequence initializes the eSPI to read 36 bytes from 16-bit address memory, start address = 0x0040:

1. Configure a parallel I/O signal to operate as the eSPI CS1 output signal.
2. Write 0xFFFF_FFFF to SPIE to clear any previous events. Configure SPIM to enable all desired eSPI interrupts.
3. Configure SPMODE = 0x8000_100F to enable normal operation, eSPI enabled.
4. Configure SPMODE1 = 0x2417_1108-REV1 = 1, PM1 = 4 (divide eSPI input clock by 10), LEN1 = 7, POL1 = 1, CS1BEF = CS1AFT = CS1CG = 1.
5. Configure SPITF = 0x0300_40xx (xx is don't care)-0x03 is read opcode while 0x0040 is the 16-bit start address.
6. Configure SPCOM = 0x0003_0026 so 3 bytes are skipped (1 for opcode and 2 for 16-bit address), TRANLEN = 36 + 3 - 1.

Chapter 16
Enhanced Secured Digital Host Controller (eSDHC)

16.1 Overview

The enhanced secured digital host controller (eSDHC) provides interface between the host system and the SD/SDIO/MMC cards, as depicted in the figure below.

The eSDHC acts as a bridge, passing host bus transactions to the SD/SDIO/MMC cards by sending commands and performing data accesses to/from the cards.

It handles the SD/SDIO/MMC protocols at the transmission level.

Different types of cards supported by the eSDHC are described below:

- MultiMediaCard (MMC)

  MMC is a universal low-cost data storage and communication medium designed to cover a wide area of applications, including mobile video and gaming, which are available from either pre-loaded MMC cards or downloadable from cellular phones, WLAN, or other wireless networks. Old MMC cards are based on a seven-pin serial bus with a single data pin, while the new high-speed MMC communication is based on advanced 11-pin serial bus designed to operate in a low voltage range.

- 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.

- 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.

The eSDHC acts as a bridge, passing host bus transactions to MMC/SD/SDIO cards by sending commands and performing data accesses to or from the cards. It handles the MMC/SD/SDIO protocol at the transmission level. The figure below shows connection of the eSDHC.

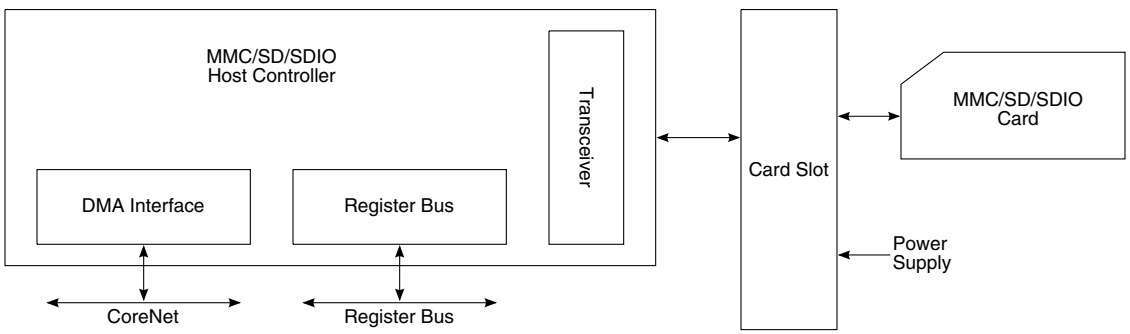


Figure 16-1. System connection of the eSDHC

The figure below is a high-level block diagram of eSDHC.

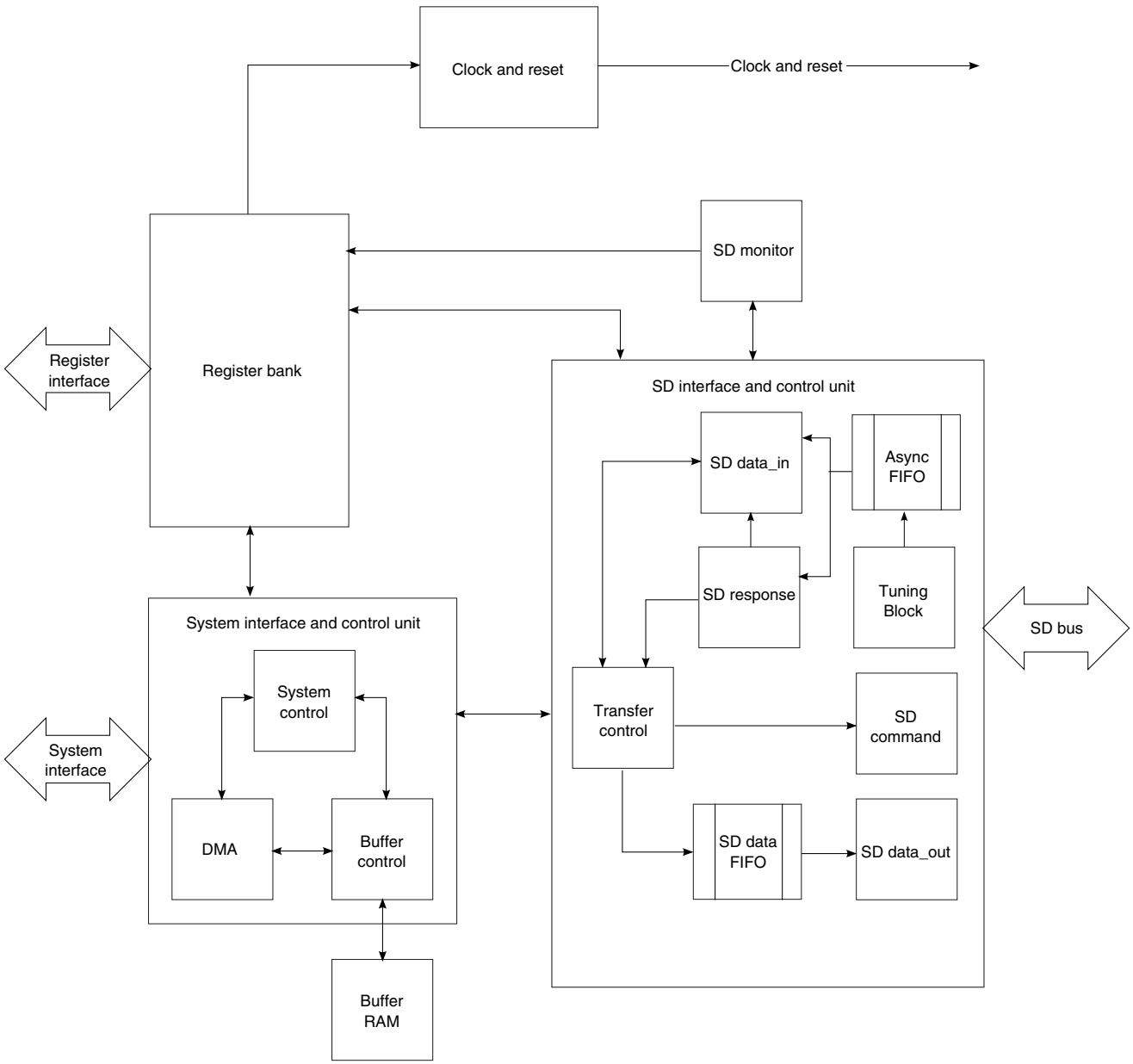


Figure 16-2. Enhanced secure digital host controller block diagram

16.1.1 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 MMC 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, MMC, and their variants, such as mini and micro.
- Card bus clock frequency up to 52 MHz
- Supports 1- or 4-bit SD and SDIO modes, 1- or 4- or 8-bit MMC 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
- Supports external SD bus voltage selection by register configuration
- 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

16.1.2 Modes and operations

16.1.2.1 Data transfer modes

The eSDHC can select the following modes for data transfer:

- SD 1 bit
- SD 4 bit
- MMC 1 bit
- MMC 4 bit
- MMC 8 bit

- Identification mode (up to 400 KHz)
- MMC full speed mode (up to 20 MHz)
- MMC high speed mode (up to 52 MHz)
- SD/SDIO full speed mode (up to 25 MHz)
- SD/SDIO high speed mode (up to 50 MHz)

16.2 External signals

16.2.1 External signals overview

The eSDHC has the following associated I/O signals:

- SDHC_CLK is an internally generated clock signal that drives the MMC, SDIO, or SD card.
- SDHC_CMD I/O sends commands to and receives responses from the card.
- SDHC_DAT[7:0] performs data transfers between the eSDHC and the card. If the eSDHC is desired to support a 4 bit data transfer, DAT[7:4] can also be optional and tied high.
- SDHC_CD_B and SDHC_WP are card detection and write protection signals from the socket.
- SDHC_CMD_DIR is an output signal for CMD line direction control
- SDHC_DAT0_DIR is an output signal DAT0 line direction control
- SDHC_DAT123_DIR is an output signal for DAT1-DAT3 lines direction control
- SD_VS is an output signal that controls the voltage of external SD Bus Supply to be high voltage (around 3.0V) or low voltage (around 1.8V). ‘0’ stands for high voltage range Optional output

16.2.2 eSDHC signal descriptions

Table 16-1. Signal properties

Name	Port	Function	Reset state	Pull up
SDHC_CLK	O	Clock for MMC/SD/SDIO card	0	N/A
SDHC_CMD	I/O	CMD line connect to card	1	Pull up
SDHC_CMD_DIR	O	CMD line direction control	0	N/A
SDHC_DAT7	I/O	DAT7 line in 8-bit mode Not used in other modes	1	Pull up
SDHC_DAT6	I/O	DAT6 line in 8-bit mode Not used in other modes	1	Pull up

Table continues on the next page...

Table 16-1. Signal properties (continued)

Name	Port	Function	Reset state	Pull up
SDHC_DAT5	I/O	DAT5 line in 8-bit mode Not used in other modes	1	Pull up
SDHC_DAT4	I/O	DAT4 line in 8-bit mode Not used in other modes	1	Pull up
SDHC_DAT3	I/O	DAT3 line in 4/8-bit mode	0	Pull up. Do not use DAT3 pin as a CD pin.
SDHC_DAT2	I/O	DAT2 line or read wait in 4/8-bit mode Read wait in 1-bit mode	1	Pull up
SDHC_DAT1	I/O	DAT1 line in 4/8-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_DAT0_DIR	O	DAT0 line direction control	0	N/A
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
SDHC_DAT123_DIR	O	DAT1-DAT3 lines direction control	0	N/A

16.3 eSDHC memory map/register definition

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.

eSDHC memory map

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
11_4000	SDMA system address register/Block attributes 2 (eSDHC_DSADDR)	32	R/W	0000_0000h	16.3.1/822

Table continues on the next page...

eSDHC memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
11_4004	Block attributes register (eSDHC_BLKATTR)	32	R/W	0000_0000h	16.3.2/823
11_4008	Command argument register (eSDHC_CMDARG)	32	R/W	0000_0000h	16.3.3/824
11_400C	Transfer type register (eSDHC_XFERTYP)	32	R/W	0000_0000h	16.3.4/824
11_4010	Command response 0 register (eSDHC_CMDRSP0)	32	R	0000_0000h	16.3.5/829
11_4014	Command response 1 register (eSDHC_CMDRSP1)	32	R	0000_0000h	16.3.6/829
11_4018	Command response 2 register (eSDHC_CMDRSP2)	32	R	0000_0000h	16.3.7/830
11_401C	Command Response 3 register (eSDHC_CMDRSP3)	32	R	0000_0000h	16.3.8/830
11_4020	Buffer data port register (eSDHC_DATPORT)	32	R/W	0000_0000h	16.3.9/832
11_4024	Present state register (eSDHC_PRSSTAT)	32	R/W	See section	16.3.10/ 833
11_4028	Protocol control register (eSDHC_PROCTL)	32	R/W	0000_0020h	16.3.11/ 838
11_402C	System control register when ESDHCCTL[CRS=0] (eSDHC_SYSCTL)	32	R/W	See section	16.3.12/ 841
11_402C	System Control Register when ESDHCCTL[CRS=1] (eSDHC_SYSCTL_ESDHCCTL_CRS_1)	32	R/W	See section	16.3.13/ 845
11_4030	Interrupt status register (eSDHC_IRQSTAT)	32	R/W	0000_0000h	16.3.14/ 848
11_4034	Interrupt status enable register (eSDHC_IRQSTATEN)	32	R/W	137F_01FFh	16.3.15/ 853
11_4038	Interrupt signal enable register (eSDHC_IRQSIGEN)	32	R/W	0000_0000h	16.3.16/ 856
11_403C	Auto CMD error status register/ System control 2 (eSDHC_AUTOCERR)	32	R	0000_0000h	16.3.17/ 859
11_4040	Host controller capabilities register (eSDHC_HOSTCAPBLT)	32	R	See section	16.3.18/ 863
11_4044	Watermark level register (eSDHC_WML)	32	R/W	0010_0010h	16.3.19/ 866
11_4050	Force event register (eSDHC_FEVT)	32	W	0000_0000h	16.3.20/ 868
11_4058	ADMA system address low register (eSDHC_ADSADDRL)	32	R/W	0000_0000h	16.3.21/ 870
11_405C	ADMA system address high register (eSDHC_ADSADDRH)	32	R/W	0000_0000h	16.3.22/ 871
11_40FC	Host controller version register (eSDHC_HOSTVER)	32	R	0000_2002h	16.3.23/ 871
11_4104	DMA error address register (eSDHC_DMAERRADDR)	32	R	0000_0000h	16.3.24/ 872
11_4104	DMA error address low register (eSDHC_DMAERRADDRL)	32	R	0000_0000h	16.3.25/ 873
11_4108	DMA error address high register (eSDHC_DMAERRADDRH)	32	R	0000_0000h	16.3.26/ 873

Table continues on the next page...

eSDHC memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
11_410C	DMA error attribute register (eSDHC_DMAERRATTR)	32	R	0000_0000h	16.3.27/ 874
11_4114	Host controller capabilities register 2 (eSDHC_HOSTCAPBLT2)	32	R	0000_AF07h	16.3.28/ 875
11_4120	Tuning control register (eSDHC_TCR)	32	R/W	8000_0020h	16.3.29/ 877
11_4140	SD direction control register (eSDHC_SD_DIRCTL)	32	R/W	0000_0001h	16.3.30/ 878
11_440C	eSDHC control register (eSDHC_ESDHCCTL)	32	R/W	0000_0000h	16.3.31/ 879

16.3.1 SDMA system address register/Block attributes 2 (eSDHC_DSADDR)

The DSADDR contains the physical system memory address used for single DMA transfers or second argument for Auto CMD23.

Address: 11_4000h base + 0h offset = 11_4000h

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	DS_ADDR																															
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

eSDHC_DSADDR field descriptions

Field	Description
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: 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 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.

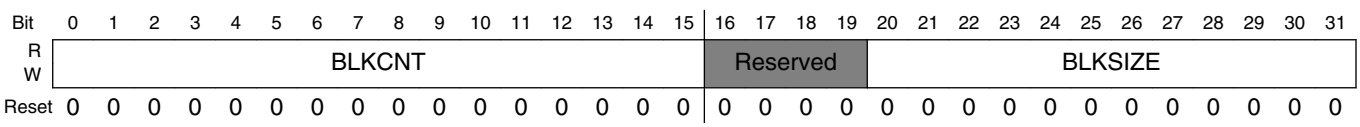
eSDHC_DSADDR field descriptions (continued)

Field	Description
	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.

16.3.2 Block attributes register (eSDHC_BLKATTR)

The BLKATTR is used to configure the number of data blocks and the number of bytes in each block.

Address: 11_4000h base + 4h offset = 11_4004h



eSDHC_BLKATTR field descriptions

Field	Description
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. 0x0000 Stop count 0x0001 1 block 0x0002 2 blocks ... 0xFFFF 65535 blocks
16–19 -	This field is 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.

Table continues on the next page...

eSDHC_BLKATTR field descriptions (continued)

Field	Description
	0x000 No data transfer 0x001 1 byte 0x002 2 bytes 0x003 3 bytes 0x004 4 bytes ... 0x1FF 511 bytes 0x200 512 bytes ... 0x800 2048 bytes

16.3.3 Command argument register (eSDHC_CMDARG)

The CMDARG contains the SD/MMC command argument.

Address: 11_4000h base + 8h offset = 11_4008h

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	CMDARG																															
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

eSDHC_CMDARG field descriptions

Field	Description
0–31 CMDARG	Command argument. The SD/MMC command argument is specified as bits 39-8 of the command format in the SD or MMC Specification. This register is write protected when PRSSTAT[CIHB] is set.

16.3.4 Transfer type register (eSDHC_XFERTYP)

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 16-6. 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 16-7. Relationship Between Parameters and the Name of the Response Type

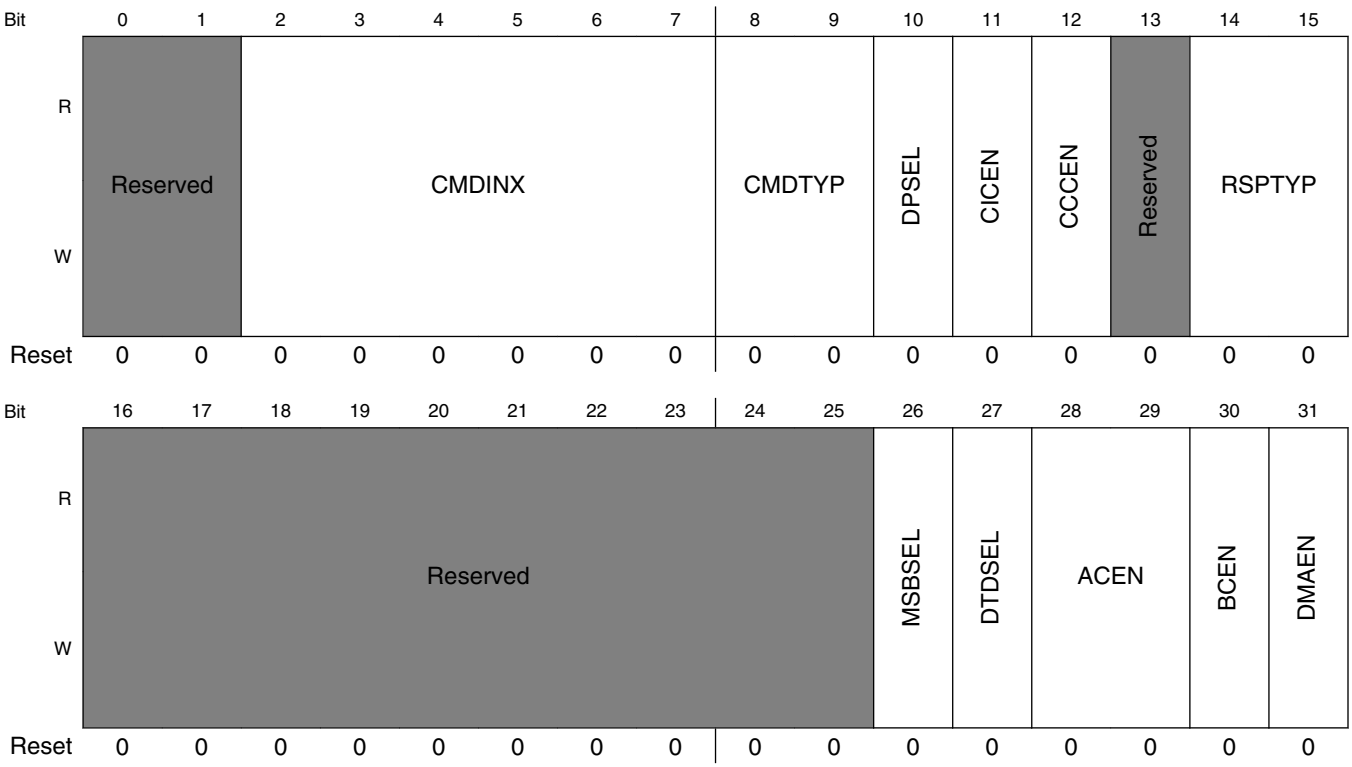
Response type	Index Check Enable	CRC Check Enable	Name of Response Type
00	0	0	No Response
01	0	1	R2
10	0	0	R3, R4
10	1	1	R1, R5, R6, R7
11	1	1	R1b, R5b

- 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.

Address: 11_4000h base + Ch offset = 11_400Ch



eSDHC_XFERTYP field descriptions

Field	Description
0–1 -	This field is reserved.
2–7 CMDINX	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. 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].

Table continues on the next page...

eSDHC_XFERTYP field descriptions (continued)

Field	Description
	• 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). 00 Normal other commands 01 Suspend CMD52 for writing bus suspend in CCCR 10 Resume CMD52 for writing function select in CCCR 11 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. 0 No data present 1 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. 0 Disable 1 Enable
12 CCCEN	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 16-7.) 0 Disable 1 Enable
13 -	This field is reserved.
14–15 RSPTYP	Response type select. 00 No response 01 Response length 136 10 Response length 48 11 Response length 48, check busy after response
16–25 -	This field is reserved.
26 MSBSEL	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 16-6.)

Table continues on the next page...

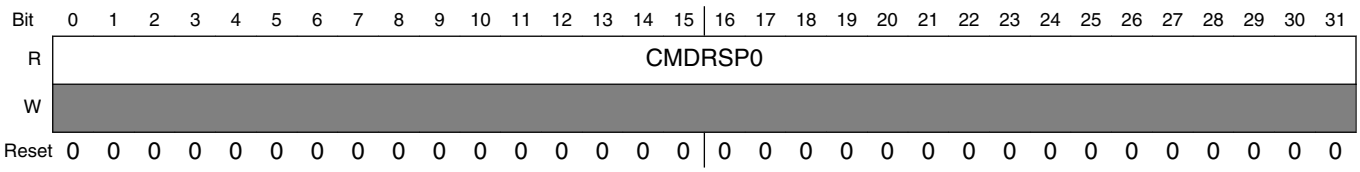
eSDHC_XFERTYP field descriptions (continued)

Field	Description
	0 Single block 1 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. 0 Write (host to card) 1 Read (card to host)
28–29 ACEN	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. 00 Auto CMD Disable 01 Auto CMD12 Enable 10 Auto CMD23 Enable 11 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. 0 Disable 1 Enable
31 DMAEN	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]. 0 Disable 1 Enable

16.3.5 Command response 0 register (eSDHC_CMDRSP0)

The CMDRSP0 is used to store part 0 of the response bits from the card.

Address: 11_4000h base + 10h offset = 11_4010h



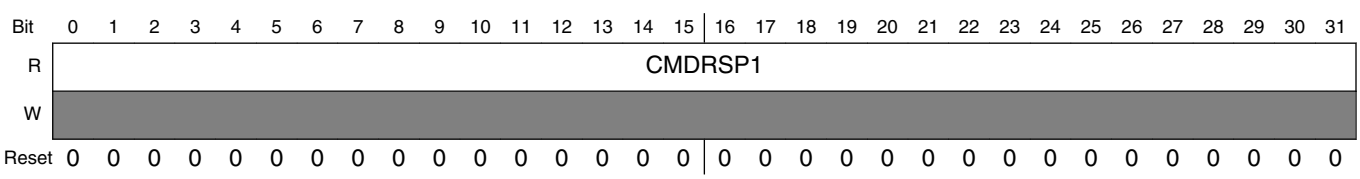
eSDHC_CMDRSP0 field descriptions

Field	Description
0–31 CMDRSP0	Command response 0. Refer to Command Response 3 register (eSDHC_CMDRSP3) for the mapping of command responses from the SD bus to this register for each response type.

16.3.6 Command response 1 register (eSDHC_CMDRSP1)

The CMDRSP1 is used to store part 1 of the response bits from the card.

Address: 11_4000h base + 14h offset = 11_4014h



eSDHC_CMDRSP1 field descriptions

Field	Description
0–31 CMDRSP1	Command response 1. Refer to Command Response 3 register (eSDHC_CMDRSP3) for the mapping of command responses from the SD bus to this register for each response type.

16.3.7 Command response 2 register (eSDHC_CMDRSP2)

The CMDRSP2 is used to store part 2 of the response bits from the card.

Address: 11_4000h base + 18h offset = 11_4018h

Bit	0–31
R	CMDRSP2
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

eSDHC_CMDRSP2 field descriptions

Field	Description
0–31 CMDRSP2	Command response 2. Refer to Command Response 3 register (eSDHC_CMDRSP3) for the mapping of command responses from the SD bus to this register for each response type.

16.3.8 Command Response 3 register (eSDHC_CMDRSP3)

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 16-12. 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

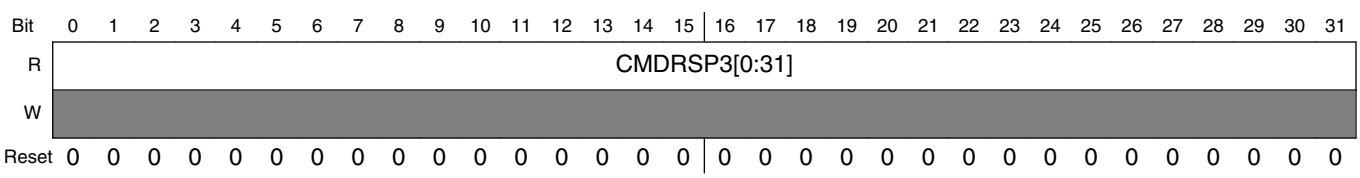
This table shows the following:

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.

Address: 11_4000h base + 1Ch offset = 11_401Ch



eSDHC_CMDRSP3 field descriptions

Field	Description
0–31 CMDRSP3[0:31]	Command response 3. Refer to Command Response 3 register (eSDHC_CMDRSP3) for the mapping of command responses from the SD bus to this register for each response type.

16.3.9 Buffer data port register (eSDHC_DATPORT)

The DATPORT is a 32-bit data port register used to access the internal buffer. Byte access is not allowed.

Address: 11_4000h base + 20h offset = 11_4020h

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	DATCONT																															
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

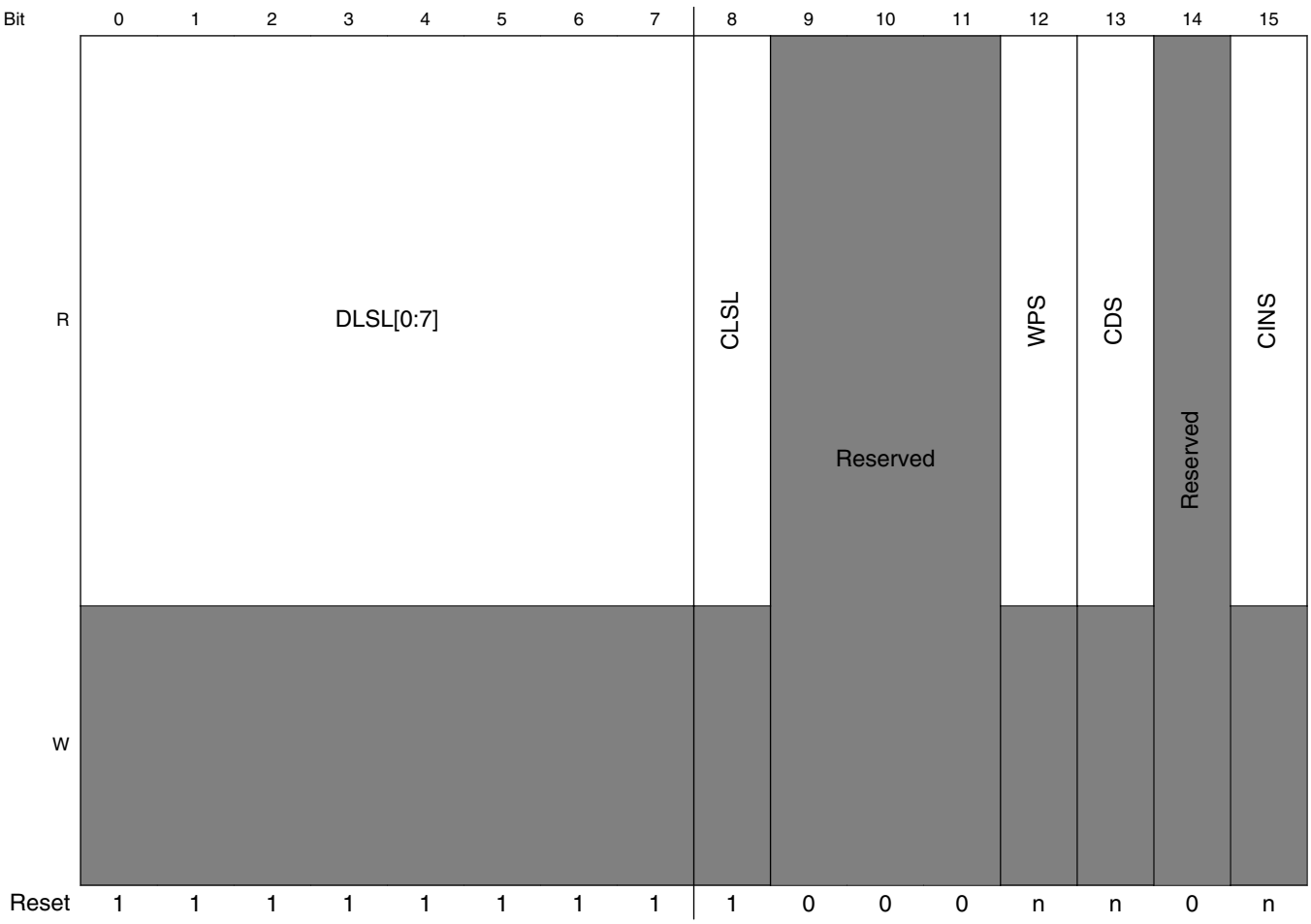
eSDHC_DATPORT field descriptions

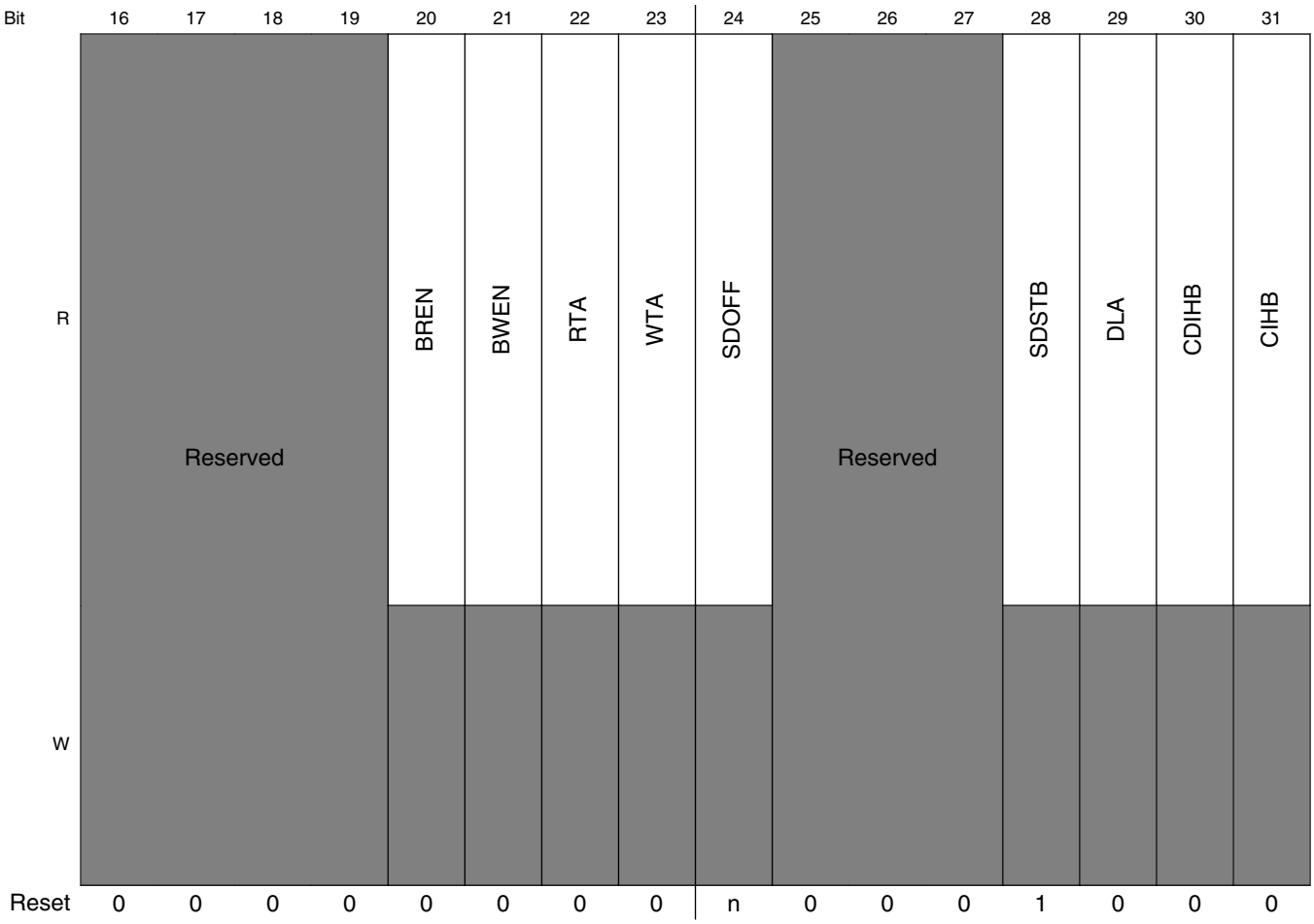
Field	Description
0–31 DATCONT	Data content. The buffer data port register is for 32-bit data access by the CPU. When the DMA is enabled, any write to this register is ignored, and any read from this register always yields zeros.

16.3.10 Present state register (eSDHC_PRSSTAT)

The host driver can get the status of the eSDHC from the PRSSTAT, which is a 32-bit, read-only register.

Address: 11_4000h base + 24h offset = 11_4024h





eSDHC_PRSSTAT field descriptions

Field	Description
0–7 DLSL[0:7]	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 DAT[0]: Data 0 line signal level DAT[1]: Data 1 line signal level DAT[2]: Data 2 line signal level DAT[3]: Data 3 line signal level DAT[4]: Data 4 line signal level DAT[5]: Data 5 line signal level DAT[6]: Data 6 line signal level DAT[7]: Data 7 line signal level
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 -	This field is reserved.

Table continues on the next page...

eSDHC_PRSSTAT field descriptions (continued)

Field	Description
12 WPS	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. 0 Write protected (SDHC_WP =1) 1 Write enabled (SDHC_WP =0)
13 CDS	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.) 0 No card present (SDHC_CD_B =1) 1 Card present (SDHC_CD_B =0)
14 -	This field is 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. The software reset for all in the system control register (SYSCTL) does not effect this bit. A software reset does not effect this bit. 0 Power on reset or no card 1 Card inserted
16–19 -	This field is 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. 0 Read disable 1 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. 0 Write disable 1 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:

Table continues on the next page...

eSDHC_PRSSTAT field descriptions (continued)

Field	Description
	• 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. 0 No valid data 1 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. 0 No valid data 1 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. 0 SD clock is active 1 SD clock is gated off
25–27 -	This field is 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. 0 Clock is changing frequency and not stable 1 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:

Table continues on the next page...

eSDHC_PRSSTAT field descriptions (continued)

Field	Description
	• 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). 0 DAT line inactive 1 DAT line active
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/MMC 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. 0 Can issue command which uses the DAT line 1 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/MMC 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. 0 Can issue command using only CMD line 1 Cannot issue command

16.3.11 Protocol control register (eSDHC_PROCTL)

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.

Address: 11_4000h base + 28h offset = 11_4028h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved					WECRM	WECINS	WECINT	Reserved				IABG	RWCTL	CREQ	SABGREQ
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					VOLT_SEL	DMAS		CDSS	CDTL	EMODE		Reserved	DTW[0:1]		LCTL
W																
Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

eSDHC_PROCTL field descriptions

Field	Description
0–4 -	This field is reserved.
5 WECRM	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. 0 Disable 1 Enable
6 WECINS	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. 0 Disable 1 Enable
7 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. 0 Disable 1 Enable
8–11 -	This field is 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. 0 Disable 1 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. 0 Disable read wait control, and stop SD clock at block gap when SABGREQ bit is set 1 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:

Table continues on the next page...

eSDHC_PROCTL field descriptions (continued)

Field	Description
	• 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. 0 No effect 1 Restart
15 SABGREQ	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 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]. 0 Transfer 1 Stop
16–20 -	This field is reserved.
21 VOLT_SEL	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. 0 Change the SD Bus Supply voltage to high voltage range, around 3.0V 1 Change the SD bus supply voltage to low voltage range, around 1.8V
22–23 DMAS	DMA select. This field is valid while DMA (SDMA or ADMA) is enabled and selects the DMA operation. 00 Single DMA is selected 01 ADMA1 is selected 10 11 64-bit address ADMA2 is selected
24 CDSS	Card detect signal selection. This bit selects the source for the card detection. 0 SD CD pin is selected (for normal purposes) 1 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. 0 Card detect test level is 0, no card inserted 1 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.

Table continues on the next page...

eSDHC_PROCTL field descriptions (continued)

Field	Description
	00 Big endian mode 01 Reserved 10 Little endian mode 11 Reserved
28 -	This field is reserved.
29–30 DTW[0:1]	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. 00 1-bit mode 01 4-bit mode 11 Reserved
31 LCTL	LED control. This bit, fully controlled by the host driver, is used to caution the user not to remove the card while the card is being accessed. If the software is going to issue multiple SD commands, this bit can be set during all these transactions. It is not necessary to change for each transaction. When the software issues multiple SD commands, setting the bit once before the first command is sufficient: it is not necessary to reset the bit between commands. 0 LED off 1 LED on

16.3.12 System control register when ESDHCCTL[CRS=0] (eSDHC_SYSCTL)

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.

Address: 11_4000h base + 2Ch offset = 11_402Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Reserved				INITA	RSTD	RSTC	RSTA	Reserved				DTOCV			
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	SDCLKFS								DVS[0:3]				SDCLKEN	Reserved		
Reset	1	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0

eSDHC_SYSCTL field descriptions

Field	Description
0–3 -	This field is 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.
	0 No reset
	1 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. 0 No reset 1 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

Table continues on the next page...

eSDHC_SYSCTL field descriptions (continued)

Field	Description
	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. 0 No reset 1 Reset
8–11 -	This field is reserved.
12–15 DTOCV	Data timeout counter value. This value determines the interval by which DAT line timeouts are detected. Refer to the data timeout error bit in the 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. 0000 SDCLK x 2^{13} 0001 SDCLK x 2^{14} ... 1110 SDCLK x 2^{27} 1111 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 (ipg_perclk) and the following divisor bits. The frequency of SDCLK is set by the following formula: Clock Frequency = (Base Clock) ÷ (prescaler x divisor) For example, 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 (ipg_perclk) 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: 0x00 Base clock 0x01 Base clock divided by 2 0x02 Base clock divided by 4 0x04 Base clock divided by 8

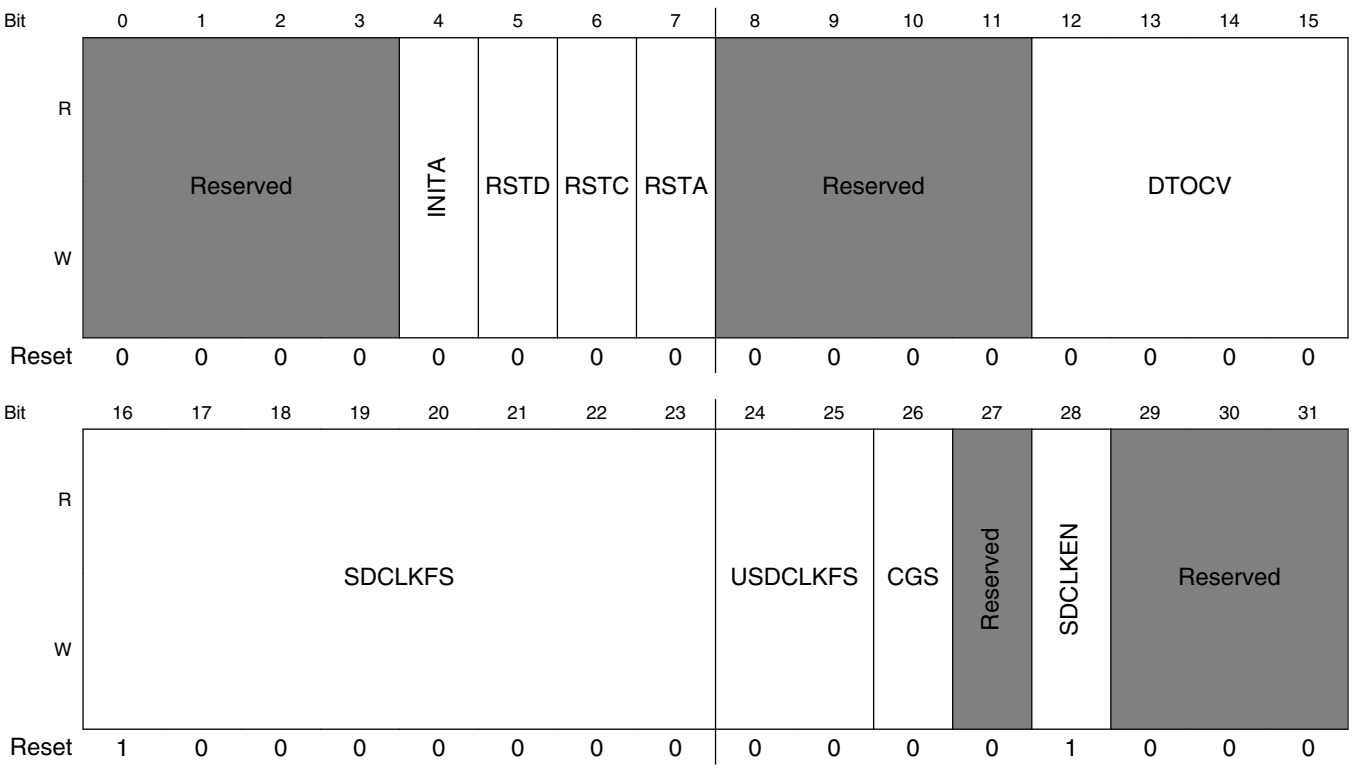
Table continues on the next page...

eSDHC_SYSCTL field descriptions (continued)

Field	Description
	0x10 Base clock divided by 32 0x08 Base clock divided by 16 0x20 Base clock divided by 64 0x40 Base clock divided by 128 0x80 Base clock divided by 256
24–27 DVS[0:3]	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: 0x0 Divisor by 1 0x1 Divisor by 2 0x2 Divisor by 3 ... 0xE Divisor by 15 0xF 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 -	This field is reserved.

16.3.13 System Control Register when ESDHCCTL[CRS=1] (eSDHC_SYSCTL_ESDHCCTL_CRS_1)

Address: 11_4000h base + 2Ch offset = 11_402Ch



eSDHC_SYSCTL_ESDHCCTL_CRS_1 field descriptions

Field	Description
0–3 -	This field is 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

Table continues on the next page...

eSDHC_SYSCTL_ESDHCCTL_CRS_1 field descriptions (continued)

Field	Description
	• 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. 0 No reset 1 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. 0 No reset 1 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. 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. 0 No reset 1 Reset
8–11 -	This field is reserved.
12–15 DTOCV	Data timeout counter value. This value determines the interval by which DAT line timeouts are detected. Refer to the data timeout error bit in the 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.

Table continues on the next page...

eSDHC_SYSCTL_ESDHCCTL_CRS_1 field descriptions (continued)

Field	Description
	0000 SDCLK x 2^{13} 0001 SDCLK x 2^{14} ... 1110 SDCLK x 2^{27} 1111 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) i.e.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)
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. 0 Divided clock mode is selected 1 Programmable clock mode is selected
27 -	This field is 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 -	This field is reserved.

16.3.14 Interrupt status register (eSDHC_IRQSTAT)

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 16-19. 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 16-20. 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 16-21. 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

Address: 11_4000h base + 30h offset = 11_4030h

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

eSDHC_IRQSTAT field descriptions

Field	Description
0–2 -	This field is reserved.
3 DMAE	DMA error. This bit indicates that DMA (SDMA or ADMA) transfer has failed. 0 No error 1 Error
4 -	This field is 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. 1 Error 0 No 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. 0 No error 1 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. 0 No error 1 Error
8 -	This field is 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. 0 No error 1 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. 0 No error 1 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 0 No error 1 Time out

Table continues on the next page...

eSDHC_IRQSTAT field descriptions (continued)

Field	Description
12 CIE	Command index error. Occurs if a command index error occurs in the command response. 0 No error 1 Error
13 CEBE	Command end bit error. Occurs when detecting that the end bit of a command response is 0. 0 No error 1 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. 0 No error 1 CRC error generated
15 CTOE	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. 0 No error 1 Time out
16–18 -	This field is 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. 1 Re-tuning should be performed 0 Re-tuning is not required
20–22 -	This field is 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. 0 No card interrupt 1 Generate card interrupt
24 CRM	Card removal. This status is set if the PRSSTAT[CINS] changes from 1 to 0.

Table continues on the next page...

eSDHC_IRQSTAT field descriptions (continued)

Field	Description
	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. 0 Card state unstable or inserted 1 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. 0 Card state unstable or removed 1 Card inserted
26 BRR	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 (eSDHC_PRSSTAT) for additional information. 0 Not ready to read buffer 1 Ready to read buffer
27 BWR	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 (eSDHC_PRSSTAT) for additional information. 0 Not ready to write buffer 1 Ready to write buffer
28 DINT	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. 0 No DMA Interrupt 1 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). 0 No block gap event 1 Transaction stopped at block gap

Table continues on the next page...

eSDHC_IRQSTAT field descriptions (continued)

Field	Description
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. 0 Transfer not complete 1 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]. 0 Command not complete 1 Command complete

16.3.15 Interrupt status enable register (eSDHC_IRQSTATEN)

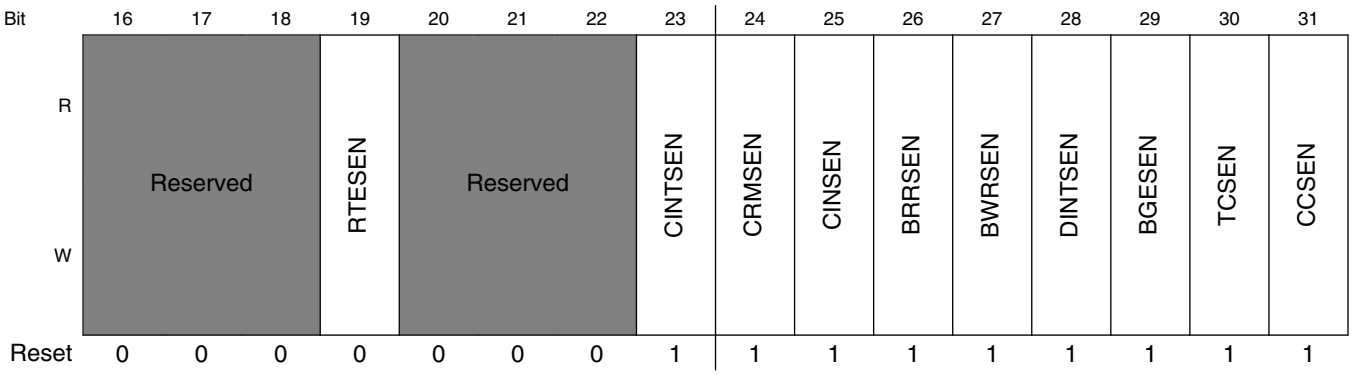
Setting the bits to 1 in the IRQSTATEN enables the corresponding interrupt status to be set by the specified event. If any bit is cleared, the corresponding interrupt status bit is also cleared (that is, when the bit in this register is cleared, the corresponding bit in interrupt status register, IRQSTAT, 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.

Address: 11_4000h base + 34h offset = 11_4034h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Reserved			DMAESEN	Reserved	TNESEN	ADMAESEN	AC12ESEN	Reserved	DEBESEN	DCESEN	DTOESEN	CIESEN	CEBESEN	CCESEN	CTOESEN
Reset	0	0	0	1	0	0	1	1	0	1	1	1	1	1	1	1



eSDHC_IRQSTATEN field descriptions

Field	Description
0–2 -	This field is reserved.
3 DMAESEN	DMA error status enable. 0 Masked 1 Enabled
4 -	This field is reserved.
5 TNESEN	Tuning error status enable 1 Enabled 0 Masked
6 ADMAESEN	ADMA error status enable. 0 Masked 1 Enabled
7 AC12ESEN	Auto CMD12 error status enable. 0 Masked 1 Enabled
8 -	This field is reserved.
9 DEBESEN	Data end bit error status enable. 0 Masked 1 Enabled
10 DCESEN	Data CRC error status enable. 0 Masked 1 Enabled
11 DTOESEN	Data timeout error status enable. 0 Masked 1 Enabled
12 CIESEN	Command index error status enable.

Table continues on the next page...

eSDHC_IRQSTATEN field descriptions (continued)

Field	Description
	0 Masked 1 Enabled
13 CEBESEN	Command end bit error status enable. 0 Masked 1 Enabled
14 CCESEN	Command CRC error status enable. 0 Masked 1 Enabled
15 CTOESEN	Command timeout error status enable. 0 Masked 1 Enabled
16–18 -	This field is reserved.
19 RTESEN	Re-tuning event status enable. 0 Masked 1 Enabled
20–22 -	This field is 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. 0 Masked 1 Enabled
24 CRMSEN	Card removal status enable. 0 Masked 1 Enabled
25 CINSEN	Card insertion status enable. 0 Masked 1 Enabled
26 BRRSEN	Buffer read ready status enable. 0 Masked 1 Enabled
27 BWRSEN	Buffer write ready status enable. 0 Masked 1 Enabled
28 DINTSEN	DMA interrupt status enable. 0 Masked 1 Enabled

Table continues on the next page...

eSDHC_IRQSTATEN field descriptions (continued)

Field	Description
29 BGESEN	Block gap event status enable. 0 Masked 1 Enabled
30 TCSEN	Transfer complete status enable. 0 Masked 1 Enabled
31 CCSEN	Command complete status enable. 0 Masked 1 Enabled

16.3.16 Interrupt signal enable register (eSDHC_IRQSIGEN)

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.

Address: 11_4000h base + 38h offset = 11_4038h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved			DMAEIEN	-	TNESEN	ADMAEIEN	AC12EIEN	Reserved	DEBEIEN	DCEIEN	DTOEIEN	CIEIEN	CEBEIEN	CCEIEN	CTOEIEN
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			RTEIEN	Reserved			CINTIEN	CRMIEN	CINSIEN	BRRIEN	BWRIEN	DINTIEN	BGEIEN	TCIEN	CCIEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eSDHC_IRQSIGEN field descriptions

Field	Description
0–2 -	This field is reserved.
3 DMAEIEN	DMA error interrupt enable. 0 Masked 1 Enabled
4 -	0 Masked 1 Enable
5 TNESEN	Tuning error interrupt enable 1 Enable 0 Masked
6 ADMAEIEN	ADMA Error Interrupt Enable.
7 AC12EIEN	Auto CMD12 error interrupt enable. 0 Masked 1 Enabled
8 -	This field is reserved.
9 DEBEIEN	Data end bit error interrupt enable. 0 Masked 1 Enabled
10 DCEIEN	Data CRC error interrupt enable. 0 Masked 1 Enabled
11 DTOEIEN	Data timeout error interrupt enable. 0 Masked 1 Enabled
12 CIEIEN	Command index error interrupt enable. 0 Masked 1 Enabled
13 CEBEIEN	Command end bit error interrupt enable. 0 Masked 1 Enabled
14 CCEIEN	Command CRC error interrupt enable. 0 Masked 1 Enabled
15 CTOEIEN	Command timeout error interrupt enable. 0 Masked 1 Enabled

Table continues on the next page...

eSDHC_IRQSIGEN field descriptions (continued)

Field	Description
16–18 -	This field is reserved.
19 RTEIEN	Re-tuning event interrupt enable 1 Enable 0 Masked
20–22 -	This field is reserved.
23 CINTIEN	Card interrupt interrupt enable. 0 Masked 1 Enabled
24 CRMIEN	Card removal interrupt enable. 0 Masked 1 Enabled
25 CINSIEN	Card insertion interrupt enable. 0 Masked 1 Enabled
26 BRRIEN	Buffer read ready interrupt enable. 0 Masked 1 Enabled
27 BWRIEN	Buffer write ready interrupt enable. 0 Masked 1 Enabled
28 DINTIEN	DMA interrupt enable. 0 Masked 1 Enabled
29 BGEIEN	Block gap event interrupt enable. 0 Masked 1 Enabled
30 TCIEN	Transfer complete interrupt enable. 0 Masked 1 Enabled
31 CCIEN	Command complete interrupt enable. 0 Masked 1 Enabled

16.3.17 Auto CMD error status register/ System control 2 (eSDHC_AUTOCERR)

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 16-25. Relationship Between Command CRC Error and Command Timeout Error for Auto CMD12

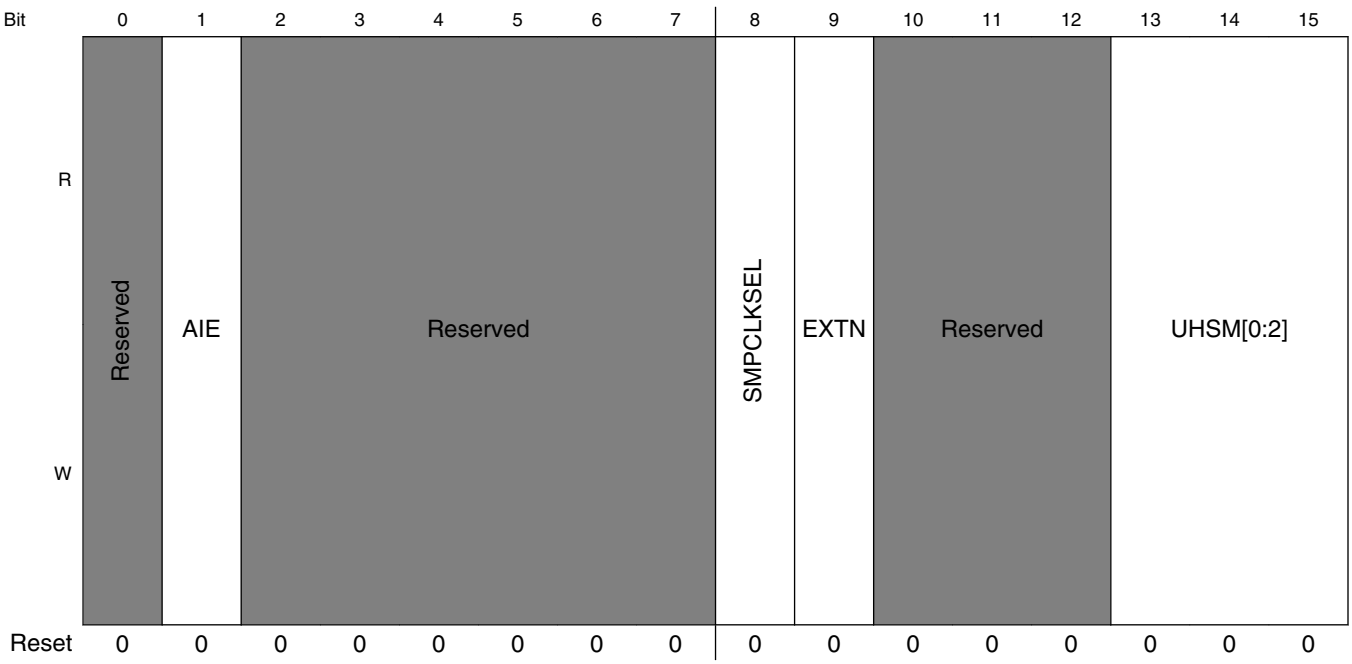
Auto CMD12 CRC Error	Auto CMD12 Timeout Error	Type of Error
0	0	No Error
0	1	Response Timeout Error
1	0	Response CRC Error
1	1	CMD line conflict

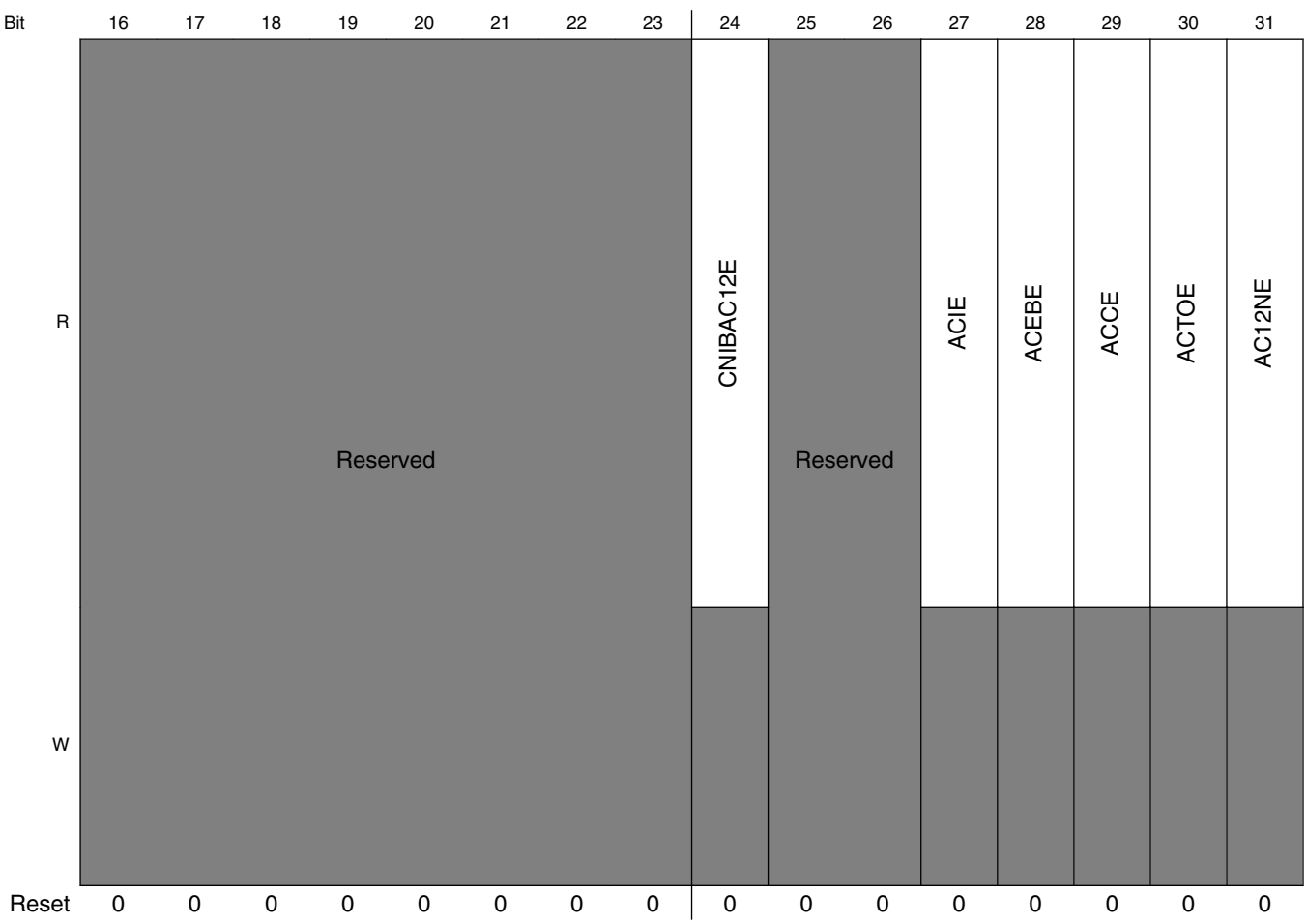
Changes in Auto CMD12 error status register (AUTOC12ERR) can be classified in three scenarios:

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.

Address: 11_4000h base + 3Ch offset = 11_403Ch





eSDHC_AUTOCERR field descriptions

Field	Description
0 -	This field is 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. 1 Enabled 0 Disabled
2–7 -	This field is 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. 1 Tuning procedure completed successfully 0 Tuning procedure unsuccessful
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 1 Execute tuning 0 Not tuned or tuning not compelted
10–12 -	This field is reserved.
13–15 UHSM[0:2]	UHS mode select. This field is used to select one of UHS-1 modes for SD 3.0 card; and select HS200 or DDR mode for MMC card. Host driver should reset SDCLKEN in system control register before changing this field, and then set SDCLKEN again. 000 SDR12 for SD, or max 52 MHz mode for SD 2.0 / MMC 4.2 or older spec 001 SDR25 for SD 010 SDR50 for SD 011 SDR104 for SD, HS200 for MMC 100 DDR 101-111 Reserved
16–23 -	This field is 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. 0 No error 1 Not Issued
25–26 -	This field is reserved.
27 ACIE	Auto CMD index error. Occurs if the command index error occurs in response to a command.

Table continues on the next page...

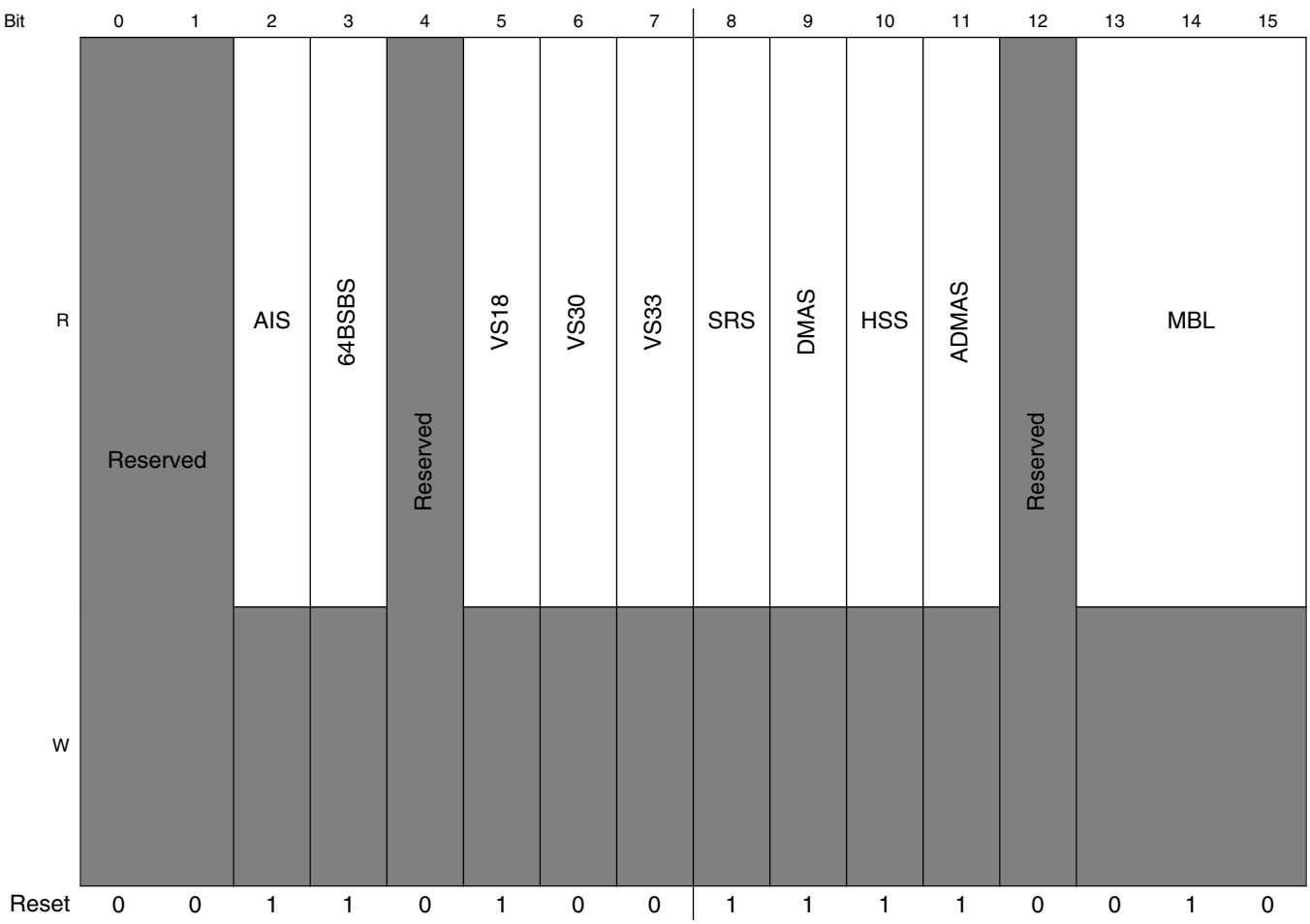
eSDHC_AUTOCERR field descriptions (continued)

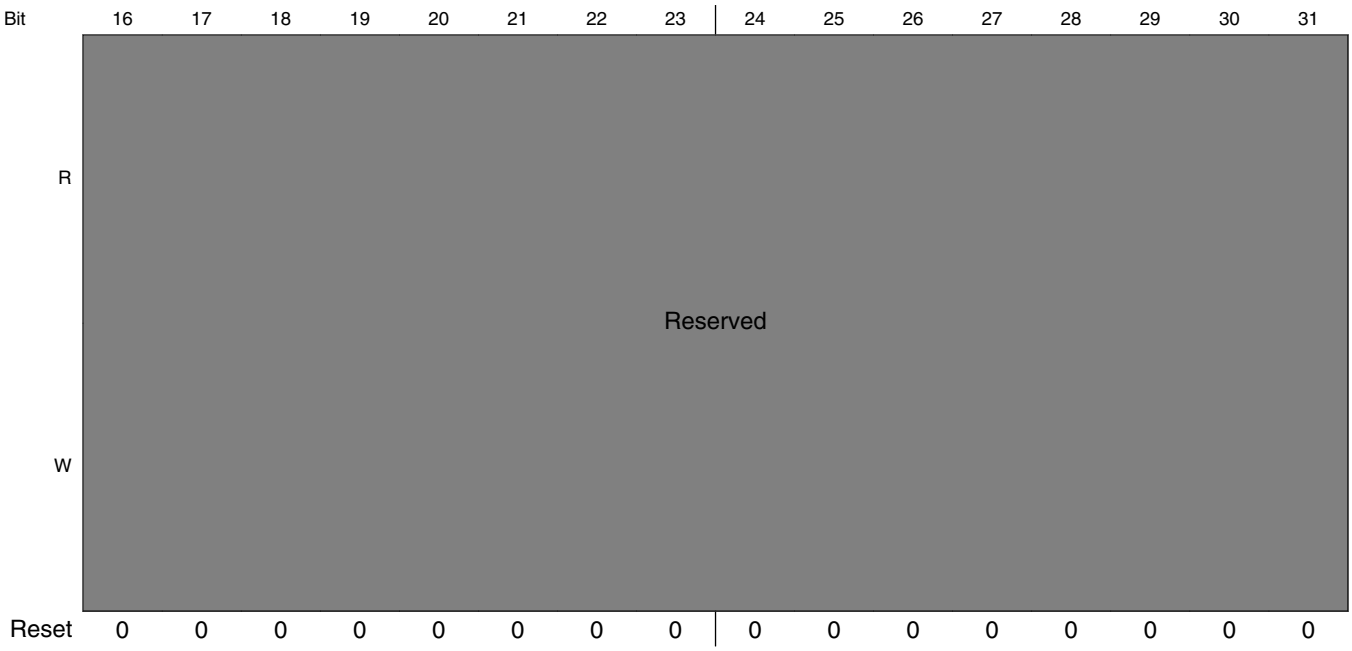
Field	Description
	0 No error 1 Error, the CMD index in response is not CMD12
28 ACEBE	Auto CMD end bit error. Occurs when detecting that the end bit of command response is 0 which should be 1. 0 No error 1 End bit error generated
29 ACCE	Auto CMD CRC error. Occurs when detecting a CRC error in the command response. 0 No CRC error 1 CRC error met in Auto CMD12 response
30 ACTOE	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. 0 No error 1 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. 0 Executed 1 Not executed

16.3.18 Host controller capabilities register (eSDHC_HOSTCAPBLT)

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.

Address: 11_4000h base + 40h offset = 11_4040h





eSDHC_HOSTCAPBLT field descriptions

Field	Description
0–1 -	This field is reserved.
2 AIS	Asynchronous Interrupt Support. This bit indicates whether the eSDHC supports SDIO asynchronous interrupt. Refer to SDIO Specification Version 3.0 0 Asynchronous interrupt not supported 1 Asynchronous interrupt supported
3 64BSBS	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. 0 64-bit system bus not supported 1 64-bit system bus supported
4 -	This field is reserved.
5 VS18	Voltage support 1.8V. This bit should depend on the host system ability. 0 1.8V not supported 1 1.8V supported
6 VS30	Voltage Support 3.0V. This bit shall depend on the Host System ability. 0 3.0V not supported 1 3.0V supported
7 VS33	Voltage support 3.3V. This bit should depend on the host system ability. 0 3.3V not supported 1 3.3V supported

Table continues on the next page...

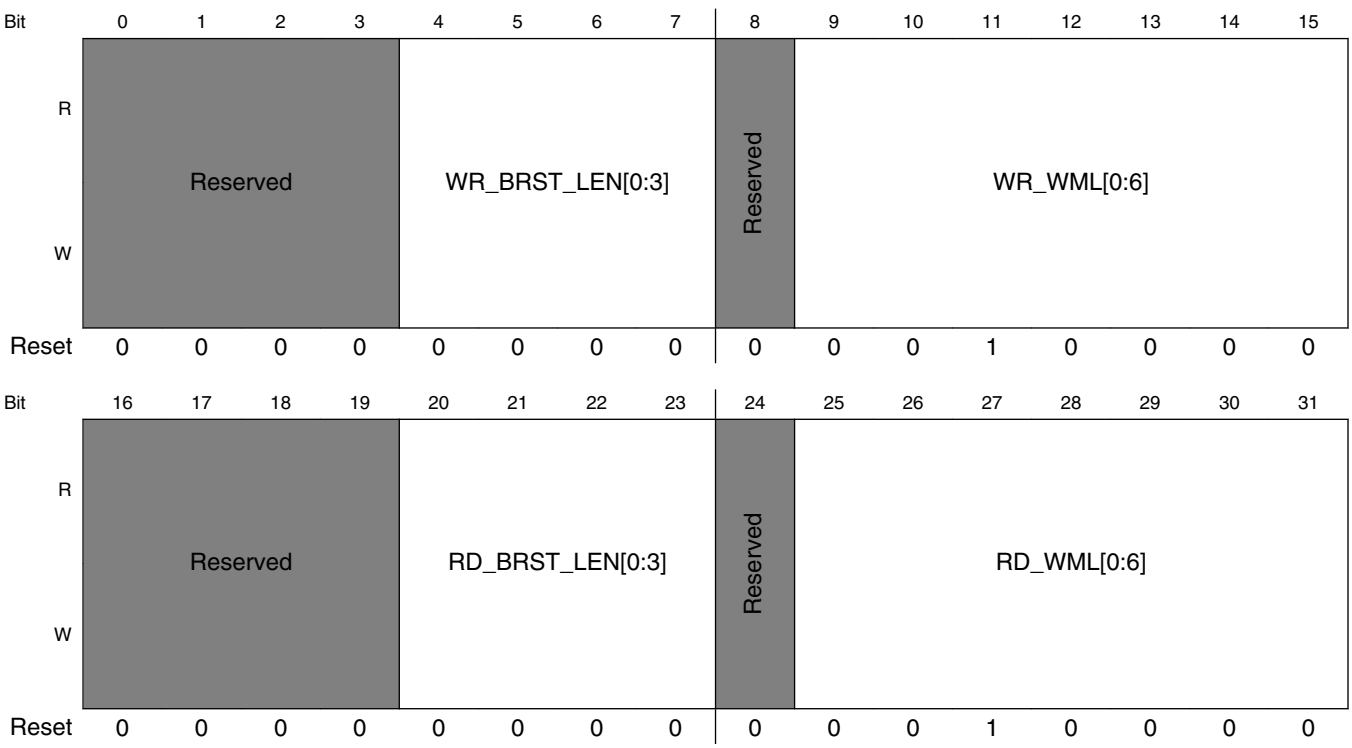
eSDHC_HOSTCAPBLT field descriptions (continued)

Field	Description
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. 0 Not supported 1 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. 0 DMA not supported 1 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. 0 High speed not supported 1 High speed supported
11 ADMAS	ADMA support. This bit indicates whether the eSDHC supports the ADMA feature. 0 Advanced DMA not supported 1 Advanced DMA supported
12 -	This field is 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. 000 512 bytes 001 1024 bytes 010 2048 bytes 011-111 Reserved
16–31 -	This field is reserved.

16.3.19 Watermark level register (eSDHC_WML)

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.

Address: 11_4000h base + 44h offset = 11_4044h



eSDHC_WML field descriptions

Field	Description
0–3 -	This field is reserved.
4–7 WR_BRST_LEN[0:3]	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 0x0 16 transfers in a single burst 0x1 1 transfer in a single burst 0x2 2 transfers in a single burst ... 0xF 15 transfers in a single burst
8 -	This field is reserved.

Table continues on the next page...

eSDHC_WML field descriptions (continued)

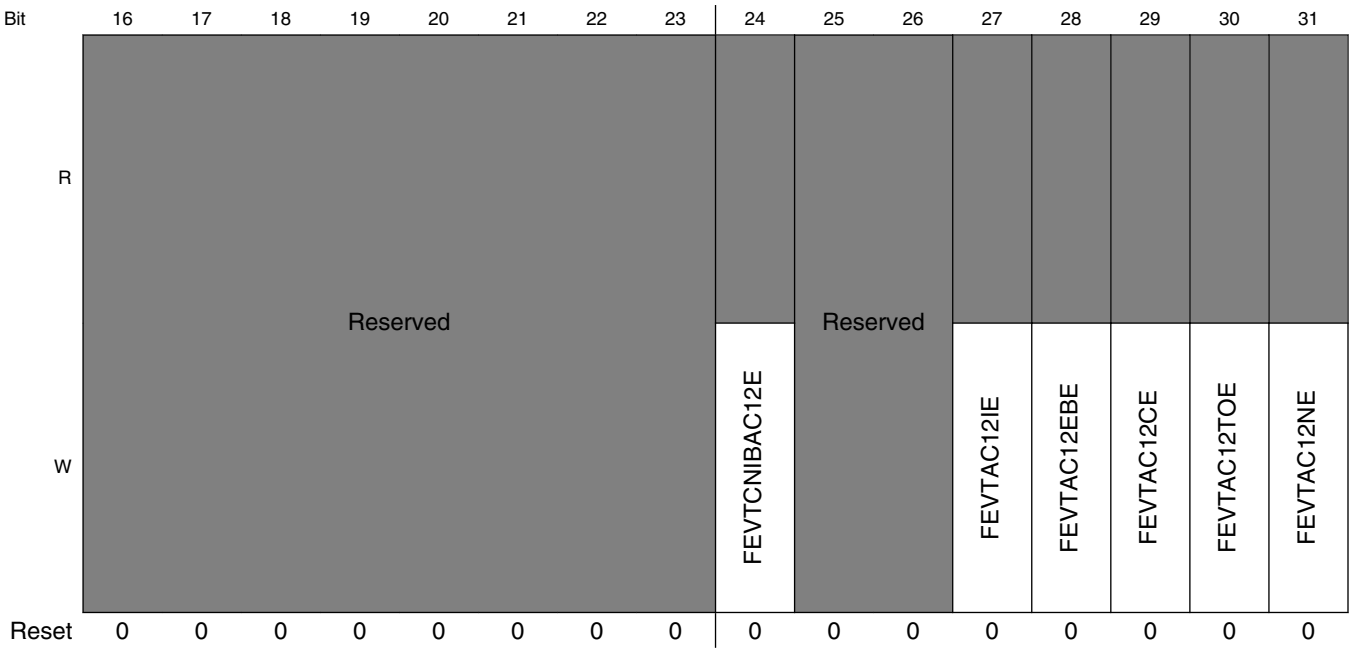
Field	Description
9–15 WR_WML[0:6]	Write watermark level. The number of words (32-bit) used as the watermark level (FIFO threshold) for SD write in CPU polling mode. 0x00 128 words 0x01 1 word 0x02 2 words 0x7F 127 words
16–19 -	This field is reserved.
20–23 RD_BRST_ LEN[0:3]	Max read burst length. Burst 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 0x0 16 transfers in a single burst 0x1 1 transfers in a single burst 0x2 2 transfers in a single burst 0xF 15 transfers in a single burst
24 -	This field is reserved.
25–31 RD_WML[0:6]	Read watermark level. The number of words (32-bit) used as the watermark level (FIFO threshold) for SD read in CPU polling mode. 0x00 128 words 0x01 1 word 0x02 2 words 0x7F 127 words

16.3.20 Force event register (eSDHC_FEVT)

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.

Address: 11_4000h base + 50h offset = 11_4050h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved				Reserved				Reserved							
W				FEVTDMAE			FEVTADMAE	FEVTAC12E		FEVTDEBE	FEVTDCE	FEVTDTOE	FEVTCIE	FEVTCEBE	FEVTCCE	FEVTCTOE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



eSDHC_FEVT field descriptions

Field	Description
0–2 -	This field is reserved.
3 FEVTDMAE	Force event DMA error. Forces the IRQSTAT[DMAE] to be set.
4–5 -	This field is reserved.
6 FEVTADMAE	Force event ADMA error. Forces the IRQSTAT[ADMAE] to be set.
7 FEVTAC12E	Force event Auto CMD12 error. Forces IRQSTAT[AC12E] to be set.
8 -	This field is 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.

Table continues on the next page...

eSDHC_FEVT field descriptions (continued)

Field	Description
16–23 -	This field is reserved.
24 FEVTCNIBAC12E	Force event command not executed by Auto CMD12 error. Forces AUTOC12ERR[CNIBAC12E] to be set.
25–26 -	This field is 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.

16.3.21 ADMA system address low register (eSDHC_ADSADDRL)

The ADSADDRL contains the physical system memory address used for ADMA transfers.

Address: 11_4000h base + 58h offset = 11_4058h

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	ADS_ADDR_LOW[0:31]																															
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

eSDHC_ADSADDRL field descriptions

Field	Description
0–31 ADS_ADDR_ LOW[0:31]	ADMA system address low. This register holds lower 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.

16.3.22 ADMA system address high register (eSDHC_ADSADDRH)

The ADSADDRH contains the physical system memory address used for ADMA transfers.

Address: 11_4000h base + 5Ch offset = 11_405Ch

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	ADS_ADDR_HIGH[0:31]																															
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

eSDHC_ADSADDRH field descriptions

Field	Description
0–31 ADS_ADDR_HIGH[0:31]	ADMA system address high. This register holds higher 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. 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.

16.3.23 Host controller version register (eSDHC_HOSTVER)

The HOSTVER contains the vendor host controller version information. All bits are read only and will read the same as the power-reset value.

Address: 11_4000h base + FCh offset = 11_40FCh

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																VVN[0:7]								SVN[0:7]							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0

eSDHC_HOSTVER field descriptions

Field	Description
0–15 -	This field is reserved.
16–23 VVN[0:7]	Vendor version number. These status bits are reserved for the vendor version number. The host driver should not use this status. Others) Reserved except

Table continues on the next page...

eSDHC_HOSTVER field descriptions (continued)

Field	Description
	0x00 Freescale eSDHC Version 1.0 0x10 Freescale eSDHC Version 2.0 0x11 Freescale eSDHC Version 2.1 0x12 Freescale eSDHC Version 2.2 0x13 Freescale eSDHC Version 2.3 0x20 Freescale eSDHC Version 3.0
24–31 SVN[0:7]	Specification version number. These status bits indicate the host controller specification version. Patterns not shown are reserved. 0x00 SD Host Specification Version 1.0 0x01 SD Host Specification Version 2.0, supports test event register . and ADMA 0x02 SD Host Specification Version 3.0

16.3.24 DMA error address register (eSDHC_DMAERRADDR)

The DMAERRADDR contains the address of the transaction on which DMA error occured.

Address: 11_4000h base + 104h offset = 11_4104h

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	DMA_ADDRn																															
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

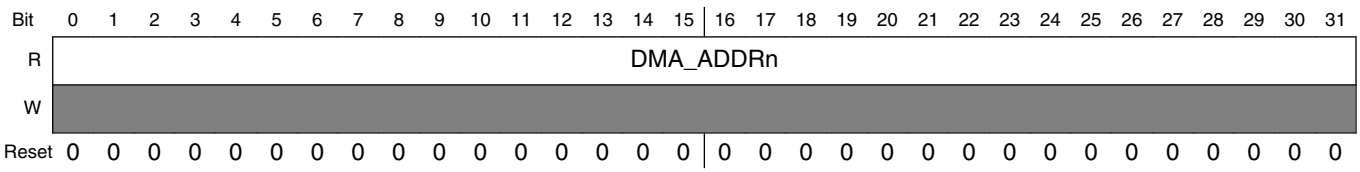
eSDHC_DMAERRADDR field descriptions

Field	Description
0–31 DMA_ADDRn	DMA error address. This field contains the system address of the transaction on which DMA error occured.

16.3.25 DMA error address low register (eSDHC_DMAERRADDRL)

The DMAERRADDRL contains the address of the transaction on which DMA error occured.

Address: 11_4000h base + 104h offset = 11_4104h



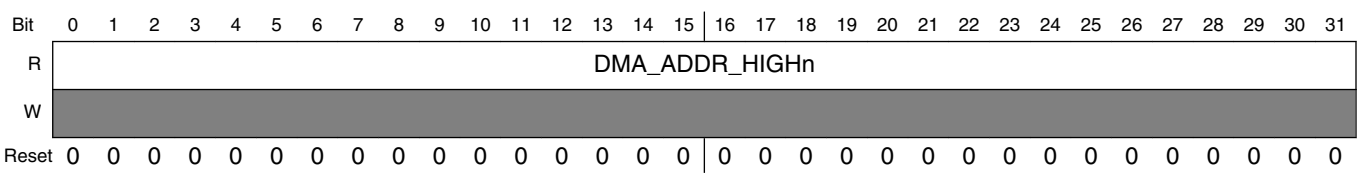
eSDHC_DMAERRADDRL field descriptions

Field	Description
0–31 DMA_ADDRn	DMA error address. This field contains the system low address of the transaction on which DMA error occured.

16.3.26 DMA error address high register (eSDHC_DMAERRADDRH)

The DMAERRADDRH contains the address of the transaction on which DMA error occured.

Address: 11_4000h base + 108h offset = 11_4108h



eSDHC_DMAERRADDRH field descriptions

Field	Description
0–31 DMA_ADDR_ HIGHn	DMA error address. This field contains the system high address of the transaction on which DMA error occured.

16.3.27 DMA error attribute register (eSDHC_DMAERRATTR)

The DMAERRATTR contains attributes of the transaction on which DMA error occured.

Address: 11_4000h base + 10Ch offset = 11_410Ch

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	Reserved									DMA_SIZE			DMA_LEN			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

eSDHC_DMAERRATTR field descriptions

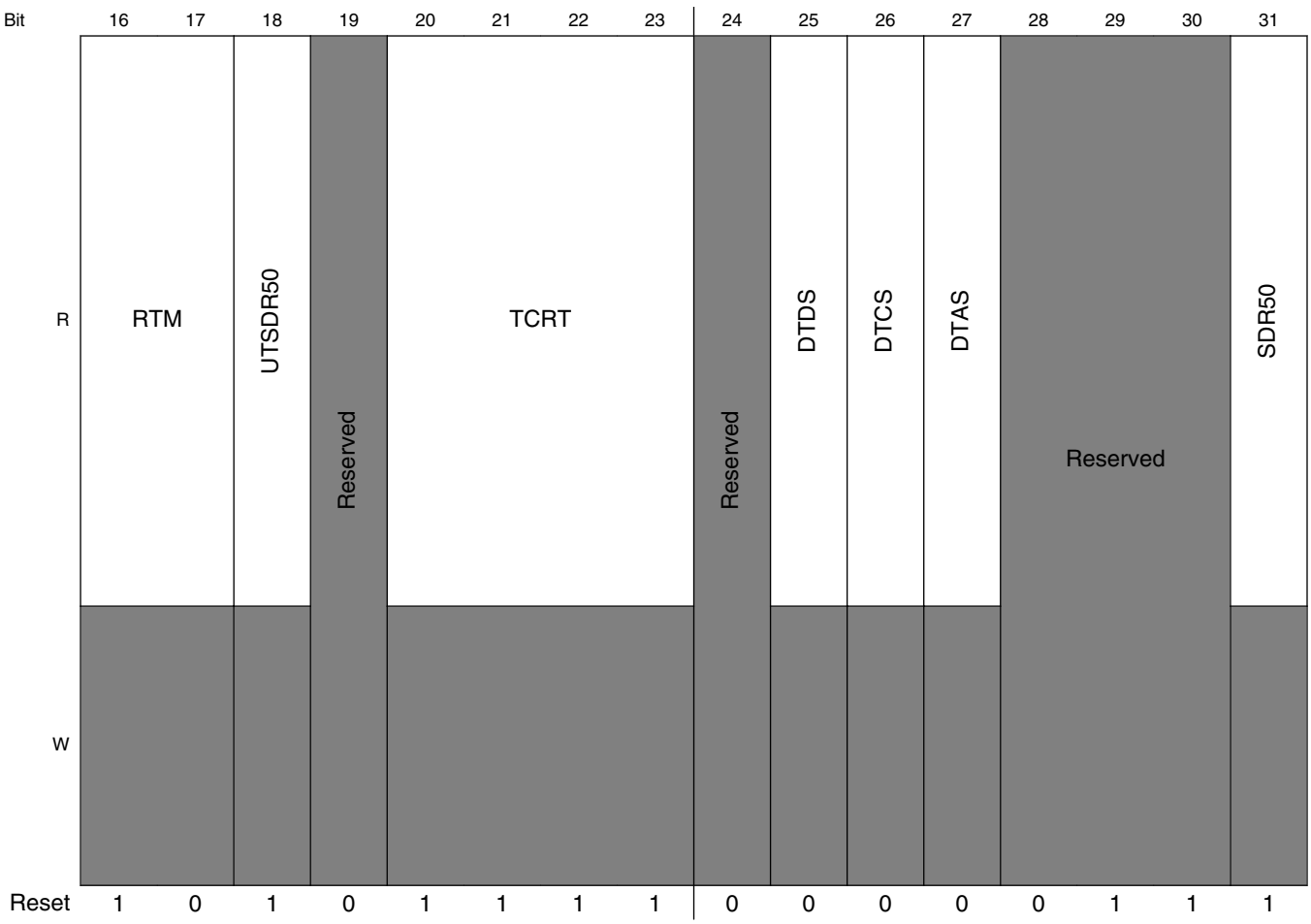
Field	Description
0–24 -	This field is reserved.
25–27 DMA_SIZE	System bus burst size. This field contains burst size of the transaction on which DMA error occured.
28–31 DMA_LEN	System bus burst length. This field contains burst length of the transaction on which DMA error occured.

16.3.28 Host controller capabilities register 2 (eSDHC_HOSTCAPBLT2)

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.

Address: 11_4000h base + 114h offset = 11_4114h

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



eSDHC_HOSTCAPBLT2 field descriptions

Field	Description
0–15 -	This field is reserved.
16–17 RTM	Re-tuning modes. This bit indicates the supported re-tuning modes for UHS. 00 Mode 1 - Software Timer 01 Mode 2 - Software Timer and Re-tuning request 10 Mode 3 - Software Timer, and Auto Re-tuning during data transfer 11 Reserved
18 UTSDR50	Use tuning for SDR50. This bit indicates whether the host support tuning for SDR50 mode. 0 Tuning for SDR50 mode not supported 1 Tuning for SDR50 mode supported
19 -	This field is 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. 1111 Get timer information from other source.

Table continues on the next page...

eSDHC_HOSTCAPBLT2 field descriptions (continued)

Field	Description
	1110-1100 Reserved 1011 1024 seconds ... 0011 4 seconds 0010 2 seconds 0001 1 second 0000 Re-Tuning timer disabled
24 -	This field is reserved.
25 DTDS	Driver type D support. This bit indicates whether the system is capable of using driver type D. 0 Driver Type D not supported 1 Driver Type D Supported
26 DTCS	Driver type C support. This bit indicates whether the system is capable of using driver type C. 0 Driver Type C not supported 1 Driver Type C Supported
27 DTAS	Driver type A support. This bit indicates whether the system is capable of using driver type A. 0 Driver Type A not supported 1 Driver Type A Supported
28–30 -	This field is reserved.
31 SDR50	SDR50 support. This bit indicates whether the eSDHC supports the SDR50. 0 SDR50 not supported 1 SDR50 supported

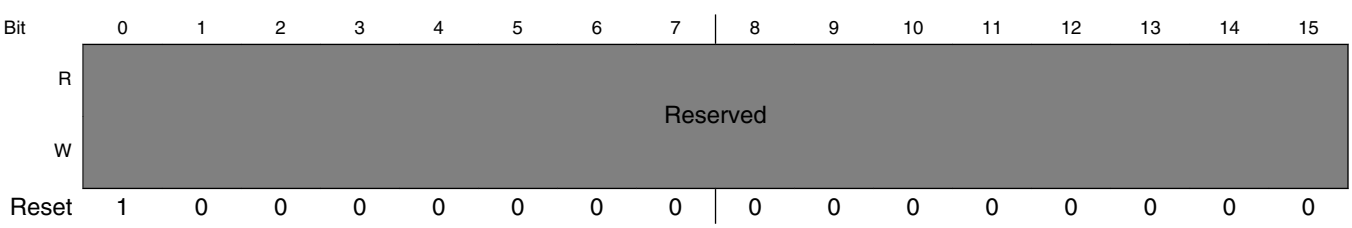
16.3.29 Tuning control register (eSDHC_TCR)

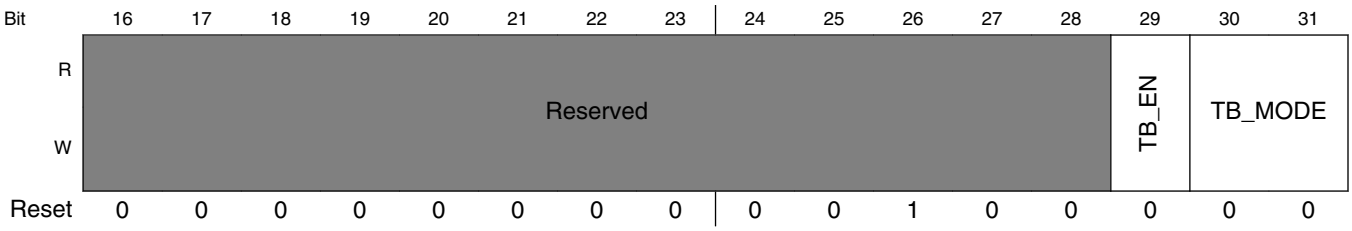
This register contains fields for controlling tuning block.

NOTE

Write or read to reserved fields of this register does not guarantee specific value to be written or read.

Address: 11_4000h base + 120h offset = 11_4120h





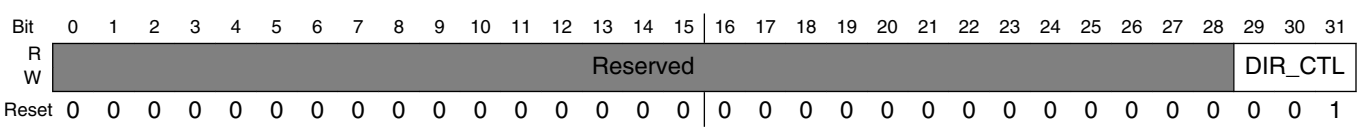
eSDHC_TCR field descriptions

Field	Description
0–28 -	This field is 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. 0 Tuning block is disabled 1 Tuning block is enabled
30–31 TB_MODE	Tuning Mode. Selects tuning mode when tuning block is enabled. Refer to re-tuning modes in Table 16-41 00 Mode 1 - Software Timer 01 Mode 2 - Software Timer, and Re-tuning request 10 Mode 3 - Software Timer, and Auto Re-tuning during data transfer 11 Reserved

16.3.30 SD direction control register (eSDHC_SD_DIRCTL)

This register contains control for CMD and DAT lines direction.

Address: 11_4000h base + 140h offset = 11_4140h



eSDHC_SD_DIRCTL field descriptions

Field	Description
0–28 -	This field is 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 3'h7 - Seven SD clock periods for turnaround ..

Table continues on the next page...

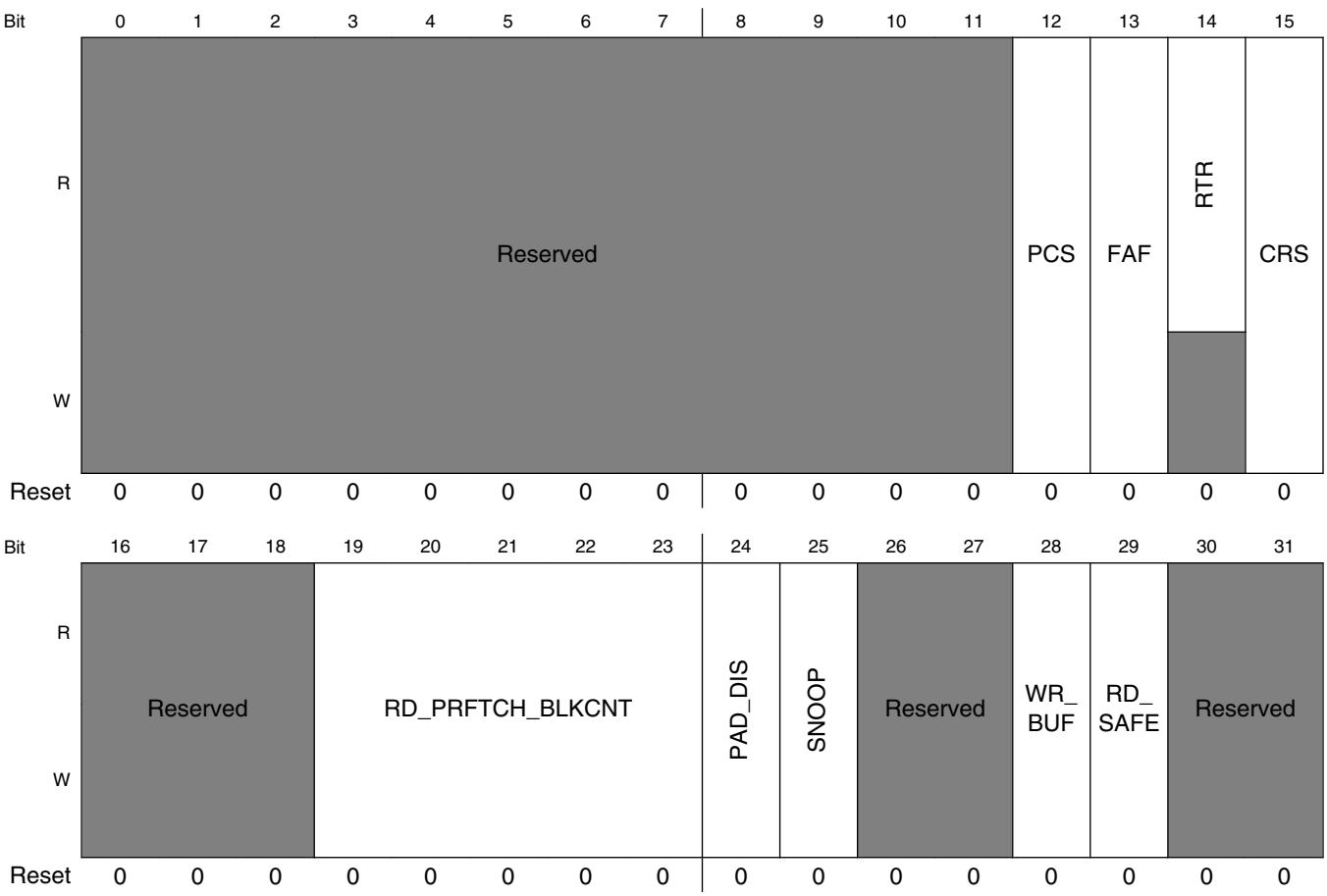
eSDHC_SD_DIRCTL field descriptions (continued)

Field	Description
	3'h2 - Two SD clock periods for turnaround 3'h1 - One SD clock period for turnaround 3'h0 - No turnaround time required

16.3.31 eSDHC control register (eSDHC_ESDHCCTL)

This register contains fields for controlling DMA transfers.

Address: 11_4000h base + 40Ch offset = 11_440Ch



eSDHC_ESDHCCTL field descriptions

Field	Description
0–11 -	This field is reserved.

Table continues on the next page...

eSDHC_ESDHCCTL field descriptions (continued)

Field	Description
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. 0 Platform clock is used 1 Peripheral 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. 0 No Flush 1 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 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. 0 No re-tuning request 1 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. 0 SDCLKFS is defined as 8-bit field, and DVS is active in system control register 1 SDCLKFS is defined as 10-bit field, CGS is active in system control register
16–18 -	This field is 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: 00 No prefetch 01 1 SD block prefetch 02 2 SD block prefetch ... 1F 31 SD block prefetch
24 PAD_DIS	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. 0 DMA will pad data at the end each block transfer. 1 DMA will not pad data at the end of each block transfer.
25 SNOOP	Snoop attribute. Snoop enable for DMA transaction. 0 DMA transactions are not snooped by the CPU data cache. 1 DMA transactions are snooped by the CPU data cache.
26–27 -	This field is 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.

Table continues on the next page...

eSDHC_ESDHCCTL field descriptions (continued)

Field	Description
	0 Non-last write transactions are not bufferable. 1 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. 0 It is not safe to read more bytes that were intended. 1 It is safe to read more bytes that were intended.
30–31 -	This field is reserved.

16.4 Functional description

The eSDHC block is partitioned in five major sub-blocks as shown in Figure 16-2.

- 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.

16.4.1 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.

16.4.1.1 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.

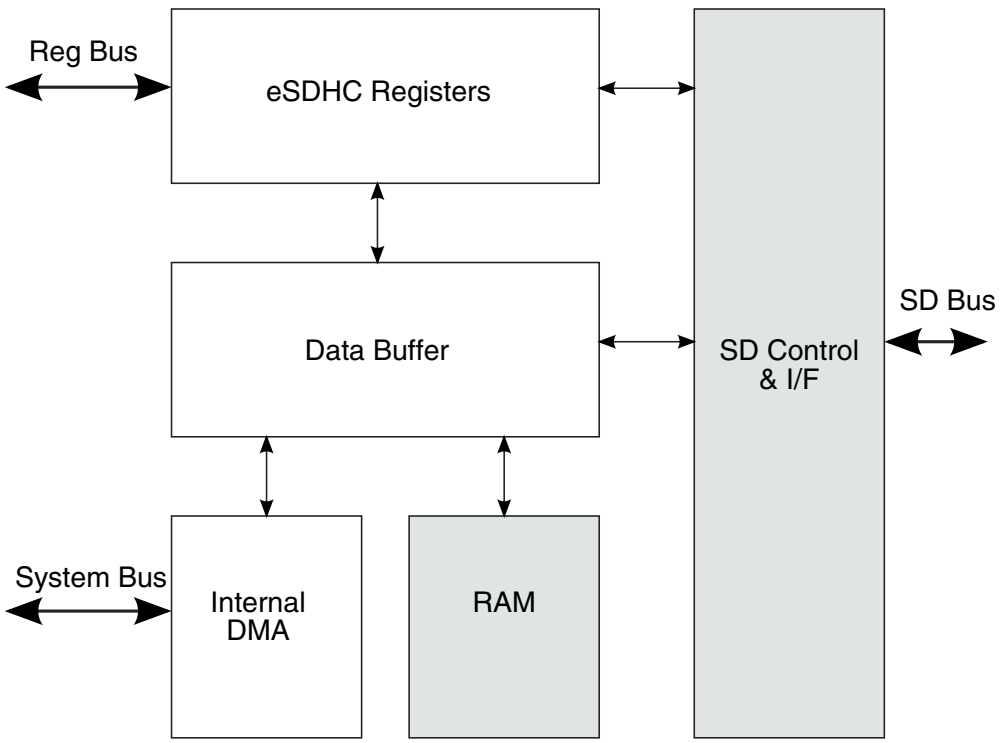


Figure 16-34. eSDHC buffer scheme

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.

16.4.1.1.1 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.

16.4.1.1.2 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.

16.4.1.1.3 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.

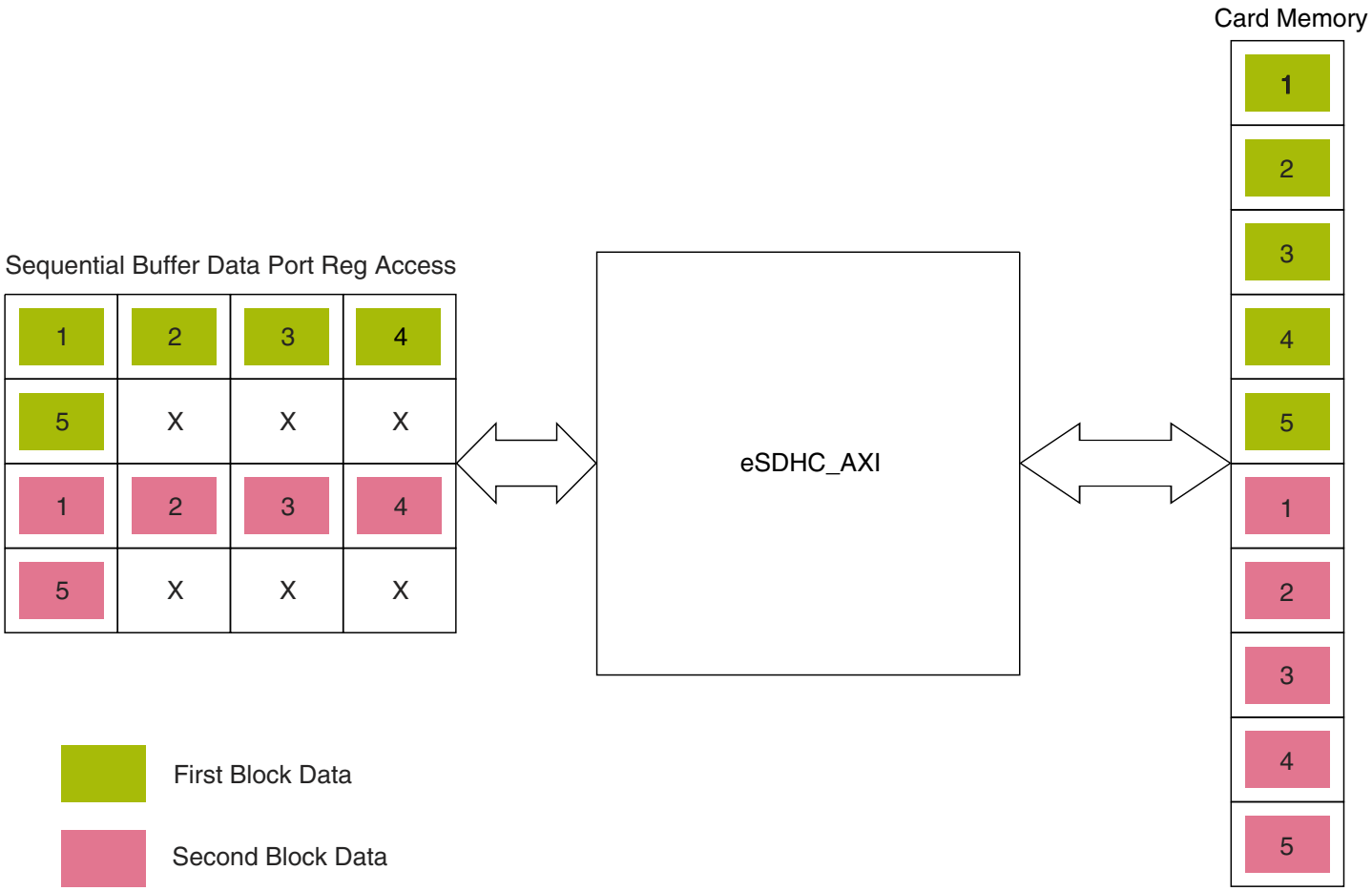


Figure 16-35. Byte stuffing for CPU polling mode

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 16-36. And, data transfer with byte stuffing disabled (when DMACTL[PAD_DIS]=1) is shown in Figure 16-37. Transfer with byte stuffing disabled eliminates the software overhead of byte stuffing. Driver may program DMA Control register accordingly.

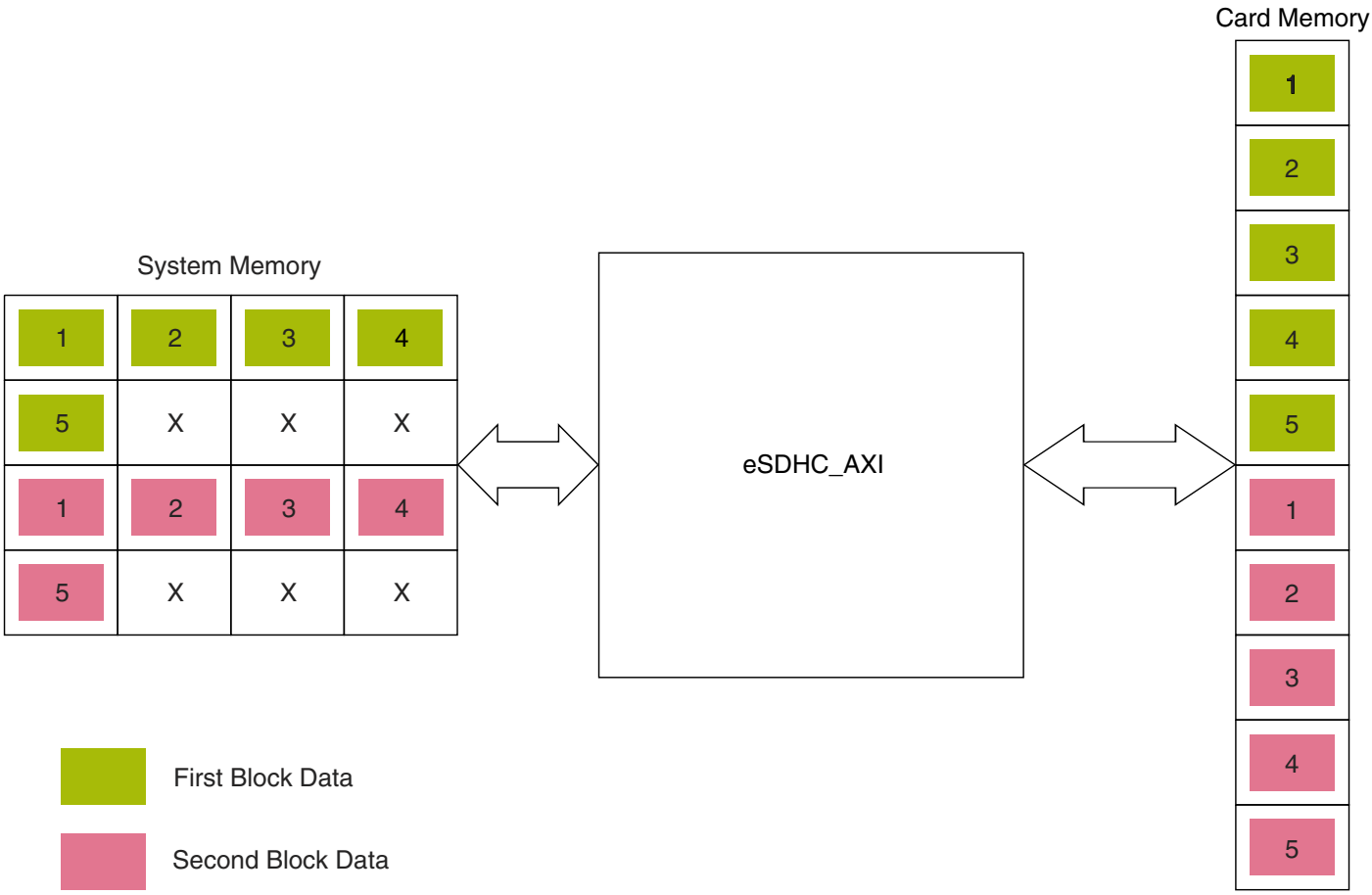


Figure 16-36. Byte stuffing for DMA mode when DMACTL[PAD_DIS]=0

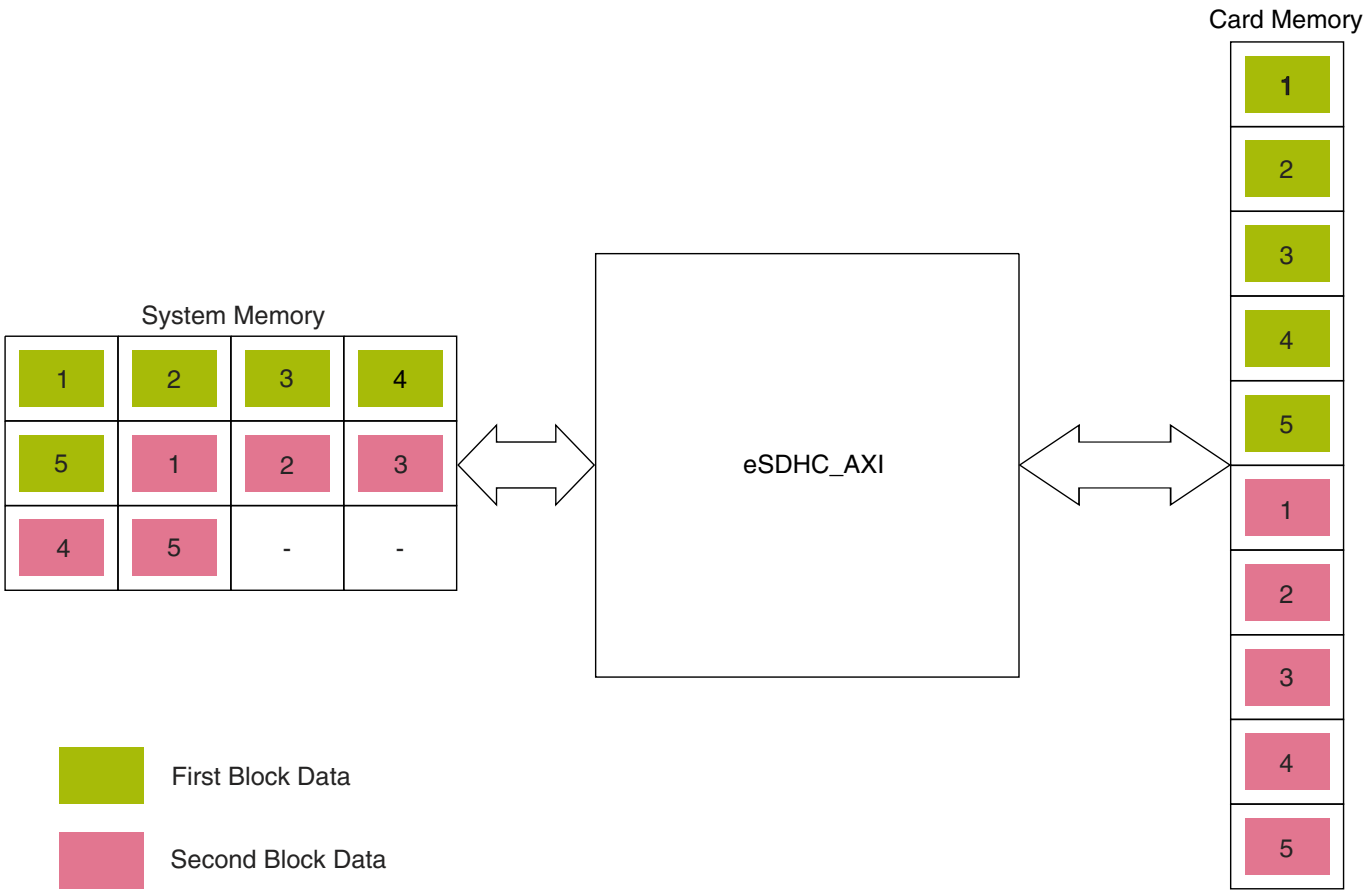


Figure 16-37. No byte stuffing for DMA mode when DMACTL[PAD_DIS]=1

16.4.1.1.4 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).

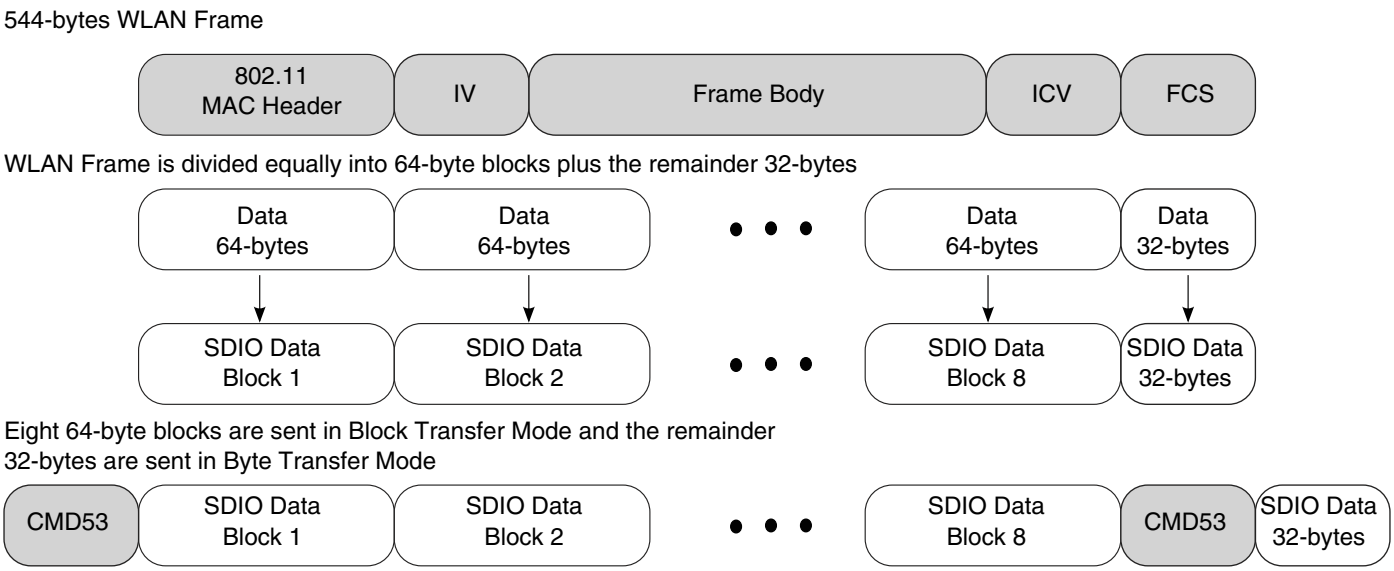


Figure 16-38. Example for dividing large data transfers

16.4.1.1.5 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.

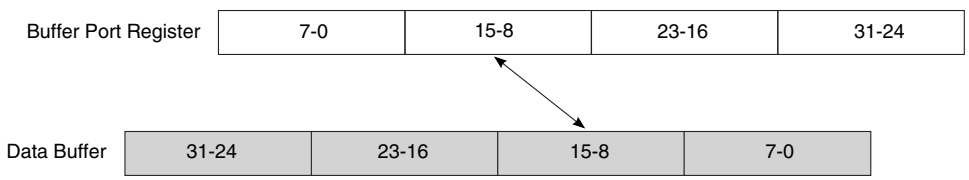


Figure 16-39. Data swap between buffer data port register and data buffer in big endian mode

16.4.1.2 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.

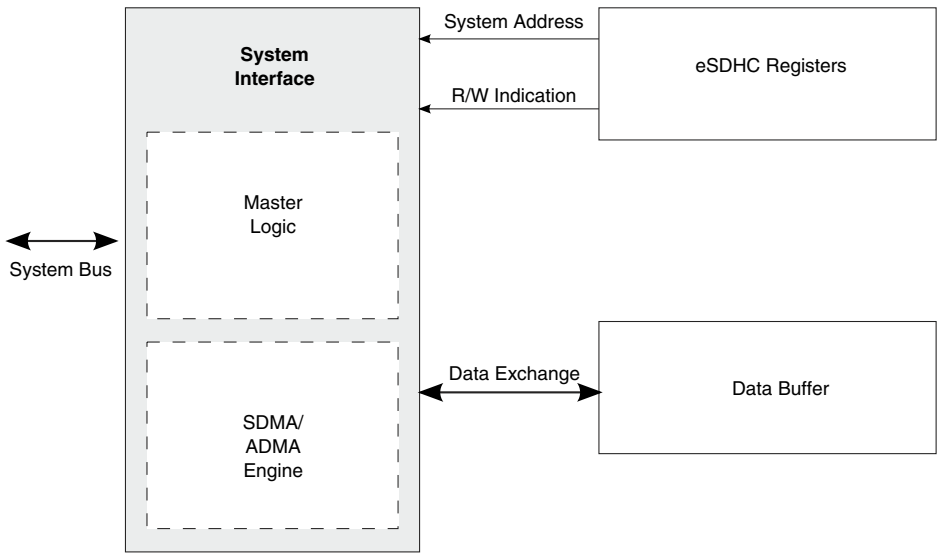


Figure 16-40. DMA system interface block

16.4.1.2.1 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)

16.4.1.2.2 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 DMAERRADDRH/L[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.

16.4.1.3 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.

16.4.1.3.1 SDMA error

SDMA will stop whenever an error is encountered on the system bus.

DMA error address low register latches the transaction address on which the error occured.

16.4.1.4 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 both 32-bit and 64-bit addressing.

16.4.1.4.1 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 16-41 explains the ADMA1 descriptor table format.

Figure 16-43 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 16-41. Format of the 32-bit address ADMA1 descriptor table

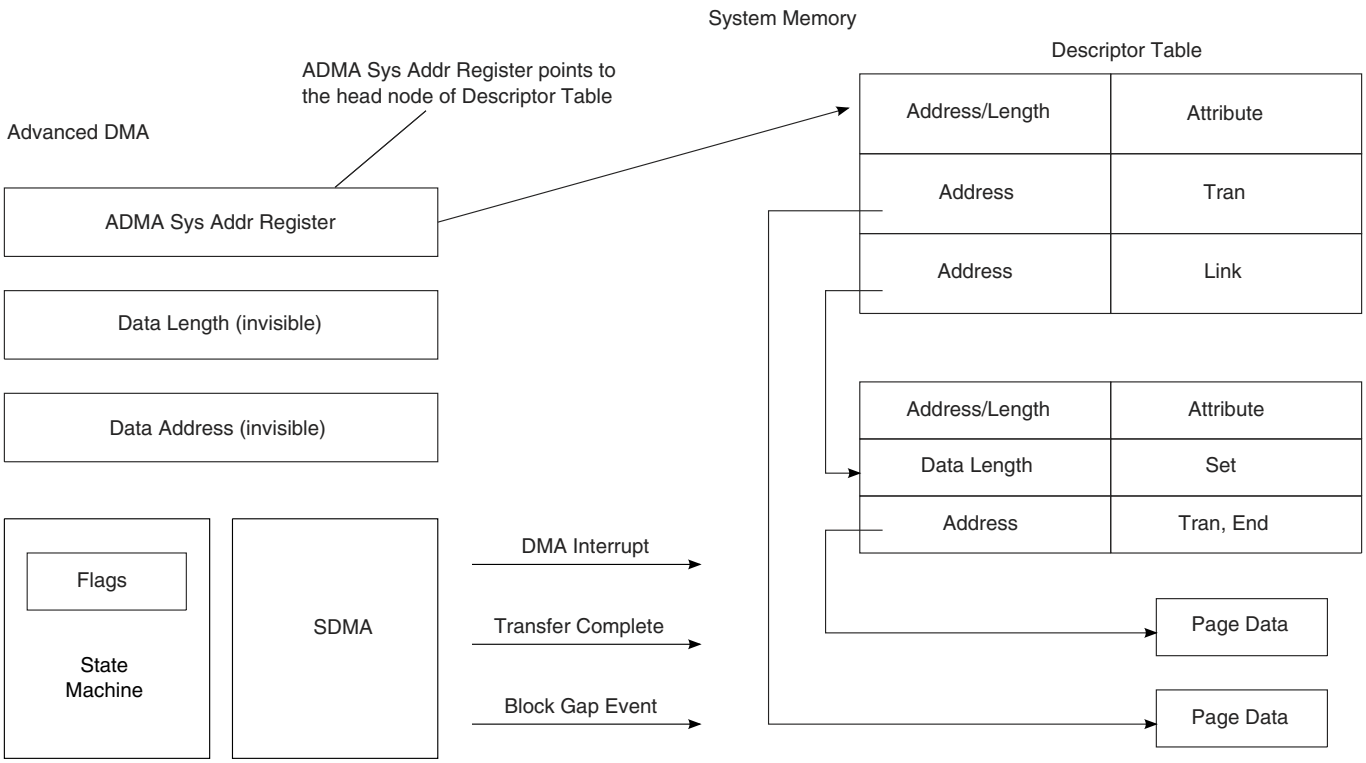


Figure 16-42. Concept and access method of ADMA1 descriptor table

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 16-43. Format of the 32-bit address ADMA2 descriptor table

Table 16-40. Format of the 64-bit address ADMA2 descriptor table

Address field		Length		Reserved			Attribute field				
95	32	31	16	15	06	05	04	03	02	01	00
64-bit Address		16-bit length		0000000000		Act2	Act1	0	Int	End	Valid

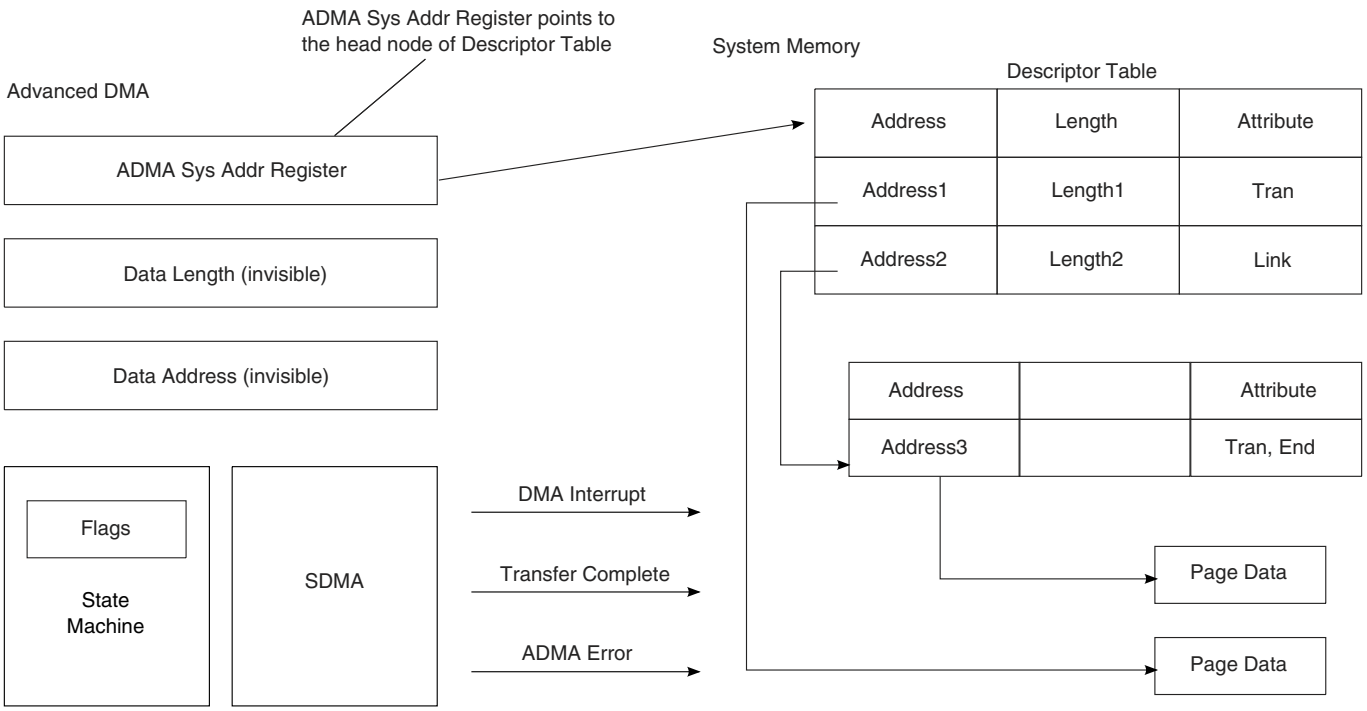


Figure 16-44. Concept and access method of ADMA2 descriptor table

16.4.1.4.2 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.

16.4.1.4.3 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 low/high register (DMAERRADDRL/H) latches the transaction address on which the error occured. The ADMA system address low/high register (ADSADDRL/H) latches the current descriptor line address which encountered the error.

16.4.2 SD interface and control unit (SDICU)

This block is partitioned into six major sub-blocks as shown in Figure 16-2.

- 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.

16.4.2.1 Command CRC

See the figure below for an illustration of the structure for the command CRC shift register.

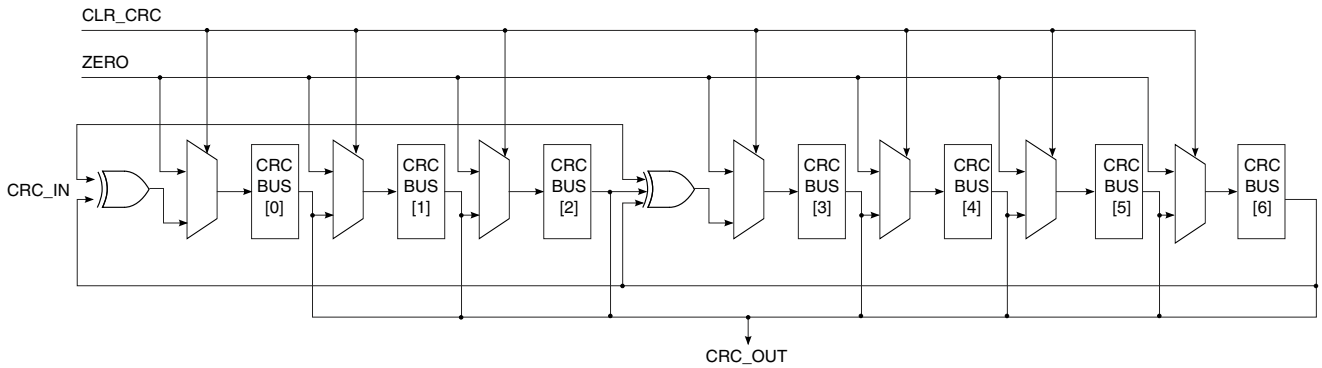


Figure 16-45. Command CRC shift register

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)]
```

16.4.2.2 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)]
```

16.4.2.3 Tuning block (SDICU)

Tuning block is used for SD UHS SDR50 mode.

Tuning block tunes the IP on tuning command (CMD19 for SD, CMD21 for MMC) 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 16-41. 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.

16.4.3 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.

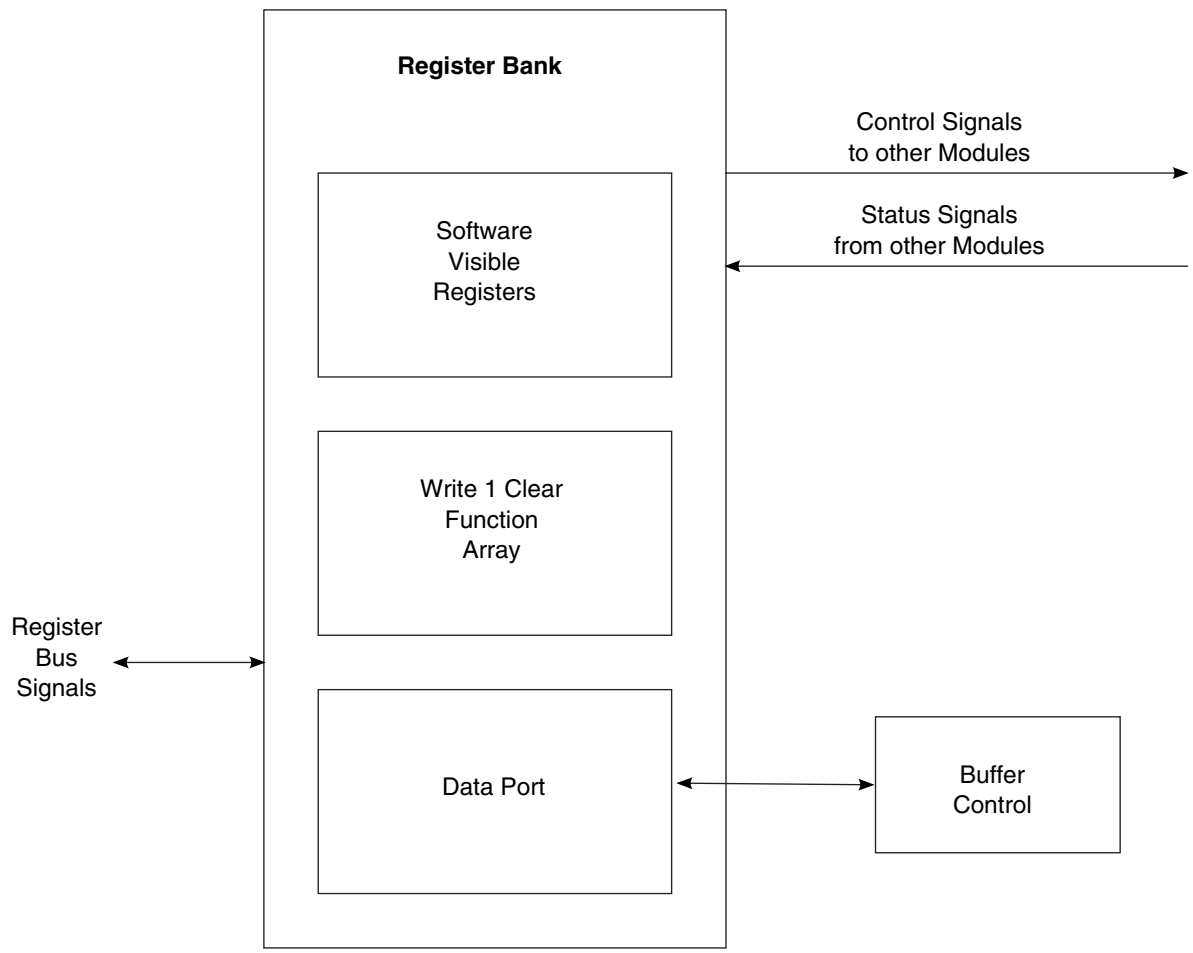


Figure 16-46. Register bank diagram

Partial access is not allowed on any register; that is, it should be 32-bit access only.

16.4.4 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.

16.4.4.1 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.

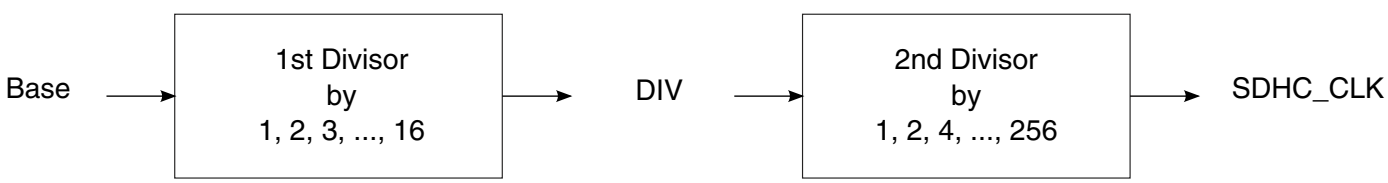


Figure 16-47. Two stages of the clock divider

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 16-2) 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.

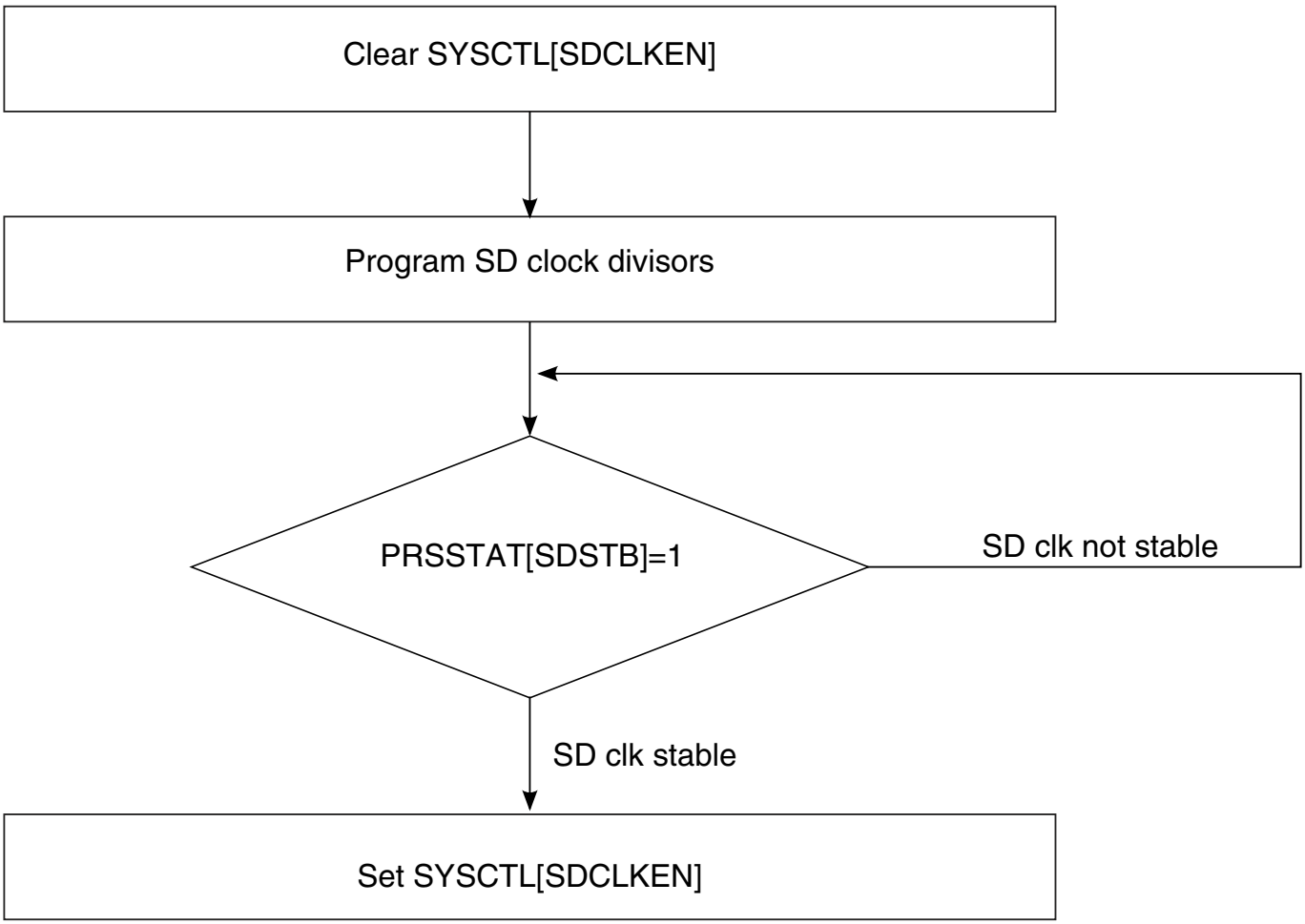


Figure 16-48. SD clock frequency change sequence

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.

The steps for configuring base clock changes are:

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

16.4.5 SD monitor

The module detects the CD_B (card detection) signal as well as the DAT3 signal. The transceiver reports the card insertion state according to the CD_B state, or 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.

16.4.5.1 SDIO card interrupt

16.4.5.1.1 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.

16.4.5.1.2 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.

16.4.5.1.3 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.

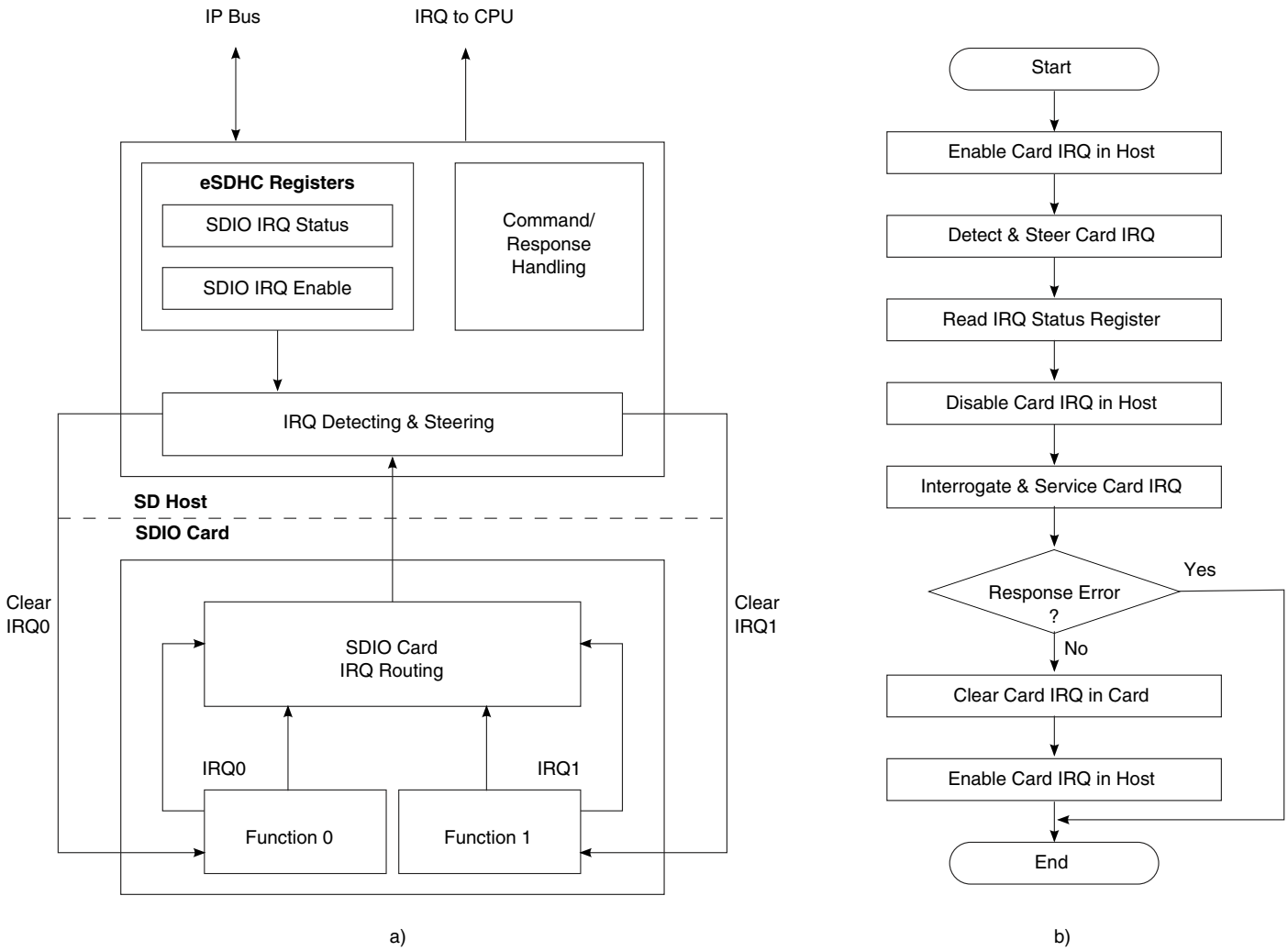


Figure 16-49. Card interrupt scheme and card interrupt detection and handling procedure

16.4.5.2 Card insertion and removal detection

The eSDHC usesthe CD pin to detect card insertion or removal.

The CD pin is always a reference for card detection. When 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.

16.4.5.3 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 (eSDHC_PROCTL) for more information on the eSDHC protocol control register.

16.4.5.3.1 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

16.5 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 MMC/SD/SDIO for the commands of bc and bcr categories.

After the Broadcast command CMD3 is issued, the cards enter standby mode. Addressed type commands are used from this point. In this mode, the CMD/DAT I/O pads will turn to push-pull mode, to have the driving capability for maximum frequency operation. Refer to Commands for MMC/SD/SDIO, for the commands of ac and adtc categories.

16.5.1 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.

16.5.2 Card identification mode

When a card is inserted to the socket or the card was reset by the host, the host needs to validate the operation voltage range, identify the cards, request the cards to publish the relative card address (RCA) or set the RCA for the MMC cards.

All data communications in the card identification mode use the command line (CMD) only.

16.5.2.1 Card detect

Figure below illustrates a flow diagram showing the detection of MMC, SDIO, and SD cards using the eSDHC.

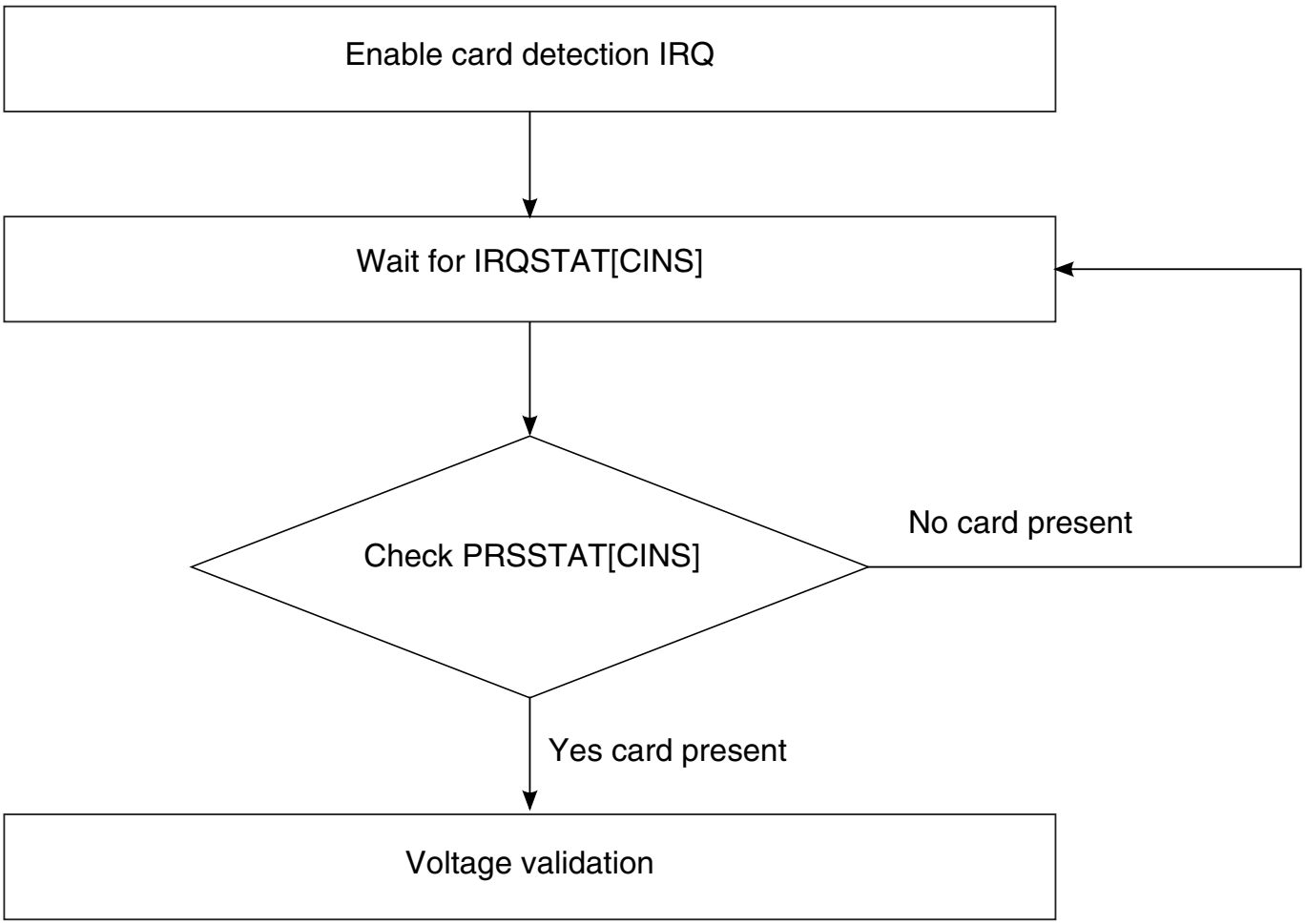


Figure 16-50. Flow diagram for card detection

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.

16.5.2.2 Reset

The host consists of three types of resets:

- Hardware reset (card and host) which is driven by POR (power on reset).

- Software reset (Host Only) is 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 MMC cards 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.

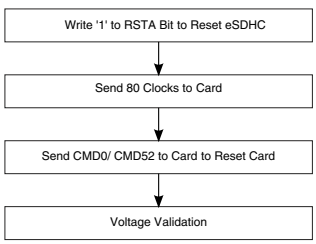


Figure 16-51. Flow chart for reset of the eSDHC and SD I/O card

```
software_reset()
{
set_bit(SYSCTRL, RSTA); // software reset the Host
set DTOCV and SDCLKFS bit fields to get the SDHC_CLK of frequency around 400kHz
configure IO pad to set the power voltage of external card to around 3.0V
poll bits CIHB and CDIHB bits of PRSSTAT to wait both bits are cleared
set_bit(SYSCTRL, INTIA); // send 80 clock ticks for card to power up
send_command(CMD_GO_IDLE_STATE, <other parameters>); // reset the card with CMD0
or send_command(CMD_IO_RW_DIRECT, <other parameters>);
}
```

16.5.2.3 Voltage validation

All cards should be able to establish communication with the host using any operation voltage in the maximum allowed voltage range specified in the card specification.

However, the supported minimum and maximum values for Vdd are defined in the operation conditions register (OCR) and may not cover the whole range. Cards that store the CID and CSD data in the preload memory are only able to communicate this information under data transfer Vdd conditions. This means, if the host and card have non-common Vdd ranges, the card will not be able to complete the identification cycle, nor will it be able to send CSD data.

Therefore, a special command Send_Op_Cont (CMD1 for MMC), SD_Send_Op_Cont (ACMD41 for SD Memory) and IO_Send_Op_Cont (CMD5 for SD I/O) is used. The voltage validation procedure is designed to provide a mechanism to identify and reject cards which do not match the Vdd range(s) desired by the host. This is accomplished by the host sending the desired Vdd voltage window as the operand of this command. Cards that 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 MMC card
          if (card is already labelled as SDCombo) { // change label
                    re-label the card as SDIO;
                    ignore the error or report it;
                    return; // card is identified as SDIO card
          } // of if (card is ...
          send_command(SEND_OP_COND, <voltage range>, <...>);
          if (RESP_TIMEOUT == wait_for_response(SEND_OP_COND)) { // CMD1 is not accepted,
```

```
either
                    label the card as UNKNOWN;
                    return;
          } // of if (RESP_TIMEOUT ...
          else label the card as MMC;
} // of else
}
```

16.5.2.4 Card registry

Card registry for the MMC and SD/SDIO/SD combo cards are different.

For the SD card, the identification process starts at a clock rate lower than 400 kHz and the power voltage higher than 2.7 V (as defined by the card 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 MMC operation, the host starts the card identification process in open-drain mode with the identification clock rate lower than 400 kHz and the power voltage higher than 2.7 V. The open drain driver stages on the CMD line allow parallel card operation during card identification. After the bus is activated the host will request the cards to send their valid operation conditions (CMD1). The response to CMD1 is the "wired OR" operation on the condition restrictions of all cards in the system. Incompatible cards are sent into the Inactive State. The host then issues the broadcast command All_Send_CID (CMD2), asking all cards for their unique card identification (CID) number. All unidentified cards (the cards in Ready State) simultaneously start sending their CID numbers serially, while bit-wise monitoring their outgoing bit stream. Those cards, whose outgoing CID bits do not match the corresponding bits on the command line in any one of the bit periods, stop

sending their CID immediately and must wait for the next identification cycle. Since the CID is unique for each card, only one card can be successfully send its full CID to the host. This card then goes into the Identification State. Thereafter, the host issues Set_Relative_Addr (CMD3) to assign to the card a relative card address (RCA). Once the RCA is received the card state changes to the Stand-by State, and the card does not react in further identification cycles, and its output driver switches from open-drain to push-pull. The host repeats the process, mainly CMD2 and CMD3, until the host receives a time-out condition to recognize the completion of the identification process.

```
card_registry()
{
do { // decide RCA for each card until response time-out
          if(card is labelled as SDCombo or SDIO) { // for SDIO card like device
                    send_command(SET_RELATIVE_ADDR, 0x00, <...>); // ask SDIO card to
publish its
RCA
                    retrieve RCA from response;
          } // end if (card is labelled as SDCombo ...
          else if (card is labelled as SD) { // for SD card
                    send_command(ALL_SEND_CID, <...>);
                    if (RESP_TIMEOUT == wait_for_response(ALL_SEND_CID)) break;
                    send_command(SET_RELATIVE_ADDR, <...>);
                    retrieve RCA from response;
          } // else if (card is labelled as SD ...
          else if (card is labelled as MMC) {
                    send_command(ALL_SEND_CID, <...>);
                    rca = 0x1; // arbitrarily set RCA, 1 here for example
                    send_command(SET_RELATIVE_ADDR, 0x1 << 16, <...>); // send RCA at upper
16
bits
          } // end of else if (card is labelled as MMC ...
} while (response is not time-out);
}
```

16.5.3 Card access

This section describes access to the card.

16.5.3.1 Block write

16.5.3.1.1 Normal write

During a block write (CMD24 - 27, CMD60, CMD61), one or more blocks of data are transferred from the host to the card with a CRC appended to the end of each block by the host.

If the CRC fails, the card 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 MMC 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/MMC cards, 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/MMC cards, 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.

16.5.3.1.2 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/MMC, 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.

16.5.3.2 Block read

16.5.3.2.1 Normal read

For block reads, the basic unit of data transfer is a block whose maximum size is stored in areas defined by the corresponding card specification.

A CRC is appended to the end of each block, ensuring data transfer integrity. The CMD17, CMD18, CMD53, CMD60, CMD61, and so on, can initiate a block read. After completing the transfer, the card returns to the transfer state. For multi-blocks read, data blocks will be continuously transferred until a stop command is issued.

The software flow to read from a card 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/MMC, 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/MMC, 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.

16.5.3.2.2 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/MMC, 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 2.
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.

16.5.3.3 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.

16.5.3.3.1 Suspend

After setting the SABGREQ bit, the host driver may send a Suspend command to switch to another function of the SDIO card. The 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, send the Suspend command to suspend the active function. The CMDTYP bit field must be 2'b00.
3. Check the BS bit of the CCCR in the response. If it is 1, repeat this step until the BS bit is cleared or abandon the suspend operation according to the driver strategy.
4. Send another normal I/O command to the suspended function. The CMDTYP of this command must be 2'b01, so the 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.
5. 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).

6. Begin operation for another function on the SDIO card.

16.5.3.3.2 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 #5 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.

16.5.3.4 ADMA usage

To use the ADMA in a data transfer, the host driver must prepare the correct descriptor chain prior to sending the read/write command.

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.

16.5.3.5 Tuning block procedure

Tuning block is used for SD UHS SDR50 mode.

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 SYSCTL2[UHSM]
6. Set TUNECTL[TB_EN] and program appropriate TUNECTL[TB_MODE]
7. Set SYSCTL[SDCLKEN]
8. Wait for PRSSTAT[SDSTB] to be set
9. Execute tuning procedure

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.

The steps for tuning procedure are:

1. Set BLKATTR2[EXTN], execute tuning.
2. Issue SEND_TUNING_BLK Command (CMD19 for SD, CMD21 for MMC).
3. Wait for IRQSTAT[BRR], buffer read ready, to be set.
4. Clear BRR.
5. Check BLKATTR2[EXTN] to be cleared.
6. Repeat steps 2-5, if EXTN is not cleared.
7. Check BLKATTR2[SAMPCLKSEL], sampling clock select. Its 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.

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.

16.5.3.6 DDR

The steps for using DDR mode are:

1. Clear SYSCTL[SDCLKEN]
2. Wait for PRSSTAT[SDSTB] to be set
3. Program SYSCTL2[UHSM] to 4
4. If required, change the clock division ratio in SYSCTL register
5. Set SYSCTL[SDCLKEN]
6. Wait for PRSSTAT[SDSTB] to be set
7. Again clear SYSCTL[SDCLKEN]
8. Wait for PRSSTAT[SDSTB] to be set
9. Set ESDHCCTL[FAF]
10. Wait for ESDHCCTL[FAF] to be cleared
11. Set SYSCTL[SDCLKEN]
12. 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
3. SD clock divisor should be even for DDR mode

16.5.3.7 Transfer error

16.5.3.7.1 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.

16.5.3.7.2 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 (DMAERRADDRL/H). 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.

16.5.3.7.3 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.

16.5.3.7.4 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.

16.5.3.8 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.

16.5.4 Switch function

MMC cards transferring data at bus widths other than 1-bit is a new feature added to the MMC spec.

The high speed timing mode for all card devices, was also recently defined in various card specifications. To enable these new features, a "switch" command should be issued by the host driver.

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 MMC cards, the high speed mode is queried by a CMD8 and enabled by a CMD6 (with the mnemonic symbol as SWITCH).

The 4-bit and 8-bit bus width of the MMC is also enabled by the SWITCH command, but with a different argument.

These new functions can also be disabled by a software reset. For SDIO cards, it can be done by setting the RES bit in the CCCR register. For other cards, it can be accomplished by issuing a CMD0. This method of restoring to the normal mode is not recommended because a complete identification process is needed before the card is ready for data transfer.

For the sake of simplicity, the following flowcharts do not show current capability check, which is recommended in the function switch process.

16.5.4.1 Query, enable and disable SDIO high speed mode

```
enable_sdio_high_speed_mode(void)
{
send CMD52 to query bit SHS at address 0x13;
if (SHS bit is '0') report the SDIO card does not support high speed mode and return;
send CMD52 to set bit EHS at address 0x13 and read after write to confirm EHS bit is set;
change clock divisor value or configure the system clock feeding into 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)
}
```

16.5.4.2 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)
}
```

16.5.4.3 Query, enable and disable MMC high speed mode

```
enable_mmc_high_speed_mode(void)
{
send CMD9 to get CSD value of MMC;
check if the value of SPEC_VER field is 4 or above;
if (SPEC_VER value is less than 4) report the MMC does not support high speed mode and
return;
set BLKCNT field to 1 (block), set BLKSIZE field to 512 (bytes);
send CMD8 to get EXT_CSD value of MMC;
extract the value of CARD_TYPE field to check the 'high speed mode' in this MMC is 26MHz or
52MHz;
send CMD6 with argument 0x1B90100;
send CMD13 to wait card ready (busy line released);
send CMD8 to get EXT_CSD value of MMC;
check if HS_TIMING byte (byte number 185) is 1;
if (HS_TIMING is not 1) report MMC switching to high speed mode failed and return;
change clock divisor value or configure the system clock feeding into 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 MMC;
check if HS_TIMING byte (byte number 185) is 0;
if (HS_TIMING is not 0) report the function switch failed and return;
change clock divisor value or configure the system clock feeding into eSDHC to generate the
card_clk of the desired value below 20MHz;
(data transactions like normal peers)
}
```

16.5.4.4 Set MMC bus width

```
change_mmc_bus_width(void)
{
send CMD9 to get CSD value of MMC;
check if the value of SPEC_VER field is 4 or above;
if (SPEC_VER value is less than 4) report the MMC does not support multiple bit width and
return;
send CMD6 with argument 0x3B70x00; (8-bit, x=2; 4-bit, x=1; 1-bit, x=0)
send CMD13 to wait card ready (busy line released);
(data transactions like normal peers)
}
```

16.5.5 ADMA operation

16.5.5.1 ADMA1 operation

```
Set_adma1_descriptor
{
if (to start data transfer) {
// Make sure the address is 4KB align.
Set 'Set' type descriptor;
{
Set Act bits to 01;
Set [31:12] bits data length (byte unit);
}
Set 'Tran' type descriptor;
{
Set Act bits to 10;
Set [31:12] bits address (4KB align);
}
}
else if (to fetch descriptor at non-continuous address) {
Set Act bits to 11;
Set [31:12] bits the next descriptor address (4KB 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;
}
```

16.5.5.2 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';
}
```

16.6 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.

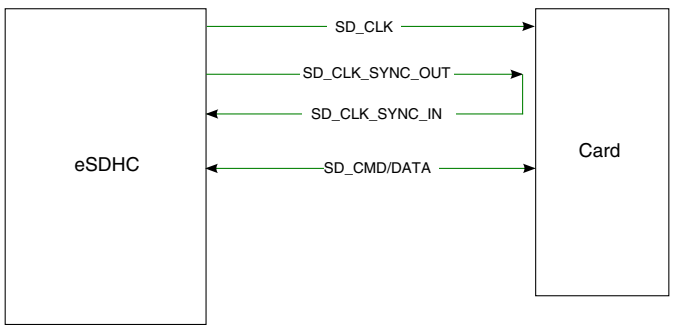


Figure 16-52. Direct Interfacing of eSDHC with Card

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.

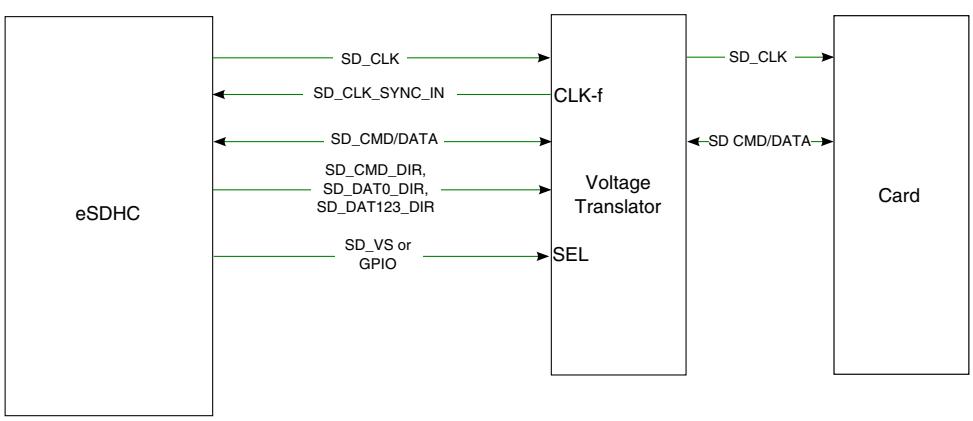


Figure 16-53. Card Interfacing with eSDHC through Voltage Translator

NOTE

CD, WP and DAT4-7 are not shown in above figures for simplicity.

16.7 Commands for MMC/SD/SDIO

See the table below for the list of commands for the MMC/SD/SDIO cards.

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 16-43.

Table 16-42. Commands for MMC/SD/SDIO cards

CMD INDEX	Type	Argument	Resp	Abbreviation	Description
CMD0	bc	[31:0] stuff bits	-	GO_IDLE_STATE	Resets all MMC and SD memory cards to idle state.
CMD1	bcr	[31:0] OCR without busy	R3	SEND_OP_COND	Asks all MMC and SD Memory cards in idle state to send their operation conditions register contents in the response on the CMD line.
CMD2	bcr	[31:0] stuff bits	R2	ALL_SEND_CID	Asks all cards to send their CID numbers on the CMD line.
CMD3[1]	ac	[31:6] RCA [15:0] stuff bits	R1 R6 (SDIO)	SET/ SEND_RELATIVE_ADDR	Assigns relative address to the card.
CMD4	bc	[31:0] DSR [15:0] stuff bits	-	SET_DSR	Programs the DSR of all cards.
CMD5	bc	[31:0] OCR without busy	R4	IO_SEND_OP_COND	Asks all SDIO cards in idle state to send their operation conditions register contents in the response on the CMD line.
CMD6[2]	adtc	[31] Mode 0: Check function 1: Switch function [30:8] Reserved for function groups 6 - 3 (All 0 or 0xFFFF) [7:4] Function group1 for command system [3:0] Function group2 for access mode	R1	SWITCH_FUNC	Checks switch ability (mode 0) and switch card function (mode 1). Refer "SD Physical Specification V1.1" for more details.

Table continues on the next page...

Table 16-42. Commands for MMC/SD/SDIO cards (continued)

CMD INDEX	Type	Argument	Resp	Abbreviation	Description
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	[31:6] RCA [15:0] stuff bits	R1b	SELECT/ DESELECT_CARD	Toggles a card between the stand-by and transfer states or between the programming and disconnect states. In both cases, the card is selected by its own relative address and gets deselected by any other address. Address 0 deselects all.
CMD8	adtc	[31:0] stuff bits	R1	SEND_EXT_CSD	The card sends its EXT_CSD register as a block of data, with a block size of 512 bytes.
CMD9	ac	[31:6] RCA [15:0] stuff bits	R2	SEND_CSD	Addressed card sends its card-specific data (CSD) on the CMD line.
CMD10	ac	[31:6] RCA [15:0] stuff bits	R2	SEND_CID	Addressed card sends its card-identification (CID) on the CMD line.
CMD11	adtc	[31:0] data address	R1	READ_DAT_UNTIL_STOP	Reads data stream from the card, starting at the given address, until a STOP_TRANSMISSION follows.
CMD12	ac	[31:0] stuff bits	R1b	STOP_TRANSMISSION	Forces the card to stop transmission.
CMD13	ac	[31:6] RCA [15:0] stuff bits	R1	SEND_STATUS	Addressed card sends its status register.
CMD14	Reserved				
CMD15	ac	[31:6] RCA [15:0] stuff bits	-	GO_INACTIVE_STATE	Sets the card to inactive state in order to protect the card stack against communication breakdowns.
CMD16	ac	[31:0] block length	R1	SET_BLOCKLEN	Sets the block length (in bytes) for all following block commands (read and write). Default block length is specified in the CSD.
CMD17	adtc	[31:0] data address	R1	READ_SINGLE_BLOCK	Reads a block of the size selected by the SET_BLOCKLEN command.
CMD18	adtc	[31:0] data address	R1	READ_MULTIPLE_BLOCK	Continuously transfers data blocks from card to host until interrupted by a stop command.
CMD19	Reserved				
CMD20	adtc	[31:0] data address	R1	WRITE_DAT_UNTIL_STOP	Writes data stream from the host, starting at the given address, until a STOP_TRANSMISION follows.
CMD21-22	Reserved				

Table continues on the next page...

Table 16-42. Commands for MMC/SD/SDIO cards (continued)

CMD INDEX	Type	Argument	Resp	Abbreviation	Description
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.
CMD28	ac	[31:0] data address	R1b	SET_WRITE_PROT	If the card has write protection features, this command sets the write protection bit of the addressed group. The properties of write protection are coded in the card specific data (WP_GRP_SIZE).
CMD29	ac	[31:0] data address	R1b	CLR_WRITE_PROT	If the card provides write protection features, this command clears the write protection bit of the addressed group.
CMD30	adtc	[31:0] write protect data address	R1	SEND_WRITE_PROT	If the card provides write protection features, this command asks the card to send the status of the write protection bits.
CMD31	Reserved				
CMD32	ac	[31:0] data address	R1	TAG_SECTOR_START	Sets the address of the first sector of the erase group.
CMD33	ac	[31:0] data address	R1	TAG_SECTOR_END	Sets the address of the last sector in a continuous range within the selection of a single sector to be selected for erase.
CMD34	ac	[31:0] data address	R1	UNTAG_SECTOR	Removes one previously selected sector from the erase selection.
CMD35	ac	[31:0] data address	R1	TAG_ERASE_GROUP_START	Sets the address of the first erase group within a range to be selected for erase.
CMD36	ac	[31:0] data address	R1	TAG_ERASE_GROUP_END	Sets the address of the last erase group within a continuous range to be selected for erase.
CMD37	ac	[31:0] data address	R1	UNTAG_ERASE_GROUP	Removes one previously selected erase group from the erase selection.
CMD38	ac	[31:0] stuff bits	R1b	ERASE	Erase all previously selected sectors.

Table continues on the next page...

Table 16-42. Commands for MMC/SD/SDIO cards (continued)

CMD INDEX	Type	Argument	Resp	Abbreviation	Description
CMD39	ac	[31:0] RCA [15] register write flag [14:8] register address [7:0] register data	R4	FAST_IO	Used to write and read 8-bit (register) data fields. The command addresses a card, and a register, and provides the data for writing if the write flag is set. The R4 response contains data read from the address register. This command accesses application dependent registers which are not defined in the MMC standard.
CMD40	bcr	[31:0] stuff bits	R5	GO_IRQ_STATE	Sets the system into interrupt mode.
CMD41	Reserved				
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.
CMD57-59	Reserved				
CMD60	Reserved				
CMD61	adtc	[31] WR [30:16] stuff bits [15:0] data unit count	R1b	RW_MULTIPLE_BLOCK	The host issues a RW_MULTIPLE_BLOCK (CMD61) to begin the data transfer for the ATA command.
CMD62-63	Reserved				
ACMD6[4]	ac	[31:2] stuff bits [1:0] bus width	R1	SET_BUS_WIDTH	Defines the data bus width ('00'=1bit or '10'=4bit bus) to be used for data transfer. The allowed data bus widths are given in SCR register.
ACMD13[4]	adtc	[31:0] stuff bits	R1	SD_STATUS	Send the SD Memory Card status.
ACMD18[4]	adtc	[31:0] stuff bits	R1	SECURE_READ_MULTI_BLOCK	Protected Area Access Command:

Table continues on the next page...

Table 16-42. Commands for MMC/SD/SDIO cards (continued)

CMD INDEX	Type	Argument	Resp	Abbreviation	Description
					Reads continuously transfer data blocks from protected area of SD memory card. Refer Security Specification Version 2.00 for more details.
ACMD22[4]	adtc	[31:0] stuff bits	R1	SEND_NUM_WR_SECTORS	Send the number of the written sectors (without errors). Responds with 32-bit plus the CRC data block.
ACMD23[4]	ac	[31:0] stuff bits	R1	SET_WR_BLK_ERASE_COUNT	-
ACMD25[4]	adtc	[31:0] stuff bits	R1	SECURE_WRITE_MULTI_BLOCK	Protected Area Access Command: Writes continuously transfer data blocks to protected area of SD memory card. Refer Security Specification Version 2.00 for more details.
ACMD26[4]	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[4]	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[4]	bcr	[31:0] OCR	R3	SD_APP_OP_COND	Asks the accessed card to send its operating condition register (OCR) contents in the response on the CMD line.
ACMD42[4]	ac		R1	SET_CLR_CARD_DETECT	-
ACMD43[4]	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.

Table continues on the next page...

Table 16-42. Commands for MMC/SD/SDIO cards (continued)

CMD INDEX	Type	Argument	Resp	Abbreviation	Description
ACMD44[4]	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[4]	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[4]	adtc	[31:0] stuff bits	R1	GET_CER_RN2	AKE Command: Reads random number RN2 as challenge2 in AKE process. Refer security specification version 2.00 for more details.
ACMD47[4]	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[4]	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[4]	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[4]	adtc	[31:0] stuff bits	R1	SEND_SCR	Reads the SD configuration register (SCR).

1. CMD3 differs for MMC and SD cards. For MMC cards, it is referred to as SET_RELATIVE_ADDR, with a response type of R1. For SD cards, it is referred to as SEND_RELATIVE_ADDR, with a response type of R6 (with RCA inside).
2. CMD6 differs completely between high speed MMC cards and high speed SD cards. Command SWITCH_FUNC is for high speed SD cards.
3. Command SWITCH is for high speed MMC cards. The Index field can contain any value from 0-255, but only values 0-191 are valid. If the Index value is in the 192-255 range the card does not perform any modification and the SWITCH_ERROR status bit in the EXT_CSD register is set. The Access Bits are shown in Table 2.
4. ACMDs 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 16-43. 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.

Table continues on the next page...

Table 16-43. EXT_CSD access modes (continued)

Bits	Access name	Operation
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.

16.8 Software restrictions

This section covers software restrictions.

16.8.1 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).

16.8.2 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.

16.8.3 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.

16.8.4 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 MMC, 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.

16.8.5 ADMA address

For ADMA1/ADMA2 operation, the ADMA system address register (ADSADDRL/H) and address defined in the address field of the descriptor table should be 4-byte aligned.

16.8.6 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.

16.8.7 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.

16.8.8 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]).

16.8.9 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 17
Universal Serial Bus Interfaces

17.1 USB 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*
- *USB 2.0 Transceiver Macrocell Interface (UTMI) Specification, Version 1.05*

The USB 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 module is either a device or host controller.

The module contains a chaining DMA (direct memory access) engine that reduces the interrupt load on the application processor and reduces the total system bus bandwidth that must be dedicated to servicing the USB interface requirements.

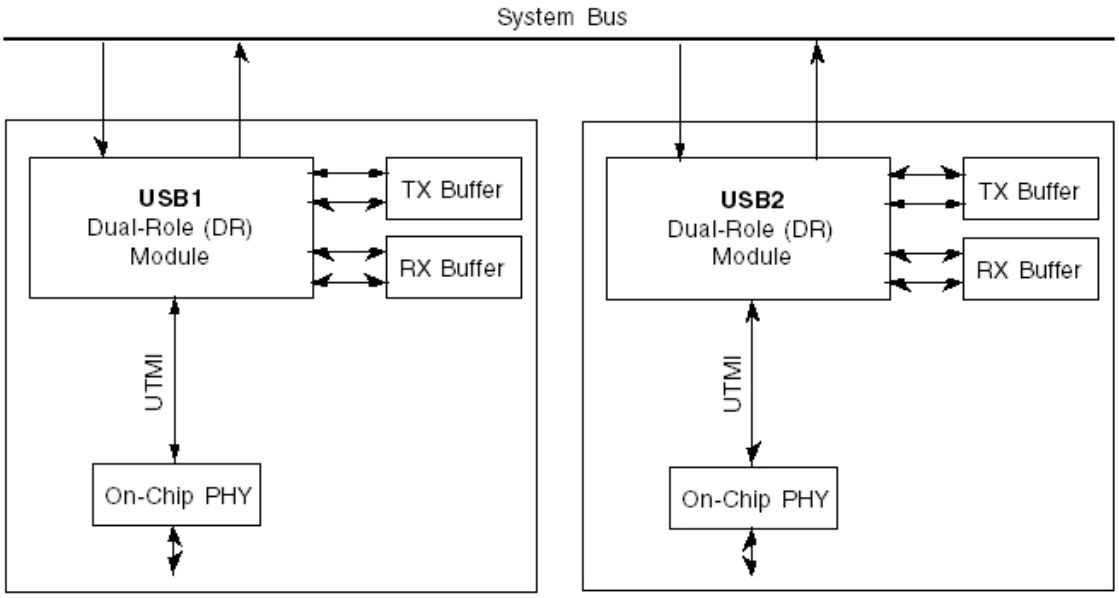


Figure 17-1. USB Interface Block Diagram

17.2 USB Features Summary

The USB 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 on-chip, USB-2.0, full-speed/high-speed PHY with UTMI
- Supports operation as a standalone USB device
  - Supports one upstream facing port
  - Supports six bidirectional USB endpoints
- Host or device support

17.3 USB Modes of Operation

The USB module operates in these modes:

- Host
- Device

NOTE

Only high-speed and full-speed operations are supported in device mode.

17.4 USB External Signals

This section contains detailed descriptions of all the USB controller signals.The table below describes the signals, indicating which interface supports each signal.

Table 17-1. USB External Signals

Signal	I/O	Description
USB*n*_UDP	I/O	On-chip PHY-USB Data Plus
USB*n*_UDM	I/O	On-chip PHY-USB Data Minus
USB*n*_VBUS_CLMP	I	On-chip PHY-VBUS_CLMP
USB*n*_UID	I	On-chip PHY-USB ID Detect
USB*n*_DRVVBUS	O	On-chip PHY-DRVVBUS is used to enable (asserted) or disable (negated) power on devices that support port power switching.
USB*n*_PWRFAULT	I	On-chip PHY-Power fault. PWRFAULT is used to indicate when a Vbus fault has occurred.
USB_CLKIN	I	USB_CLKIN is the clock input for the USB PHYs

17.4.1 UTMI Interface

UTMI was developed to specify a standard interface between USB 2.0 controllers and USB 2.0 PHY's. This interface is made available to support applications that may use a UTMI-compliant PHY. UTMI+ extensions are not supported by the USB module. Only the integrated PHY uses the UTMI interface; therefore, there are no external UTMI signals to be documented in this manual.

17.4.2 PHY Clocks

The USBCLKIN input provides the clocking signal for the UTMI interface. Detailed clock specifications are given in the chip hardware specifications.

17.5 USB Memory Map

This section provides the memory map and detailed descriptions of all USB interface registers.

The table below shows the memory mapped registers of the USB controllers and their offsets. It lists the offset, name, and a cross-reference to the complete description of each register. Note that the full register address is comprised of CCSRBAR together with the USB controller block base address and offset listed below. Undefined 4-byte address spaces within offset 0x000-0xFFF are reserved.

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 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.

USB memory map

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
21_0100	Capability register length (USB1_CAPLENGTH)	8	R	40h	17.5.1/945
21_0102	Host interface version number (USB1_HCIVERSION)	16	R	0100h	17.5.2/946
21_0104	Host controller structural parameters (USB1_HCSPARAMS)	32	R	0001_0011h	17.5.3/946
21_0108	Host controller capability parameters (USB1_HCCPARAMS)	32	R	0000_0006h	17.5.4/948
21_0120	Device interface version number (USB1_DCIVERSION)	16	R	0001h	17.5.5/950
21_0124	Device controller parameters (USB1_DCCPARAMS)	32	R	0000_0186h	17.5.6/951
21_0140	USB command (USB1_USBCMD)	32	R/W	0008_0000h	17.5.7/952
21_0144	USB status (USB1_USBSTS)	32	R/W	0000_0000h	17.5.8/956

Table continues on the next page...

USB memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
21_0148	USB interrupt enable (USB1_USBINTR)	32	R/W	0000_0000h	17.5.9/960
21_014C	USB frame index (USB1_FRINDEX)	32	R/W	0000_0000h	17.5.10/ 962
21_0154	Frame list base address (USB1_PERIODICLISTBASE)	32	R/W	0000_0000h	17.5.11/ 963
21_0154	USB device address (USB1_DEVICEADDR)	32	R/W	0000_0000h	17.5.12/ 964
21_0158	Next asynchronous list addr [host mode] (USB1_ASYNCLISTADDR)	32	R/W	0000_0000h	17.5.13/ 964
21_0158	Address at endpoint list [device mode] (USB1_ENDPOINT_ADDR)	32	R/W	0000_0000h	17.5.14/ 965
21_0164	Host TT transmit pre-buffer packet tuning (USB1_TXFILLTUNING)	32	R/W	See section	17.5.15/ 965
21_0180	Configured flag register (USB1_CONFIGFLAG)	32	R	0000_0001h	17.5.16/ 967
21_0184	Port status/control (USB1_PORTSC)	32	R/W	See section	17.5.17/ 968
21_01A8	USB device mode (USB1_USBMODE)	32	R/W	0000_5000h	17.5.18/ 975
21_01AC	Endpoint setup status (USB1_ENDPTSETUPSTAT)	32	R/W	0000_0000h	17.5.19/ 976
21_01B0	Endpoint initialization (USB1_ENDPOINTPRIME)	32	R/W	0000_0000h	17.5.20/ 977
21_01B4	Endpoint de-initialize (USB1_ENDPTFLUSH)	32	R/W	0000_0000h	17.5.21/ 978
21_01B8	Endpoint status (USB1_ENDPTSTATUS)	32	R	0000_0000h	17.5.22/ 978
21_01BC	Endpoint complete (USB1_ENDPTCOMPLETE)	32	w1c	0000_0000h	17.5.23/ 979
21_01C0	Endpoint control 0 (USB1_ENDPTCTRL0)	32	R/W	0080_0080h	17.5.24/ 981
21_01C4	Endpoint control n (USB1_ENDPTCTRL1)	32	R/W	0000_0000h	17.5.25/ 983
21_01C8	Endpoint control n (USB1_ENDPTCTRL2)	32	R/W	0000_0000h	17.5.25/ 983
21_01CC	Endpoint control n (USB1_ENDPTCTRL3)	32	R/W	0000_0000h	17.5.25/ 983
21_01D0	Endpoint control n (USB1_ENDPTCTRL4)	32	R/W	0000_0000h	17.5.25/ 983
21_01D4	Endpoint control n (USB1_ENDPTCTRL5)	32	R/W	0000_0000h	17.5.25/ 983
21_0400	Snoop n (USB1_SNOOP1)	32	R/W	0000_0000h	17.5.26/ 985

Table continues on the next page...

USB memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
21_0404	Snoop n (USB1_SNOOP2)	32	R/W	0000_0000h	17.5.26/ 985
21_0408	Age count threshold (USB1_AGE_CNT_THRESH)	32	R/W	0000_0000h	17.5.27/ 986
21_040C	Priority control (USB1_PRI_CTRL)	32	R/W	0000_0000h	17.5.28/ 987
21_0410	System interface control (USB1_SI_CTRL)	32	R/W	0000_0000h	17.5.29/ 988
21_0500	Control (USB1_CONTROL)	32	R/W	0000_0000h	17.5.30/ 990
21_1100	Capability register length (USB2_CAPLENGTH)	8	R	40h	17.5.1/945
21_1102	Host interface version number (USB2_HCIVERSION)	16	R	0100h	17.5.2/946
21_1104	Host controller structural parameters (USB2_HCSPARAMS)	32	R	0001_0011h	17.5.3/946
21_1108	Host controller capability parameters (USB2_HCCPARAMS)	32	R	0000_0006h	17.5.4/948
21_1120	Device interface version number (USB2_DCIVERSION)	16	R	0001h	17.5.5/950
21_1124	Device controller parameters (USB2_DCCPARAMS)	32	R	0000_0186h	17.5.6/951
21_1140	USB command (USB2_USBCMD)	32	R/W	0008_0000h	17.5.7/952
21_1144	USB status (USB2_USBSTS)	32	R/W	0000_0000h	17.5.8/956
21_1148	USB interrupt enable (USB2_USBINTR)	32	R/W	0000_0000h	17.5.9/960
21_114C	USB frame index (USB2_FRINDEX)	32	R/W	0000_0000h	17.5.10/ 962
21_1154	Frame list base address (USB2_PERIODICLISTBASE)	32	R/W	0000_0000h	17.5.11/ 963
21_1154	USB device address (USB2_DEVICEADDR)	32	R/W	0000_0000h	17.5.12/ 964
21_1158	Next asynchronous list addr [host mode] (USB2_ASYNCLISTADDR)	32	R/W	0000_0000h	17.5.13/ 964
21_1158	Address at endpoint list [device mode] (USB2_ENDPOINT_ADDR)	32	R/W	0000_0000h	17.5.14/ 965
21_1164	Host TT transmit pre-buffer packet tuning (USB2_TXFILLTUNING)	32	R/W	See section	17.5.15/ 965
21_1180	Configured flag register (USB2_CONFIGFLAG)	32	R	0000_0001h	17.5.16/ 967
21_1184	Port status/control (USB2_PORTSC)	32	R/W	See section	17.5.17/ 968
21_11A8	USB device mode (USB2_USBMODE)	32	R/W	0000_5000h	17.5.18/ 975
21_11AC	Endpoint setup status (USB2_ENDPTSETUPSTAT)	32	R/W	0000_0000h	17.5.19/ 976
21_11B0	Endpoint initialization (USB2_ENDPOINTPRIME)	32	R/W	0000_0000h	17.5.20/ 977
21_11B4	Endpoint de-initialize (USB2_ENDPTFLUSH)	32	R/W	0000_0000h	17.5.21/ 978

Table continues on the next page...

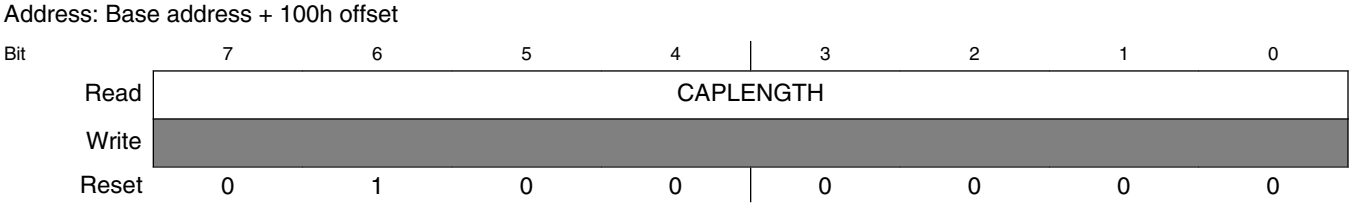
USB memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
21_11B8	Endpoint status (USB2_ENDPTSTATUS)	32	R	0000_0000h	17.5.22/ 978
21_11BC	Endpoint complete (USB2_ENDPTCOMPLETE)	32	w1c	0000_0000h	17.5.23/ 979
21_11C0	Endpoint control 0 (USB2_ENDPTCTRL0)	32	R/W	0080_0080h	17.5.24/ 981
21_11C4	Endpoint control n (USB2_ENDPTCTRL1)	32	R/W	0000_0000h	17.5.25/ 983
21_11C8	Endpoint control n (USB2_ENDPTCTRL2)	32	R/W	0000_0000h	17.5.25/ 983
21_11CC	Endpoint control n (USB2_ENDPTCTRL3)	32	R/W	0000_0000h	17.5.25/ 983
21_11D0	Endpoint control n (USB2_ENDPTCTRL4)	32	R/W	0000_0000h	17.5.25/ 983
21_11D4	Endpoint control n (USB2_ENDPTCTRL5)	32	R/W	0000_0000h	17.5.25/ 983
21_1400	Snoop n (USB2_SNOOP1)	32	R/W	0000_0000h	17.5.26/ 985
21_1404	Snoop n (USB2_SNOOP2)	32	R/W	0000_0000h	17.5.26/ 985
21_1408	Age count threshold (USB2_AGE_CNT_THRESH)	32	R/W	0000_0000h	17.5.27/ 986
21_140C	Priority control (USB2_PRI_CTRL)	32	R/W	0000_0000h	17.5.28/ 987
21_1410	System interface control (USB2_SI_CTRL)	32	R/W	0000_0000h	17.5.29/ 988
21_1500	Control (USB2_CONTROL)	32	R/W	0000_0000h	17.5.30/ 990

17.5.1 Capability register length (USBx_CAPLENGTH)

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.

Address: Base address + 100h offset



USBx_CAPLENGTH field descriptions

Field	Description
CAPLENGTH	Capability registers length. Value is 0x40.

17.5.2 Host interface version number (USBx_HCIVERSION)

HCIVERSION contains a BCD encoding of the EHCI revision number supported by this host controller. The most-significant byte of the register represents a major revision and the least-significant byte is the minor revision.

Address: Base address + 102h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	HCIVERSION															
Write																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

USBx_HCIVERSION field descriptions

Field	Description
HCIVERSION	EHCI revision number. Value is 0x0100 indicating version 1.0.

17.5.3 Host controller structural parameters (USBx_HCSPARAMS)

HCSPARAMS contains structural parameters such as the number of downstream ports.

Address: Base address + 104h offset

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

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	N_CC				N_PCC				Reserved			PPC	N_PORTS			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1

USBx_HCSPARAMS field descriptions

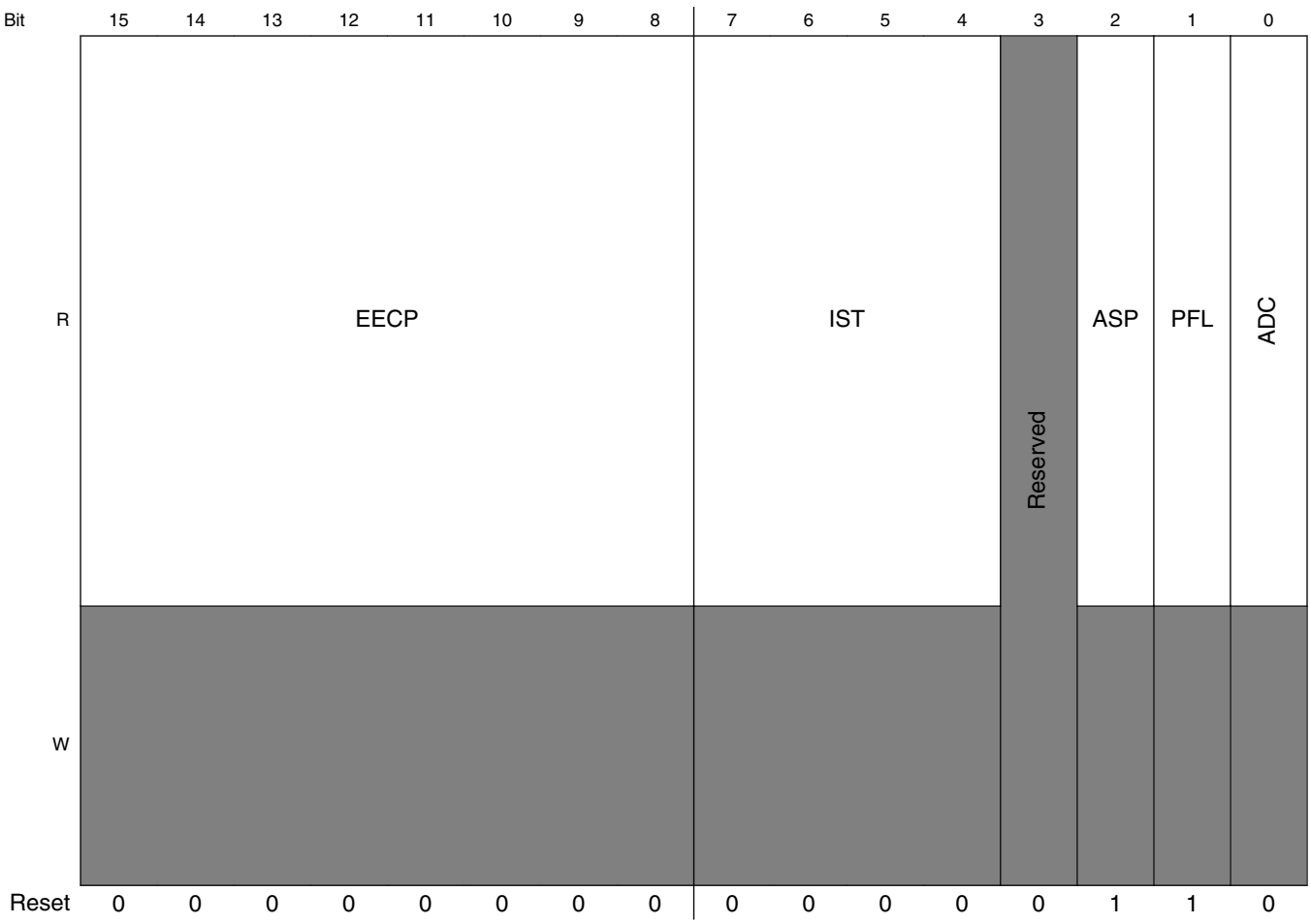
Field	Description
31–28 -	This field is reserved. Reserved
27–24 N_TT	Number of transaction translators. This is a non-EHCI field. This field indicates the number of embedded transaction translators associated the module. This field is always 1. See Embedded Transaction Translator Function.
23–20 N_PTT	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.
19–17 -	This field is reserved. Reserved
16 PI	Port indicators. Indicates whether the ports support port indicator control. Always 1. 1 The port status and control registers include a R/W field for controlling the state of the port indicator.
15–12 N_CC	Number of companion controllers associated with the USB controller. Always 0.
11–8 N_PCC	Number ports per CC. This field indicates the number of ports supported per internal companion controller. This field is always 0.
7–5 -	This field is reserved. Reserved
4 PPC	Power port control. Indicates whether the host controller supports port power control. It is always 1. 1 Ports have power port switches.
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 0x1.

17.5.4 Host controller capability parameters (USBx_HCCPARAMS)

HCCPARAMS identifies multiple mode control (time-base bit functionality) addressing capability.

Address: Base address + 108h offset

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



USBx_HCCPARAMS field descriptions

Field	Description
31–16 -	This field is reserved. Reserved
15–8 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.
7–4 IST	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.
3 -	This field is reserved. Reserved
2 ASP	Asynchronous schedule park capability. Indicates whether the USB 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.

Table continues on the next page...

USBx_HCCPARAMS field descriptions (continued)

Field	Description
	This field is always 1 (park feature supported).
1 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 4K page boundary. This requirement ensures that the frame list is always physically contiguous. This field is always 1.
0 ADC	64-bit addressing capability. Always 0; 64-bit addressing is not supported. 0 Data structures use 32-bit address memory pointers

17.5.5 Device interface version number (USBx_DCIVERSION)

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 a major revision and the least-significant byte is the minor revision.

Address: Base address + 120h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	DCIVERSION															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

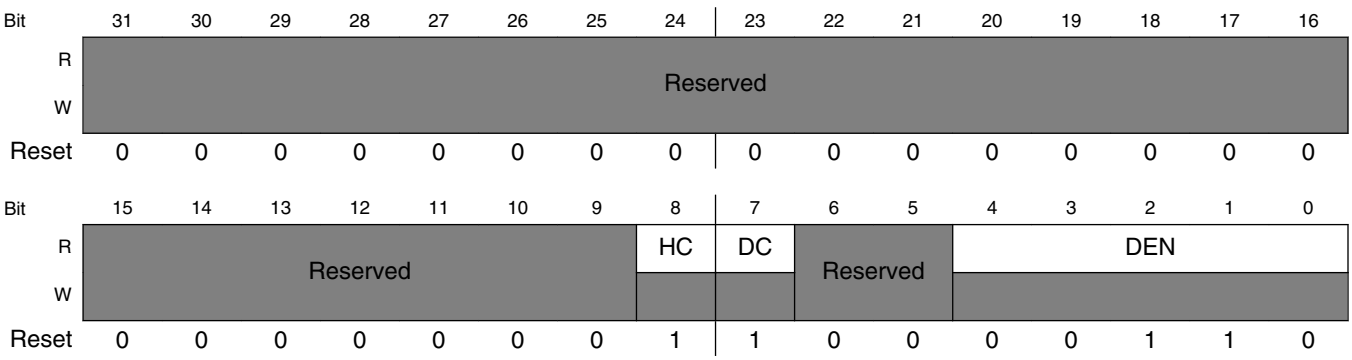
USBx_DCIVERSION field descriptions

Field	Description
DCIVERSION	Device interface revision number.

17.5.6 Device controller parameters (USBx_DCCPARAMS)

This register is not defined in the EHCI specification. This register describes the overall host/device capability of the USB module.

Address: Base address + 124h offset



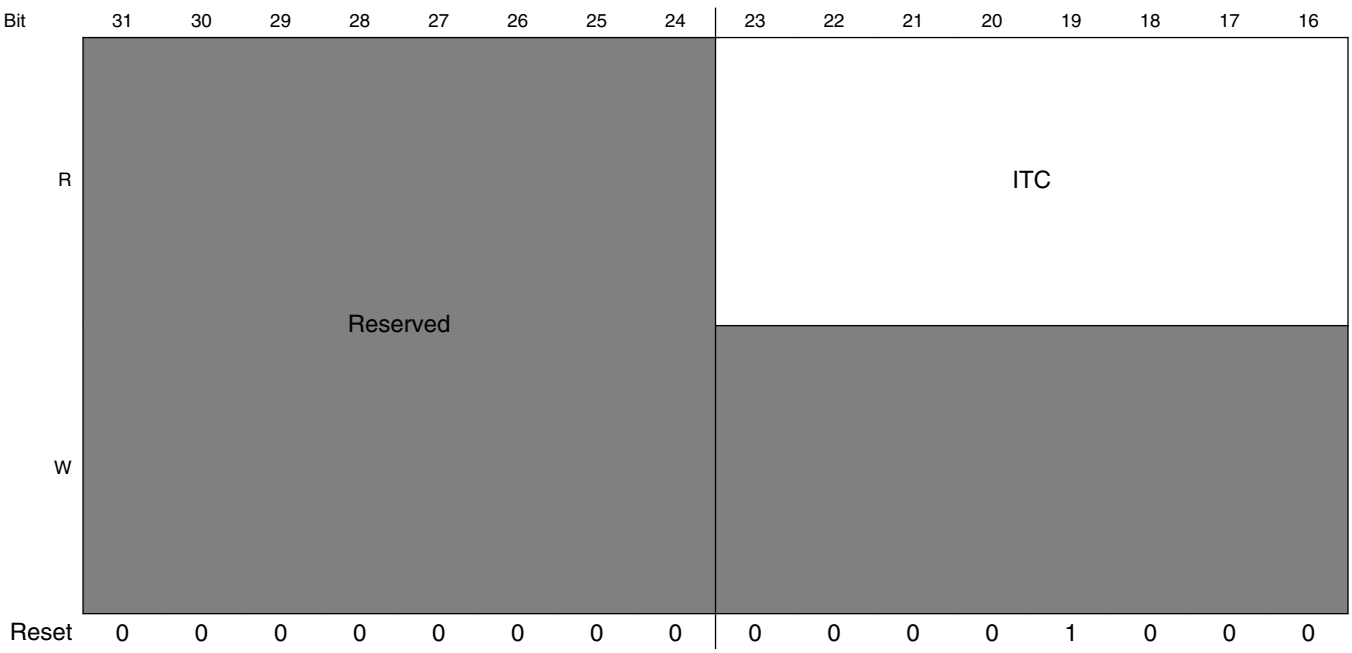
USBx_DCCPARAMS field descriptions

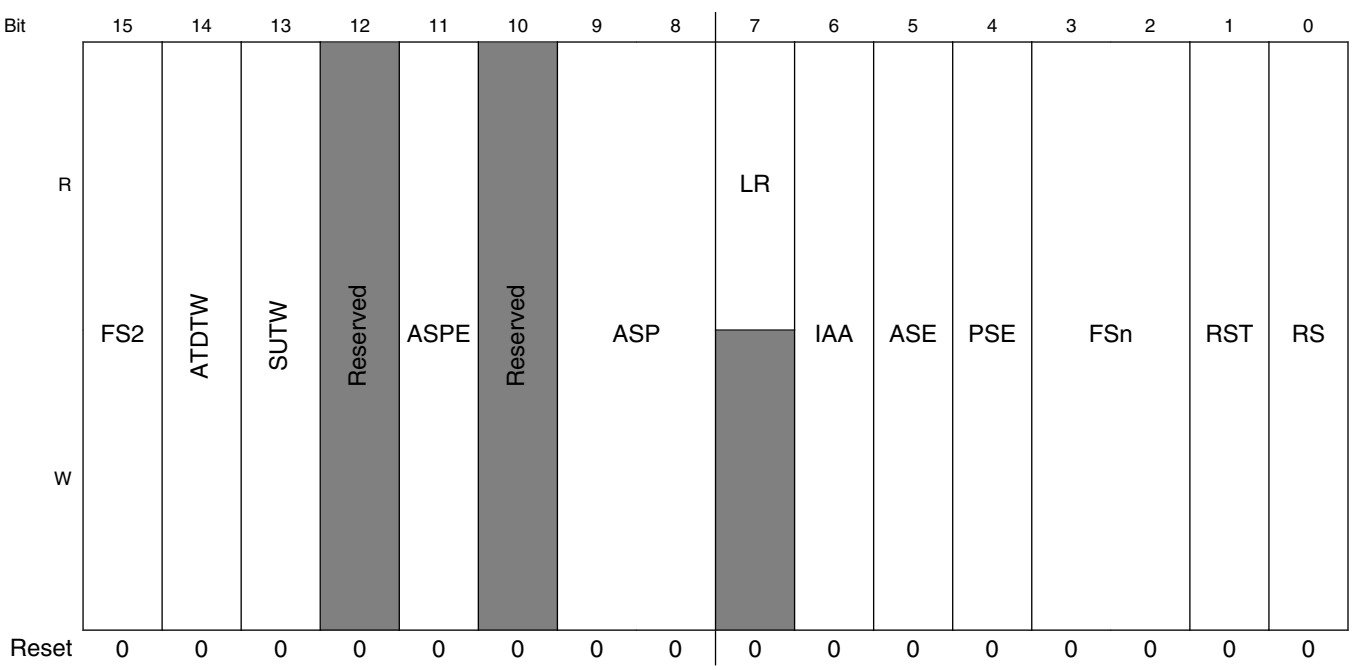
Field	Description
31–9 -	This field is reserved. Reserved
8 HC	Host capable. Always 1, indicating the USB controller can operate as an EHCI compatible USB 2.0 host.
7 DC	Device capable. Always 1, indicating the USB controller can operate as an USB 2.0 device. 1 Device capability 0 No device capability (host only)
6–5 -	This field is reserved. Reserved
DEN	Device endpoint number. Indicates the number of endpoints built into the device controller. Always 0x 6 .

17.5.7 USB command (USBx_USBCMD)

The module executes the command indicated in this register, shown below.

Address: Base address + 140h offset





USBx_USBCMD field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23–16 ITC	Interrupt threshold control. The system software uses this field to set the maximum rate at which the USB module will issue interrupts. ITC contains the maximum interrupt interval measured in microframes. Valid values are shown below. 0x00 Immediate (no threshold) 0x01 1 microframe 0x02 2 microframes 0x04 4 microframes 0x08 8 microframes 0x10 16 microframes 0x20 32 microframes 0x40 40 microframes
15 FS2	See bits 3-2 below. This is a non-EHCI bit.
14 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 state machine is hazard region where adding a dTD to a primed endpoint may go unrecognized. More information on the use of this bit is described in Device Operation.
13 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 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. More information on the use of this bit is described in Device Operation.
12 -	This field is reserved. Reserved
11 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. 0 Disabled 1 Enabled
10 -	This field is reserved. Reserved
9–8 ASP	Asynchronous schedule park mode count. This field defaults to 0x3H and is R/W. 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 will result in undefined behavior.
7 LR	Light host/device controller reset (OPTIONAL). Not implemented. Always 0.
6 IAA	Interrupt on async advance doorbell. Used as a doorbell by software to tell the USB 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 will assert 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 will yield undefined results.

Table continues on the next page...

USBx_USBCMD field descriptions (continued)

Field	Description
	This bit is only used in host mode. Setting this bit when the USB module is in device mode is selected will result in undefined results.
5 ASE	Asynchronous schedule enable. Controls whether the controller skips processing the asynchronous schedule. Only used in host mode. 0 Do not process the asynchronous schedule 1 Use the ASYNCLISTADDR register to access the asynchronous schedule.
4 PSE	Periodic schedule enable. Controls whether the controller skips processing the periodic schedule. Only used in host mode. 0 Do not process the periodic schedule. 1 Use the PERIODICLISTBASE register to access the periodic schedule.
3–2 FSn	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. 000 1024 elements (4096 bytes) 001 512 elements (2048 bytes) 010 256 elements (1024 bytes) 011 128 elements (512 bytes) 100 64 elements (256 bytes) 101 32 elements (128 bytes) 110 16 elements (64 bytes) 111 8 elements (32 bytes)
1 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 etc. to their initial value. Any transaction currently in progress on USB is immediately terminated. A USB reset is not driven on downstream ports. Software should not set this bit when USBSTS[HCH] is a zero. Attempting to reset an actively running host controller will result in undefined behavior. Device mode: • When software sets this bit, the USB controller resets its internal pipelines, timers, counters, state machines etc. to their initial value. Any transaction currently in progress on USB is immediately terminated. Writing a one to this bit in device mode is not recommended.
0 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 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:

Table continues on the next page...

USBx_USBCMD field descriptions (continued)

Field	Description
	• Setting this bit will cause the USB controller to enable a pull-up on D+ and initiate an attach event. This control bit is not directly connected to the pull-up enable, as the pull-up will become disabled upon transitioning into high-speed mode. Software should use this bit to prevent an attach event before the controller has been properly initialized. Clearing this bit will cause a detach event. 0 Stop 1 Run

17.5.8 USB status (USBx_USBSTS)

This register indicates various states of the USB 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 in figure below).

Address: Base address + 144h offset

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

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	AS	PS	RCL	HCH	Reserved	—	Reserved	SLI	SRI	URI	AAI	SEI	FRI	PCI	UEI	UI
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

USBx_USBSTS field descriptions

Field	Description
31–16 -	This field is reserved. Reserved
15 AS	Asynchronous schedule status. Reports the current real status of the asynchronous schedule. The USB 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. 0 Disabled 1 Enabled
14 PS	Periodic schedule status. Reports the current real status of the periodic schedule. The USB 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. 0 Disabled 1 Enabled
13 RCL	Reclamation. Used to detect an empty asynchronous schedule. Only used by the host mode. 0 Non-empty asynchronous schedule 1 Empty asynchronous schedule

Table continues on the next page...

USBx_USBSTS field descriptions (continued)

Field	Description
12 HCH	HC halted. This bit is a zero whenever USBCMD[RS] is a one. The USB 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. 0 Running 1 Halted
11 - Reserved	This field is reserved.
10 —	Reserved
9 - Reserved	This field is reserved.
8 SLI	DCSuspend. 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 cleared via software by writing 1 to it. Only used by the device controller. 0 Active 1 Suspended
7 SRI	Host mode: • This is a non-EHCI status bit. In host mode, this bit will be set every 125 us, 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 controller detects a Start Of (micro) Frame, this bit will be set. When a SOF is extremely late, the USB controller will automatically set this bit to indicate that an SOF was expected. Therefore, this bit will be set roughly every 1 msec in device FS mode and every 125 msec in HS mode and will be synchronized to the actual SOF that is received. Because the controller is initialized to FS before connect, this bit will be set at an interval of 1 msec during the prelude to the connect and chirp. Software writes a 1 to this bit to clear it.
6 URI	USB reset received. This is a non-EHCI bit. When the USB controller detects a USB reset and enters the default state, this bit will be set. Software can write a 1 to this bit to clear the USB reset received status bit. Only used by the device mode. 0 No reset received 1 Reset received
5 AAI	Interrupt on async advance. System software can force the controller to issue an interrupt the next time the USB 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. 0 No async advance interrupt 1 Async advance interrupt
4 SEI	System error. This bit is set whenever an error is detected on the system bus. If USBINTR[SEE] is set, an interrupt will be generated. The interrupt and status bits will remain asserted until cleared by writing a 1 to this bit. Additionally, when in host mode, USBCMD[RS] is cleared, effectively disabling the USB controller. For the USB controller in device mode, an interrupt is generated, but no other action is taken. 0 Normal operation 1 Error

Table continues on the next page...

USBx_USBSTS field descriptions (continued)

Field	Description
3 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 controller sets this bit to a one every time FHINDEX [12] toggles. Only used by the host mode.
2 PCI	Host mode: • Port change detect. 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 controller sets this bit when it enters the full or high-speed operational state. When it exits the full or high-speed operation states due to reset or suspend events, the notification mechanisms are USBSTS[URI] and USBSTS[SLI], respectively. This bit is not EHCI compatible.
1 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. Also see Table 1 in this chapter for more information on device error matrix. For the USB controller in device mode, only resume signaling is detected, all others are ignored. 0 No error 1 Error detected
0 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.

17.5.9 USB interrupt enable (USBx_USBINTR)

The interrupts to software are enabled with this register. An interrupt is generated when a bit is set and the corresponding interrupt is active. The USB status register (USBSTS), shown below, still shows interrupt sources even if they are disabled by the USBINTR register, allowing polling of interrupt events by the software.

Address: Base address + 148h offset

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

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved					Reserved	Reserved	SLE	SRE	URE	AAE	SEE	FRE	PCE	UEE	UE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USBx_USBINTR field descriptions

Field	Description
31–11 -	This field is reserved. Reserved
10 -	This field is reserved. Reserved
9 -	This field is reserved. Reserved
8 SLE	Sleep enable. This is a non-EHCI bit. When this bit is a one, and USBSTS[SLI] transitions, the USB controller will issue an interrupt. The interrupt is acknowledged by software writing a one to USBSTS[SLI]. Only used in device mode. 0 Disable 1 Enable

Table continues on the next page...

USBx_USBINTR field descriptions (continued)

Field	Description
7 SRE	SOF received enable. This is a non-EHCI bit. When this bit is a one, and USBSTS[SRI] is a one, the controller will issue an interrupt. The interrupt is acknowledged by software clearing USBSTS[SRI]. 0 Disable 1 Enable
6 URE	USB reset enable. This is a non-EHCI bit. When this bit is a one, USBSTS[URI] is a one, the device controller will issue an interrupt. The interrupt is acknowledged by software clearing USBSTS[URI] bit. Only used in device mode. 0 Disable 1 Enable
5 AAE	Interrupt on async advance enable. When this bit is a one, and USBSTS[AAI] is a one, the controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing USBSTS[AAI]. Only used in host mode. 0 Disable 1 Enable
4 SEE	System error enable. When this bit is a one, and USBSTS[SEI] is a one, the controller will issue an interrupt. The interrupt is acknowledged by software clearing USBSTS[SEI]. 0 Disable 1 Enable
3 FRE	Frame list rollover enable. When this bit is a one, and USBSTS[FRI] is a one, the controller will issue an interrupt. The interrupt is acknowledged by software clearing USBSTS[FRI]. Only used by the host mode. 0 Disable 1 Enable
2 PCE	Port change detect enable. When this bit is a one, and USBSTS[PCI] is a one, the controller will issue an interrupt. The interrupt is acknowledged by software clearing USBSTS[PCI]. 0 Disable 1 Enable
1 UEE	USB error interrupt enable. When this bit is a one, and USBSTS[UEI] is a one, the controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing USBSTS[UEI]. 0 Disable 1 Enable
0 UE	USB interrupt enable. When this bit is a one, and USBSTS[UI] is a one, the controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing USBSTS[UI]. 0 Disable 1 Enable

17.5.10 USB frame index (USBx_FRINDEX)

In host mode, the FRINDEX 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 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.

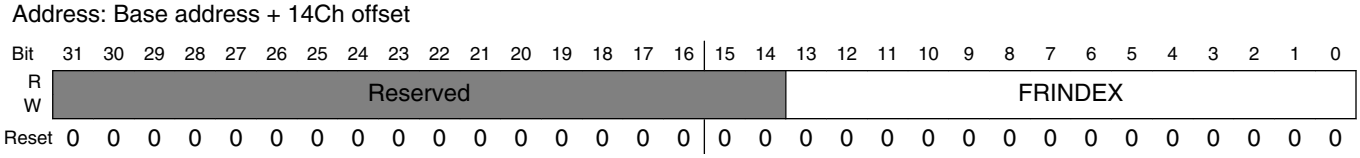
In device mode, this register is read-only and, the USB controller updates the FRINDEX[13-3] register from the frame number indicated by the SOF marker. Whenever a SOF is received by the USB bus, FRINDEX[13-3] is checked against the SOF marker. If FRINDEX[13-3] is different from the SOF marker, FRINDEX[13-3] is set to the SOF value and FRINDEX[2-0] is cleared (that is, SOF for 1 msec frame). If FRINDEX[13-3] is equal to the SOF value, FRINDEX[2-0] is incremented (that is, SOF for 125-µsec microframe.)

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 17-50. 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

Address: Base address + 14Ch offset



USBx_FRINDEX field descriptions

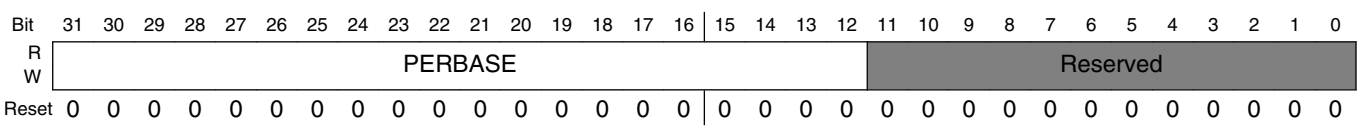
Field	Description
31–14 -	This field is reserved. Reserved
FRINDEX	Frame index. The value in this register increments at the end of each time frame (for example, microframe). Bits N-3 are used for the Frame List current index. This means that each location of the frame list is accessed 8 times (frames 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.

17.5.11 Frame list base address (USBx_PERIODICLISTBASE)

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-Kbyte 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.

Note that 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 (USB_DEVICEADDR), for more information.

Address: Base address + 154h offset



USBx_PERIODICLISTBASE field descriptions

Field	Description
31–12 PERBASE	Base address. Correspond to memory address signal [31:12]. Only used in the host mode.
-	This field is reserved. Reserved

17.5.12 USB device address (USBx_DEVICEADDR)

DEVICEADDR 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 shall reprogram the address after receiving a SET_ADDRESS descriptor.

Note that 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 Frame list base address (USB_PERIODICLISTBASE), for more information.

Address: Base address + 154h 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	USBADR							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

USBx_DEVICEADDR field descriptions

Field	Description
31–25 USBADR	Device address. This field corresponds to the USB device address.
- This field is reserved. Reserved	

17.5.13 Next asynchronous list addr [host mode] (USBx_ASYNCLISTADDR)

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] (USB_ENDPOINT_ADDR), for more information.

Address: Base address + 158h 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	ASYBASE																											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

USBx_ASYNCLISTADDR field descriptions

Field	Description
31–5 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.
-	This field is reserved. Reserved

17.5.14 Address at endpoint list [device mode] (USBx_ENDPOINT_ADDR)

The ENDPOINTLISTADDR 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 Kbytes, so in practice the queue head should be 2-Kbyte aligned.

Note that 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 Current Asynchronous List Address Register , for more information.

Address: Base address + 158h 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	EPBASE																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

USBx_ENDPOINT_ADDR field descriptions

Field	Description
31–11 EPBASE	Endpoint list address. Address of the top of the endpoint list.
-	This field is reserved. Reserved

17.5.15 Host TT transmit pre-buffer packet tuning (USBx_TXFILLTUNING)

This 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 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 ($T_s = T_0 + T_1$)

T_{ff} = Time to fetch packet into TX FIFO up to specified level.

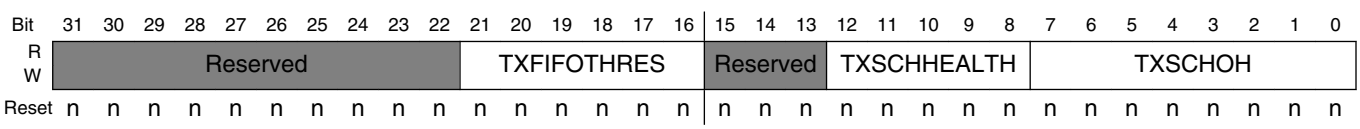
T_p = Total Packet Time (fetch and send) packet ($T_p = T_{ff} + T_s$)

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

The register is read only in host mode and the default reset value is 0x0002_0000.

Address: Base address + 164h offset



USBx_TXFILLTUNING field descriptions

Field	Description
31–22 -	This field is reserved. Reserved
21–16 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

Table continues on the next page...

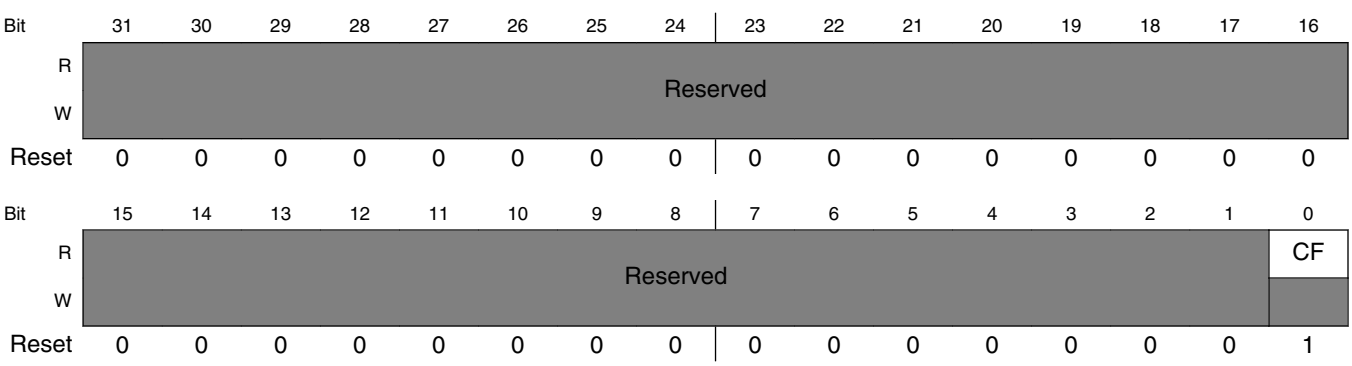
USBx_TXFILLTUNING field descriptions (continued)

Field	Description
	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.
15–13 -	This field is reserved. Reserved
12–8 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.
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. For most applications, TXSCHOH can be set to 4 or less. A good value to begin with is: TXFIFOTHRES x (BURSTSIZE 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. For example, if TXFIFOTHRES is 5 and BURSTSIZE is 8, then set TXSCHOH to 5x(8x4)÷(40x1.267)=4 for a high-speed link. 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.

17.5.16 Configured flag register (USBx_CONFIGFLAG)

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.

Address: Base address + 180h offset



USBx_CONFIGFLAG field descriptions

Field	Description
31–1 -	This field is reserved. Reserved.
0 CF	Configure flag. Always 1 indicating all port routings default to this host.

17.5.17 Port status/control (USBx_PORTSC)

The port status and control (PORTSC) register, shown in the figure below, is only reset when power is initially applied or in response to a controller reset. The initial conditions of a port are as follows:

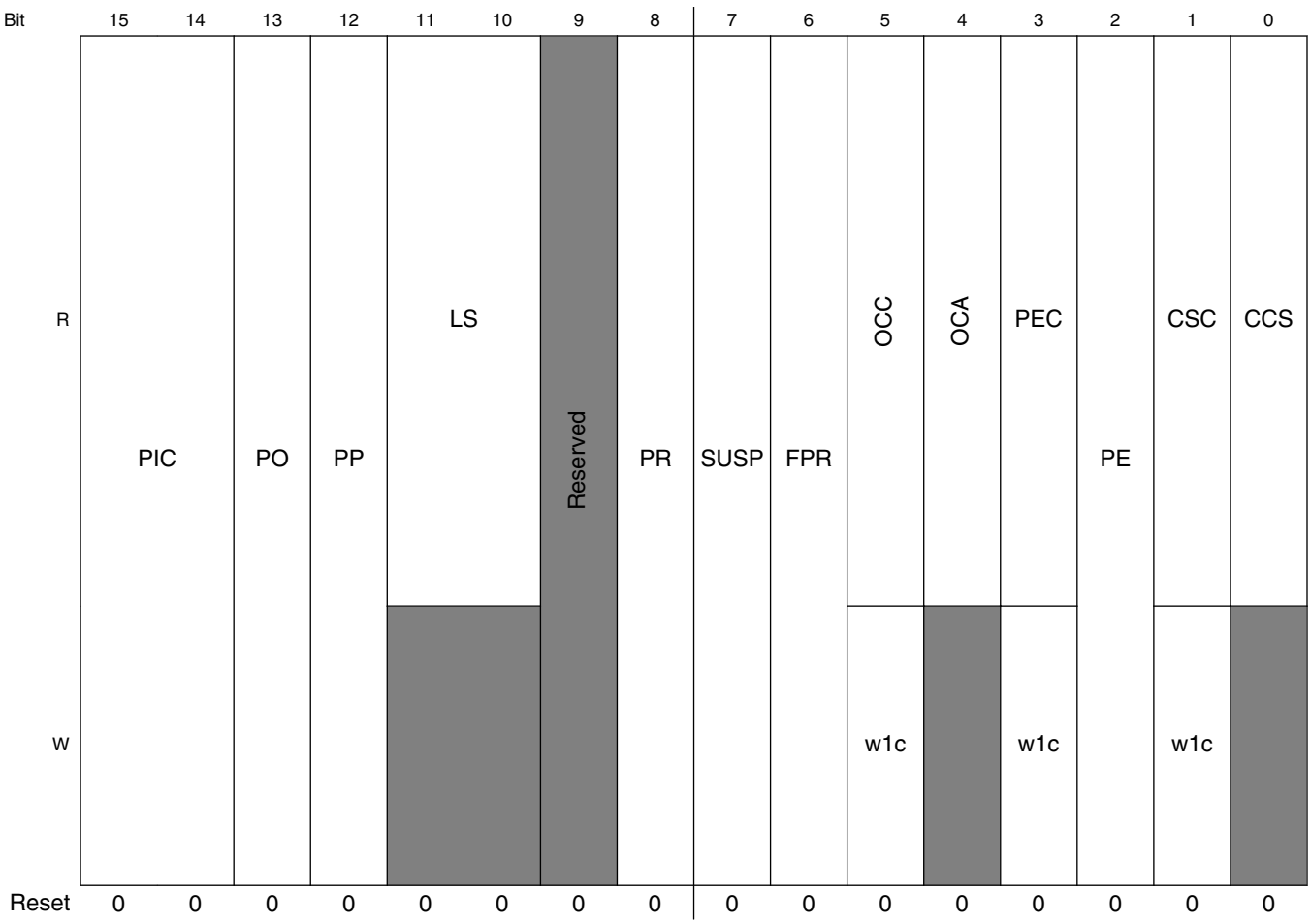
- No device connected
- Port disabled

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 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.

Address: Base address + 184h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PTS		Reserved	Reserved	PSPD		Reserved	PFSC	PHCD	WKOC	WKDS	WKCN	PTC			
W																
Reset	1	0	0	0	n	n	0	0	0	0	0	0	0	0	0	0



USBx_PORTSC field descriptions

Field	Description
31–30 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. 00 UTMI parallel interface 01 Reserved, should be cleared 10 Reserved 11 Reserved
29 -	This field is reserved. Reserved, should be cleared
28 -	This field is reserved. Reserved
27–26 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. 00 Full-speed 01 Low-speed 10 High-speed 11 Undefined

Table continues on the next page...

USBx_PORTSC field descriptions (continued)

Field	Description
25 -	This field is reserved. Reserved, should be cleared
24 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. This bit is not defined in the EHCI specification. This bit is for debugging purposes. 0 Allow the port to identify itself as high speed. 1 Force the port to only connect at full speed.
23 PHCD	PHY low power suspend. This bit is not defined in the EHCI specification. Reading this bit indicates the status of the PHY. 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 will be cleared automatically when the resume signaling has been detected or when forcing port resume. **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 . 0 Normal PHY operation. 1 Signal the PHY to enter low power suspend mode
22 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 (host mode only) for use by an external power control circuit.
21 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 (host mode only) for use by an external power control circuit.
20 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 (host mode only) for use by an external power control circuit.
19–16 PTC	Port test control. Any other value than zero indicates that the port is operating in test mode. Refer to Chapter 7 of the USB Specification Revision 2.0 [3] for details on each 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. 0000 Not Enabled. 0001 J_STATE.

Table continues on the next page...

USBx_PORTSC field descriptions (continued)

Field	Description
	0010 K_STATE. 0011 SEQ_NAK. 0100 Packet. 0101 FORCE_ENABLE. 0110-1111 Reserved
15–14 PIC	Port indicator control. Control the link indicator signals. These signals are valid for host mode only. Refer to the USB Specification Revision 2.0 [3] for a description on how these bits are to be used. This field is output from the module on the USB port control signals for use by an external LED driving circuit. 00 Off 01 Amber 10 Green 11 Undefined
13 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.
12 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, etc. 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.
11–10 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. 00 SE0 10 J-state 01 K-state 11 Undefined
9 -	This field is reserved. Reserved, should be cleared
8 PR	Port reset. This field is zero if Port Power(PP) is zero. 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...

USBx_PORTSC field descriptions (continued)

Field	Description
	• This bit is a read only status bit. Device reset from the USB bus is also indicated in the USBSTS register. 1 Port is in reset. 0 Port is not in reset.
7 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. Device mode: 1 Port in suspend state. 0 Port not in suspend state. Default. In device mode this bit is a read-only status bit.
6 FPR	Force port resume. This bit is not-EHCI compatible. Host mode: • 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 remains a 1 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 msec or more, software must set this bit to one to drive resume signaling before clearing. The USB controller will set this bit to one if a J-to-K transition is detected while the port is in the Suspend state. The bit will be cleared when the device returns to normal operation. Also, when this bit transitions to a one because a J-to-K transition detected, USBSTS[PCI] is also set. Settings:

Table continues on the next page...

USBx_PORTSC field descriptions (continued)

Field	Description
	1 Resume detected/driven on port. 0 No resume (K-state) detected/driven on port.
5 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. 1 Over current detect. 0 No over current.
4 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. 1 Port currently in over-current condition. 0 Port not in over-current condition.
3 PEC	Port enable/disable change This field is zero if Port Power(PP) is zero. 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. Device mode: • The device port is always enabled. (This bit will be zero). 1 Port disabled. 0 No change.
2 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. Device mode: • The device port is always enabled. (This bit will be one).
1 CSC	Connect change status Host mode:

Table continues on the next page...

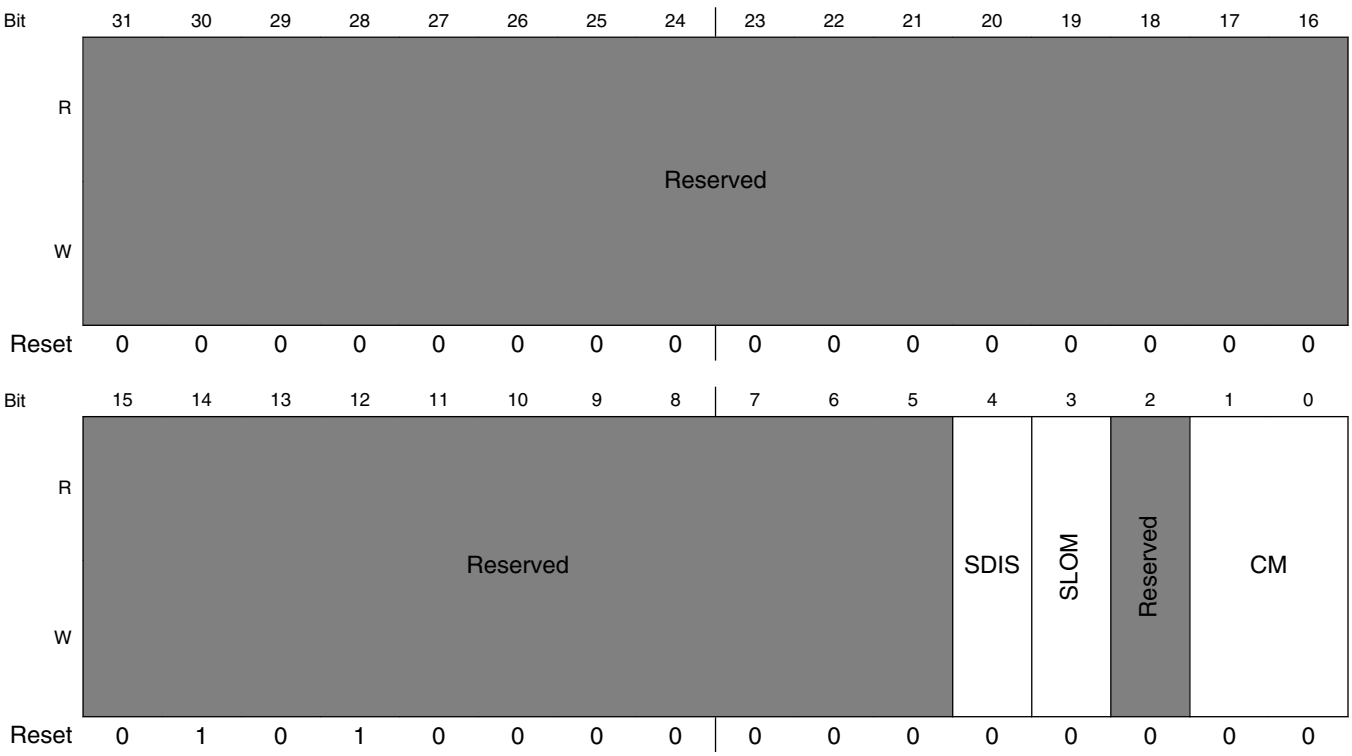
USBx_PORTSC field descriptions (continued)

Field	Description
	• 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 will be 'setting' an already-set bit (that is, the bit will remain set). Software clears this bit by writing a one to it. 1 Connect Status has changed. 0 No change. • This field is zero if Port Power(PP) is zero. Device mode: • This bit is undefined.
0 CCS	Current connect status Host mode: 1 Device is present 0 No device present. This field is zero if Port Power(PP) is zero in host mode. Device mode: 1 Attached 0 Not attached. A 1 indicates that the device successfully attached and is operating in either high-speed or full-speed as indicated by the High Speed Port bit in this register. A zero indicates that the device did not attach successfully or was forcibly disconnected by the software writing a zero to USBCMD[RS] (run bit). It does not state the device being disconnected or suspended.

17.5.18 USB device mode (USBx_USBMODE)

The USBMODE register is not defined in the EHCI specification. This register controls the operating mode of the module.

Address: Base address + 1A8h offset



USBx_USBMODE field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SDIS	Stream disable 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 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.

Table continues on the next page...

USBx_USBMODE field descriptions (continued)

Field	Description
	1 Active. 0 Inactive. 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 will be responded to with a NYET handshake when stream disable is active.
3 SLOM	Setup lockout mode. In device mode, this bit controls behavior of the setup lock mechanism. See Control Endpoint Operation Model. 1 Setup lockouts off. DCD requires use of setup data buffer tripwire in USBCMD (SUTW). 0 Setup lockouts on
2 -	This field is reserved. Reserved
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. 00 Idle 01 Reserved 10 Device controller. Reserved for host-only controller (USB1 and USB2) 11 Host controller. Reserved for device-only controller

17.5.19 Endpoint setup status (USBx_ENDPTSETUPSTAT)

The ENDPTSETUPSTAT register is not defined in the EHCI specification. This register contains the endpoint setup status. It is only used in device mode.

Address: Base address + 1ACh 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	Reserved																										ENDPTSETUPSTAT					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USBx_ENDPTSETUPSTAT field descriptions

Field	Description
31–6 -	This field is reserved. Reserved
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

Table continues on the next page...

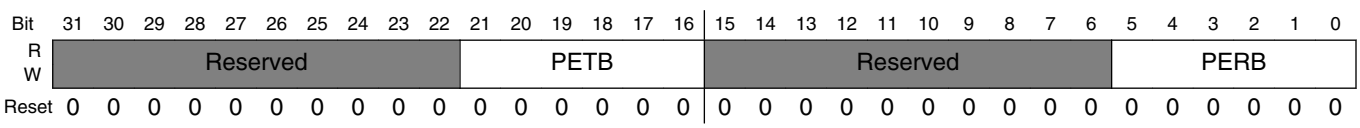
USBx_ENDPTSETUPSTAT field descriptions (continued)

Field	Description
	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.

17.5.20 Endpoint initialization (USBx_ENDPOINTPRIME)

This register is not defined in the EHCI specification. This register is used to initialize endpoints. It is only used in device mode.

Address: Base address + 1B0h offset



USBx_ENDPOINTPRIME field descriptions

Field	Description
31–22 -	This field is reserved. Reserved
21–16 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 will be momentarily set by hardware during hardware re-priming operations when a dTD is retired, and the dQH is updated.
15–6 -	This field is reserved. Reserved
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 will be momentarily set by hardware during hardware re-priming operations when a dTD is retired, and the dQH is updated.

17.5.21 Endpoint de-initialize (USBx_ENDPTFLUSH)

This register is not defined in the EHCI specification. This register is only used in device mode.

Address: Base address + 1B4h 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	Reserved										FETB						Reserved										FERB					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USBx_ENDPTFLUSH field descriptions

Field	Description
31–22 -	This field is reserved. Reserved
21–16 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.
15–6 -	This field is reserved. Reserved
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.

17.5.22 Endpoint status (USBx_ENDPTSTATUS)

This register is not defined in the EHCI specification. This register is only used in device mode.

Address: Base address + 1B8h offset

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

USBx_ENDPTSTATUS field descriptions

Field	Description
31–22 -	This field is reserved. Reserved, should be cleared
21–16 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 will be momentarily cleared by hardware during hardware endpoint re-priming operations when a dTD is retired, and the dQH is updated.
15–6 -	This field is reserved. Reserved, should be cleared
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. Note that these bits will be momentarily cleared by hardware during hardware endpoint re-priming operations when a dTD is retired, and the dQH is updated.

17.5.23 Endpoint complete (USBx_ENDPTCOMPLETE)

This register is not defined in the EHCI specification. This register is only used in device mode.

Address: Base address + 1BCh offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved										ETCE						Reserved										ERCE					
W											w1c																w1c					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USBx_ENDPTCOMPLETE field descriptions

Field	Description
31–22 -	This field is reserved. Reserved, should be cleared
21–16 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 will be 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.

Table continues on the next page...

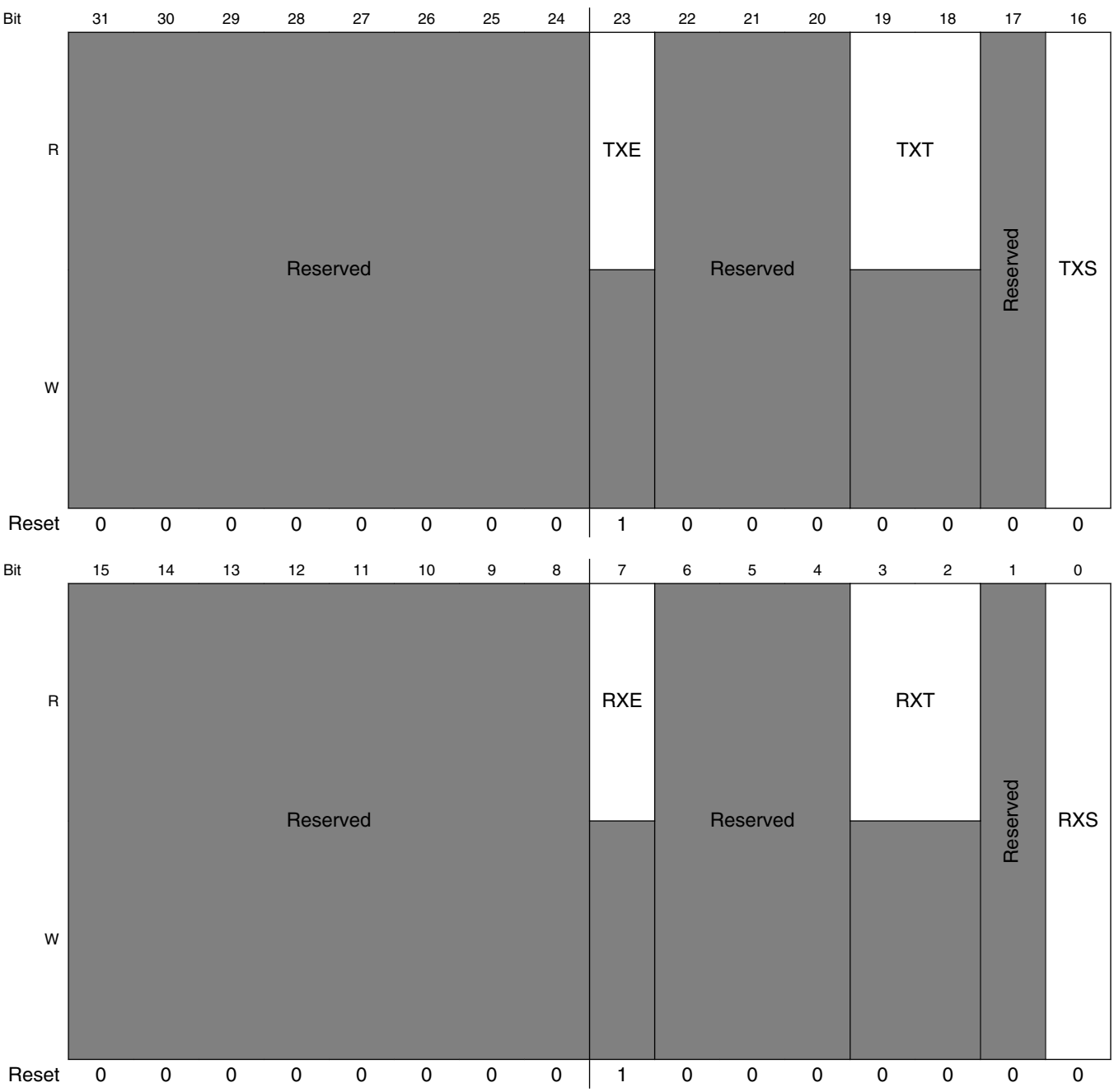
USBx_ENDPTCOMPLETE field descriptions (continued)

Field	Description
15–6 -	This field is reserved. Reserved, should be cleared
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 will be set simultaneously with the USBINT. Writing a one will clear the corresponding bit in this register. ERCE[5] corresponds to endpoint 5.

17.5.24 Endpoint control 0 (USB*x*_ENDPTCTRL0)

This register is not defined in the EHCI specification. Every device will implement endpoint 0 as a control endpoint.

Address: Base address + 1C0h offset



USBx_ENDPTCTRL0 field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23 TXE	TX endpoint enable. Endpoint zero is always enabled. 0 Disable 1 Enable
22–20 -	This field is reserved. Reserved
19–18 TXT	TX endpoint type. Endpoint zero is always a control endpoint (00).
17 -	This field is reserved. Reserved
16 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. 1 Endpoint stalled 0 Endpoint OK
15–8 -	This field is reserved. Reserved
7 RXE	RX endpoint enable. Endpoint zero is always enabled. 0 Disabled 1 Enabled
6–4 -	This field is reserved. Reserved
3–2 RXT	RX endpoint type. Endpoint zero is always a control endpoint (00).
1 -	This field is reserved. Reserved
0 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. 1 Endpoint stalled 0 Endpoint OK

17.5.25 Endpoint control n (USB*x*_ENDPTCTRL*n*)

These registers are not defined in the EHCI specification. There is an ENDPTCTRL *n* register of each endpoint in a device.

Address: Base address + 1C4h offset + (4d × i), where i=0d to 4d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved								TXE	TXR	TXI	Reserved	TXT		TXD	TXS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								RXE	RXR	RXI	Reserved	RXT		RXD	RXS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USB*x*_ENDPTCTRL*n* field descriptions

Field	Description
31–24 -	This field is reserved. Reserved, should be cleared
23 TXE	TX endpoint enable 0 Disabled 1 Enabled
22 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.
21 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. 0 PID sequencing enabled 1 PID sequencing disabled
20 -	This field is reserved. Reserved, should be cleared

Table continues on the next page...

USBx_ENDPTCTRLn field descriptions (continued)

Field	Description
19–18 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. 00 Control 01 Isochronous 10 Bulk 11 Interrupt
17 TXD	TX endpoint data source. This bit should always be written as 0, which selects the dual port memory/DMA engine as the source.
16 TXS	TX endpoint stall. This bit will be set automatically upon receipt of a SETUP request if this endpoint is not configured as a control endpoint. It will be cleared automatically upon receipt of a SETUP request if this endpoint is configured as a control endpoint. Software can write a one to this bit to force the endpoint to return a STALL handshake to the host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above. 0 Endpoint OK 1 Endpoint stalled
15–8 -	This field is reserved. Reserved, should be cleared
7 RXE	RX endpoint enable 0 Disabled 1 Enabled
6 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.
5 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. 1 PID sequencing enabled 0 PID sequencing disabled
4 -	This field is reserved. Reserved, should be cleared
3–2 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. 00 Control 01 Isochronous 10 Bulk 11 Interrupt
1 RXD	RX endpoint data sink. This bit should always be written as 0, which selects the dual port memory/DMA engine as the sink.
0 RXS	RX endpoint stall. This bit will be set automatically upon receipt of a SETUP request if this endpoint is not configured as a control endpoint. It will be cleared automatically upon receipt a SETUP request if this endpoint is configured as a control endpoint,

Table continues on the next page...

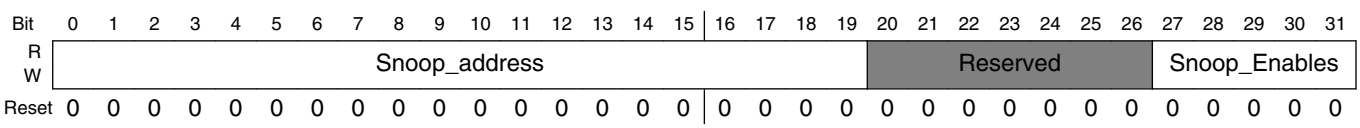
USBx_ENDPTCTRLn field descriptions (continued)

Field	Description
	Software can write a one to this bit to force the endpoint to return a STALL handshake to the host. It will continue to returning STALL until this bit is either cleared by software or automatically cleared as above, 1 Endpoint stalled 0 Endpoint OK

17.5.26 Snoop n (USBx_SNOOPn)

Note that 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 Kbytes to 2 Gbytes within each starting base address that is programmed by the core. The SNOOP *n* [20-26] are not used.

Address: Base address + 400h offset + (4d × i), where i=0d to 1d



USBx_SNOOPn field descriptions

Field	Description
0–19 Snoop_address	The starting base address for which transactions are snooped.
20–26 -	This field is reserved. Reserved, should be cleared
27–31 Snoop_Enables	0x00 Snooping disabled 0x0B 4-Kbyte snoop range starting at the value defined by SNOOP *n* [0-19] 0x0C 8-Kbyte snoop range starting at the value defined by SNOOP *n* [0-18] 0x0D 16-Kbyte snoop range starting at the value defined by SNOOP *n* [0-17] 0x0E 32-Kbyte snoop range starting at the value defined by SNOOP *n* [0-16] 0x0F 64-Kbyte snoop range starting at the value defined by SNOOP *n* [0-15] 0x10 128-Kbyte snoop range starting at the value defined by SNOOP *n* [0-14] 0x11 256-Kbyte snoop range starting at the value defined by SNOOP *n* [0-13]

Table continues on the next page...

USBx_SNOOPn field descriptions (continued)

Field	Description
	0x12 512-Kbyte snoop range starting at the value defined by SNOOP *n* [0-12] 0x13 1-Mbyte snoop range starting at the value defined by SNOOP *n* [0-11] 0x14 2-Mbyte snoop range starting at the value defined by SNOOP *n* [0-10] 0x15 4-Mbyte snoop range starting at the value defined by SNOOP *n* [0-9] 0x16 8-Mbyte snoop range starting at the value defined by SNOOP *n* [0-8] 0x17 16-Mbyte snoop range starting at the value defined by SNOOP *n* [0-7] 0x18 32-Mbyte snoop range starting at the value defined by SNOOP *n* [0-6] 0x19 64-M byte snoop range starting at the value defined by SNOOP *n* [0-5] 0x1A 31-Mbyte snoop range starting at the value defined by SNOOP *n* [0-4] 0x1B 256-Mbyte snoop range starting at the value defined by SNOOP *n* [0-3] 0x1C 512-Mbyte snoop range starting at the value defined by SNOOP *n* [0-2] 0x1D 1-Gbyte snoop range starting at the value defined by SNOOP *n* [0-1] 0x1E 2-Gbyte snoop range starting at the value defined by SNOOP *n* [0]

17.5.27 Age count threshold (USBx_AGE_CNT_THRESH)

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. It is only enabled if PRI_CTRL[pri_en] = 1. The threshold value is in units of platform clock /2 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 PRI_CTL[pri_en] = 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.

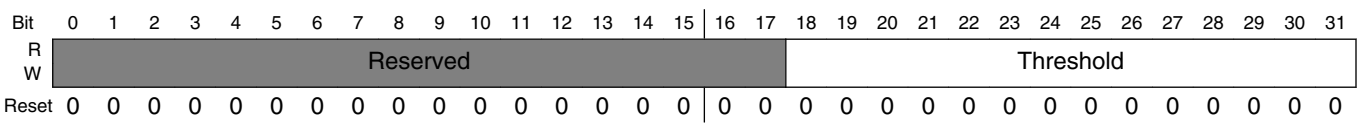
The AGE_CNT_THRESH is compared against the value of the aging counter during each clock cycle of the current transaction. If AGE_CNT_THRESH is equal to zero, an elevated priority is always chosen . If the aging counter is less than the AGE_CNT_THRESH value, default (low) priority is selected . If the aging counter is greater than or equal to the AGE_CNT_THRESH value and PRI_CTL[pri_en] = 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 PRI_CTRL[pri_en] to 1.
- 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.

Address: Base address + 408h offset



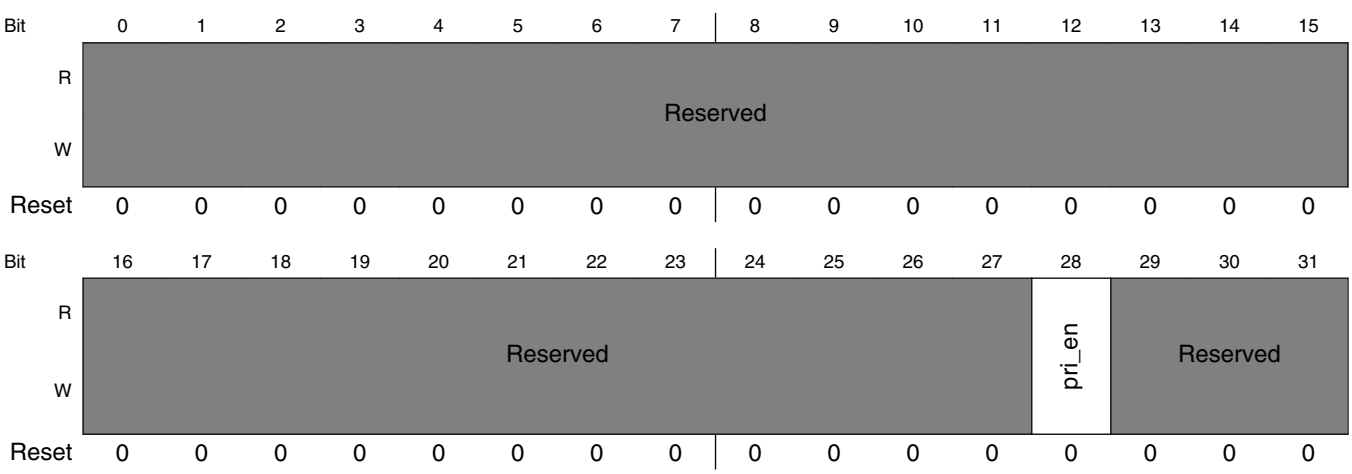
USBx_AGE_CNT_THRESH field descriptions

Field	Description
0–17 -	This field is reserved. Reserved, should be cleared
18–31 Threshold	Aging counter threshold value.

17.5.28 Priority control (USBx_PRI_CTRL)

The priority control register (PRI_CTRL) enables dynamic priority elevation as configured in the AGE_CNT_THRESH register. Note that this register uses big-endian byte ordering and is not defined in the EHCI specification.

Address: Base address + 40Ch offset



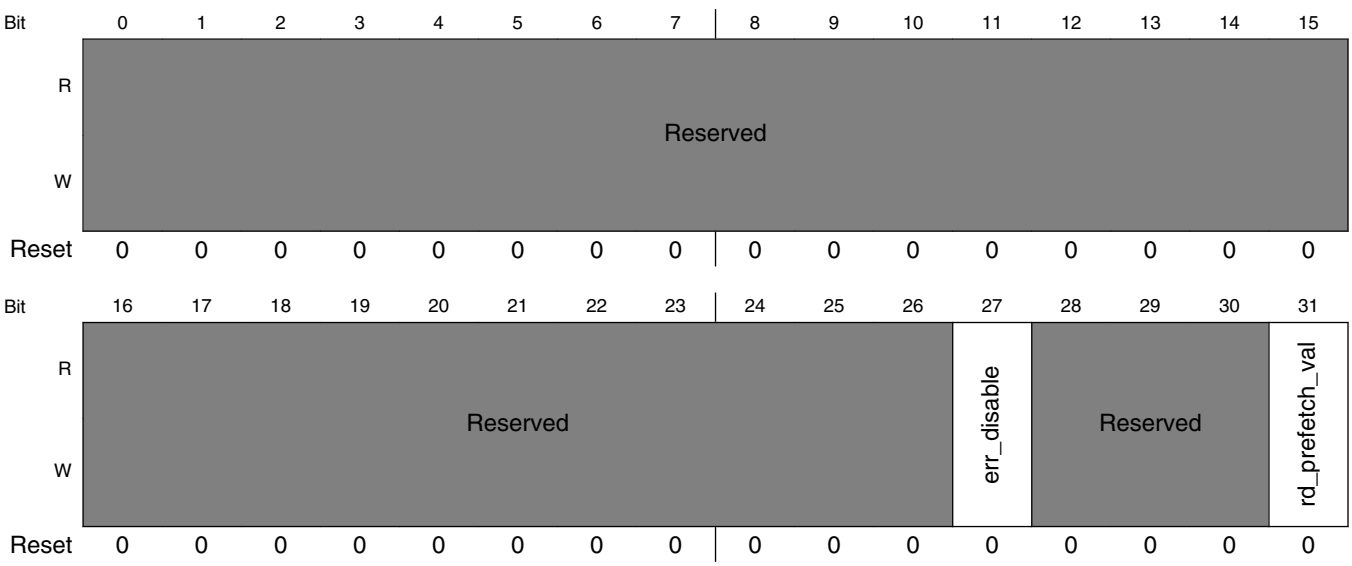
USBx_PRI_CTRL field descriptions

Field	Description
0–27 -	This field is reserved. Reserved
28 pri_en	High priority enable. 0 Normal operation. USB has default (low) priority. 1 High priority enabled.
29–31 -	This field is reserved. Reserved.

17.5.29 System interface control (USBx_SI_CTRL)

Note that 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.

Address: Base address + 410h offset



USBx_SI_CTRL field descriptions

Field	Description
0–26 -	This field is reserved. Reserved, should be cleared
27 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]. 0 enable 1 disable

Table continues on the next page...

USBx_SI_CTRL field descriptions (continued)

Field	Description
28–30 -	This field is reserved. Reserved, should be cleared
31 rd_prefetch_val	Selects whether 32 bytes or 64 bytes are fetched during burst read transactions at the system interface. When this input is LOW 64 bytes are fetched and when it is HIGH 32 bytes are fetched. The setting of rd_prefetch_val must match the setting of the larger of TXPBURST and RXPBURST fields in the BURSTSIZE register. If either of these fields is 64 bytes, then rd_prefetch_val must be left cleared. Otherwise, this value should be set. 0 64-byte fetch. 1 32-byte fetch.

17.5.30 Control (USBx_CONTROL)

Note that this register uses big-endian byte ordering and is not defined in the EHCI specification. The USB control register contains the general-purpose IP control register outputs.

Address: Base address + 500h offset

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved														PHY_CLK_VALID	WU_INT
W															w1c	w1c
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					PHY_CLK_SEL	UTMI_PHY_EN	Reserved						USB_EN	WU_INT_EN	Reserved
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USBx_CONTROL field descriptions

Field	Description
0–13 -	This field is reserved. Reserved
14 PHY_CLK_ VALID	Indicates whether the PHY clock is valid (read only). When in UTMI mode, this bit reflects the value of the UTMI PHY Clk Valid signal. 0 USB PHY clock is not valid 1 USB PHY clock is valid
15 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. 0 Normal operation or Low Power mode waiting for wakeup event 1 Low power wakeup event has occurred
16–20 -	This field is reserved. Reserved
21 PHY_CLK_SEL	Selects the source of the USB link controller transceiver clock. When cleared the UTMI PHY is the source of the clock. When set, the clock is sourced from the external ULPI PHY. 0 UTMI is clock source 1 Reserved
22 UTMI_PHY_EN	Enable the UTMI PHY. The UTMI PHY is reset when placed in the disable mode. 0 UTMI PHY disabled 1 UTMI PHY enabled
23–28 -	This field is reserved. Reserved
29 USB_EN	Used to enable the USB interface. 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 USB*n* _DIR is forced to appear asserted 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. 1 Normal operation. 0 Safe mode.
30 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. 0 System wakeup interrupt disabled 1 System wakeup interrupt enabled
31 -	This field is reserved. Reserved

17.6 Functional description

The USB DR module can be broken down into functional sub-blocks, which are described below.

17.6.1 System Interface

The system interface block contains all the control and status registers that allow a processor to interface to the USB 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.

17.6.2 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.

17.6.3 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.

17.6.4 PHY Interface

The UTMI interface connects internally to an on-chip PHY . The USB module interfaces to any UTMI-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. See table below for more information.

Table 17-116. Supported PHY Interfaces

PHY	Function
UTMI	Host/Device

17.7 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 that 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.

17.7.1 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.

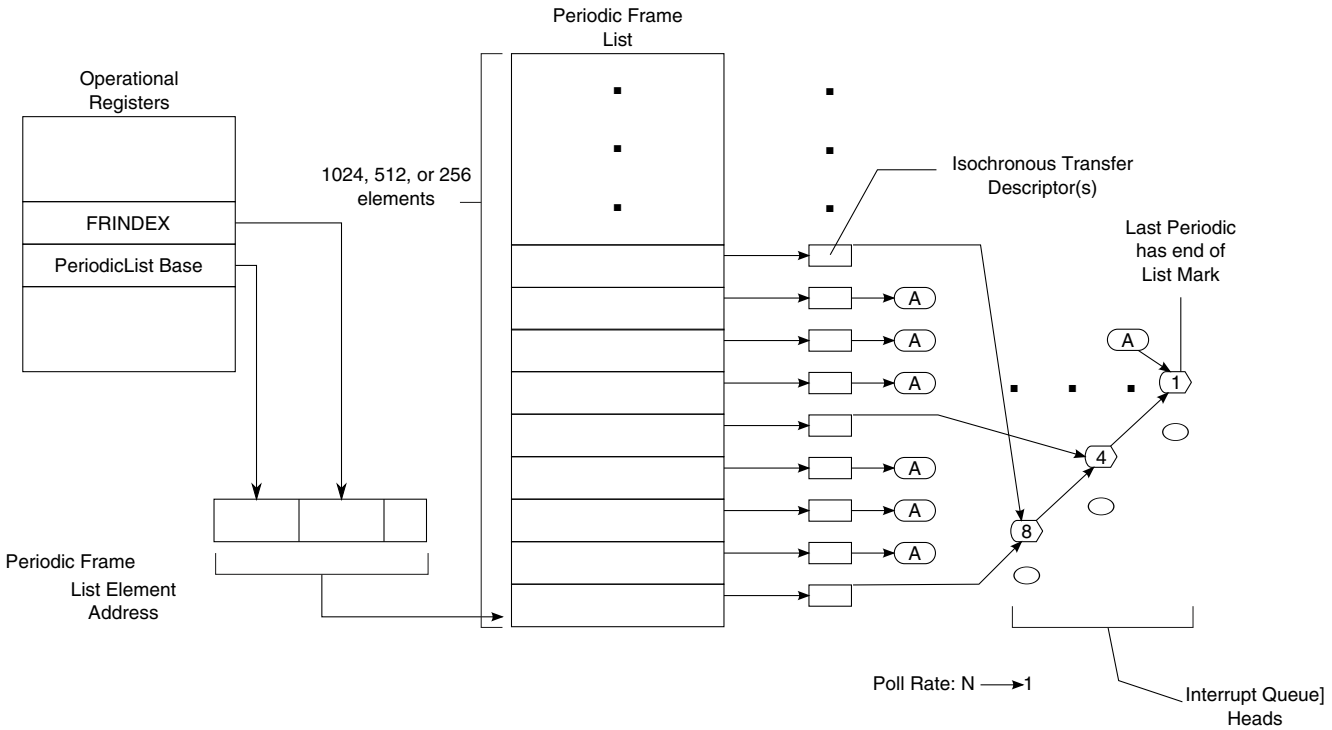


Figure 17-113. Periodic schedule organization

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 17-117. 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 17-118. Typ field encodings

Typ	Description
00	Isochronous transfer descriptor
01	Queue head
10	Split transaction isochronous transfer descriptor
11	Frame span traversal node

17.7.2 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.

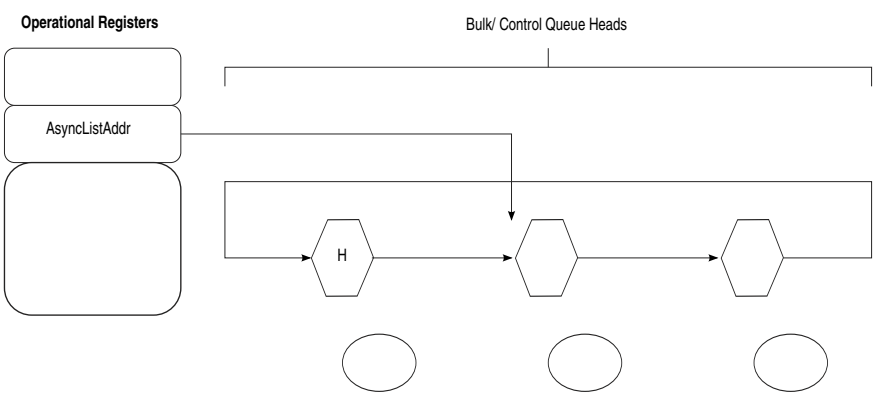


Figure 17-114. Asynchronous schedule organization

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.

17.7.3 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 17-119. 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
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.

17.7.3.1 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 17-120. 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.

17.7.3.2 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 17-121. 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.

17.7.3.3 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 17-122. 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 17-123. 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 17-124. 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 17-125. 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.

17.7.4 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.

Table 17-126. 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.

17.7.4.1 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 17-127. 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.
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.

17.7.4.2 siTD Endpoint Capabilities/Characteristics

DWords 1 and 2 specify static information about the full-speed endpoint, the addressing of the parent Companion Controller, and micro-frame scheduling control.

Table 17-128. 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
22-16	Hub Address	This field holds the device address of the companion controllers' hub.
15-12	-	Reserved
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.

Table 17-129. Micro-Frame Schedule Control

Bits	Name	Description
31-16	-	Reserved
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 micro-frames the host controller should execute complete-split transactions. When the criteria for using this field is met, an all-zeros value has undefined behavior. The host controller uses the value of the three low-order bits of the FRINDEX register to index into this bit field. If the FRINDEX register value indexes to a position where the 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 micro-frames the host controller should execute start-split transactions. The host controller uses the value of the three low-order bits of the FRINDEX register to index into this bit field. If the FRINDEX register value indexes to a position where the 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.

17.7.4.3 siTD Transfer State

DWords 3-6 manage the state of the transfer.

Table 17-130. siTD Transfer Status and Control

Bits	Name	Description	
31	ioc	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.	
30	P	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).	
29-26	-	Reserved	
25-16	Total Bytes to Transfer	This field is initialized by software to the total number of bytes expected in this transfer. Maximum value is 1023 (3FFh)	
15-8	µFrame C-prog-mask	Split complete progress mask. This field is used by the host controller to record which split-completes have been executed.	
7-0	Status	This field records the status of the transaction executed by the host controller for this slot. This field is a bit vector with the following encoding:	
		Status Bits	Definition
		7	Active. Set by software to enable the execution of an isochronous split transaction by the host controller.
		6	ERR. Set by the host controller when an ERR response is received from the companion controller.
		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, etc.). This bit will only be set for IN transactions.
		2	Missed micro-frame. The host controller detected that a host-induced hold- off caused the host controller to miss a required complete-split transaction.
		1	Split transaction state (SplitXstate). The bit encodings are: 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

17.7.4.4 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.

Table 17-131. 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).

Table 17-132. 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
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.

17.7.4.5 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.

Table 17-133. siTD Back Link Pointer

Bits	Name	Description
31-5	Back Pointer	A physical memory pointer to an siTD
4-1	-	Reserved
0	T	Terminate 0 siTD Back Pointer field is valid 1 siTD Back Pointer field is not valid

17.7.5 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 20480 (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.

Table 17-134. 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[2]																			ioc	C_Page[1][2]			Cerr[1][2]		PID Code			Status[1][2]					0x08
Buffer Pointer (Page 0)																								Current Offset[1][2]									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.
2. Host controller read/write; all others read-only.

Queue element transfer descriptors must be aligned on 32-byte boundaries.

17.7.5.1 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 17-135. 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.
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

17.7.5.2 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 will always use this pointer when the current qTD is retired due to short packet. The table below describes the alternate qTD next element transfer pointer.

Table 17-136. 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

17.7.5.3 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 below reference fields are defined in the queue head. See Queue Head, for more information on these fields.

Table 17-137. 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.	
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 set-up, the host controller decrements the count and writes it back to the qTD if the transaction fails. If the counter counts from one to zero, the host controller marks the qTD inactive, sets the Halted bit to a one, and error status bit for the error that caused Cerr to decrement to zero. An interrupt will be generated if USBINTR[UEE] is set. If the host controller driver (HCD) software programs this field to zero during set-up, the host controller will not count errors for this qTD and there will be 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,

Table continues on the next page...

Table 17-137. qTD Token (DWord 2) (continued)

Bits	Name	Description	
			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) 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 micro-frame. This bit is ignored unless the QH[EPS] field indicates a full- or low-speed endpoint and the queue head is in the periodic list. This bit is set when the host controller detected that a host-induced hold-off caused the host controller to miss a required complete-split transaction. If the host controller sets this bit to a one, then it remains a one for the duration of the transfer.
		1	Split transaction state (SplitXstate). This bit is ignored by the host controller unless the QH[EPS] field indicates a full- or low-speed endpoint. When a full- or low-speed device, the host controller uses this bit to track the state of the split- transaction. The functional requirements of the host controller for managing this state bit and the split transaction protocol depends on whether the endpoint is in the periodic or asynchronous schedule. The bit encodings are:

Table continues on the next page...

Table 17-137. qTD Token (DWord 2) (continued)

Bits	Name	Description	
			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.
		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.

17.7.5.4 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.

Table 17-138. 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.

17.7.6 Queue Head

Figure below shows the queue head structure, which must be aligned on a 64-byte boundary.

Table 17-139. 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
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[1]
Current qTD Pointer[2]																											00000					0x0C
Next qTD Pointer[2]																											0000				T[2]	0x10[3]
Alternate Next qTD Pointer[2]																											NakCnt[2]				T[2]	0x14[3,4]
dt[5]	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 0x04 through 0x0B 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 0x14 through 0x27 contain the transfer results.
5. Host controller read/write; all others read-only.

17.7.6.1 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.

Table 17-140. 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

Table continues on the next page...

Table 17-140. Queue Head DWord 0 (continued)

Bits	Name	Description
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. 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.

17.7.6.2 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 17-141. 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.

Table continues on the next page...

Table 17-141. Endpoint characteristics: Queue head DWord 1 (continued)

Bits	Name	Description
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.
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 17-142. 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.

Table continues on the next page...

Table 17-142. Endpoint capabilities: Queue head DWord 2 (continued)

Bits	Name	Description
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 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.

17.7.6.3 Queue Head 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 will never follow the next transfer queue element or alternate queue element pointers unless it is actively attempting to advance the queue. For the duration of the transfer, the host controller keeps the incremental status of the transfer in the overlay area. When the transfer is complete, the results are written back to the original queue element.

The DWord3 of a queue head contains a pointer to the source qTD currently associated with the overlay. The host controller uses this pointer to write back the overlay area into the source qTD after the transfer is complete.

Table 17-143. 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.

Table 17-144. 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.

17.7.7 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 17-145. 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.

17.7.7.1 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 17-146. 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.

17.7.7.2 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 17-147. 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.

17.8 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.

17.8.1 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 internal UTMI PHY, the following initialization sequence is required:

1. After power-on reset, the UTMI PHY will be in disabled state and the PLL will be held reset.
2. Set the USB_CONTROL[PHY_CLK_SEL] bits to select the UTMI PHY as the source of USB controller PHY clock.

3. Set the USB_CONTROL[UTMI_PHY_EN] to enable the UTMI PHY and release the PLL.
4. Wait (approx 10ms) for PHY clock to become valid. This can be determined by polling the CONTROL[PHY_CLK_VALID] status bit.

Once the PHY clock is valid 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.

17.8.2 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) bits.

17.8.3 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 USB 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 17-148 for summary of behavior for over-current detection when the controller is halted (suspended from a device component point of view).

17.8.4 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.

17.8.4.1 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 micro-frame or frame boundary occurs. If evaluated immediately, the port is not suspended until the current transaction (if one is executing) completes. Therefore, there may be several micro-frames of activity on the port until the host controller evaluates the Suspend bit. The host controller must evaluate the Suspend bit at least every frame boundary.

System software can initiate a resume on 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.

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 17-148. 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 WKDSCNNT_E bit, 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 WKDSCNNT_E bit, 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 WKCNNT_E 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]

Table continues on the next page...

Table 17-148. Behavior During Wake-up Events (continued)

Port Status and Signaling Type	Signaled Port Response	Device State	
		D0	not D0
Port is not connected and the port's WKCNNT_E 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 WKOC_E 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 WKOC_E 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.

[2] PME# asserted if enabled (Note: PME Status must always be set).

[3] PME# not asserted.

17.8.5 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 PERIODICLISTBASE register. See Frame list base address (USB_PERIODICLISTBASE), for more information. 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 micro-frame, if the periodic schedule is enabled 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 Figure 17-115). 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 micro-frame.

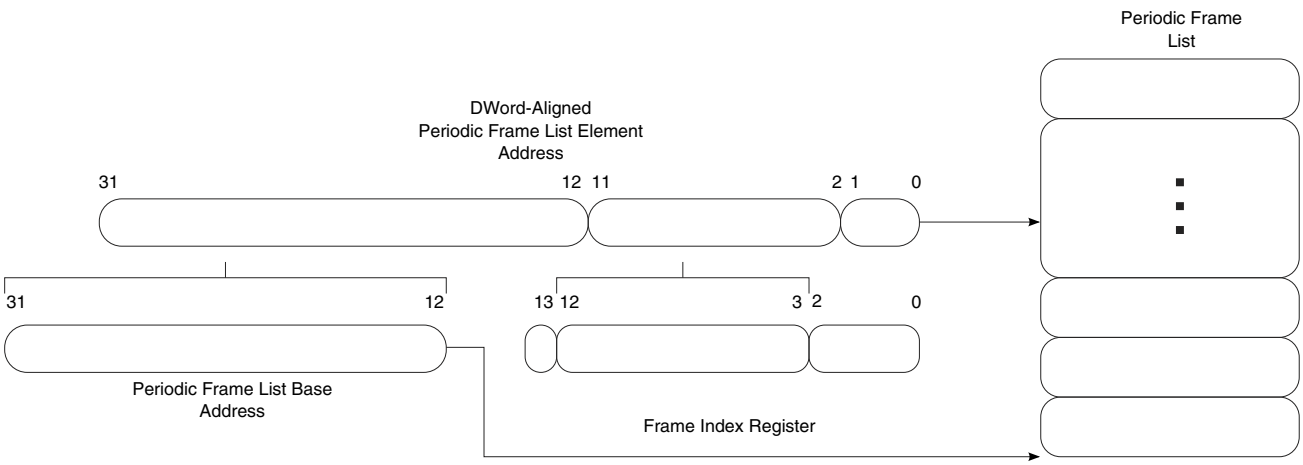


Figure 17-115. Derivation of Pointer into Frame List Array

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 below.

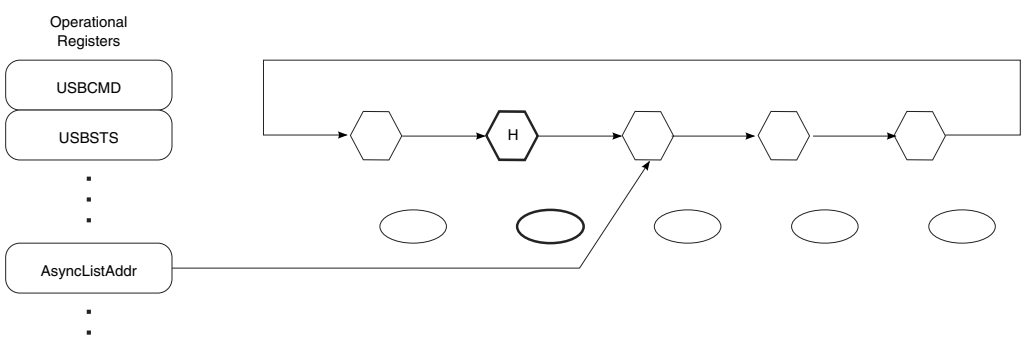


Figure 17-116. General Format of Asynchronous Schedule List

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.

17.8.6 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.

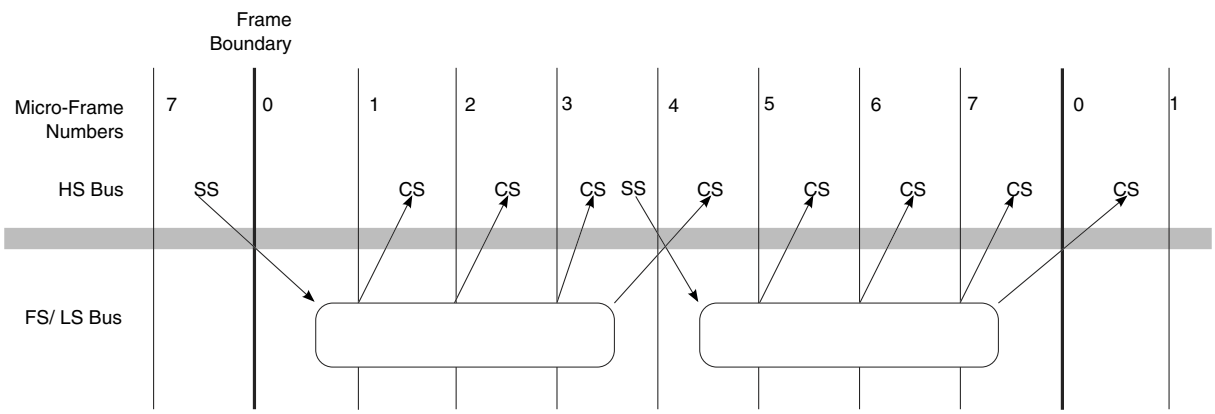


Figure 17-117. Frame boundary relationship between HS bus and FS/LS bus

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.

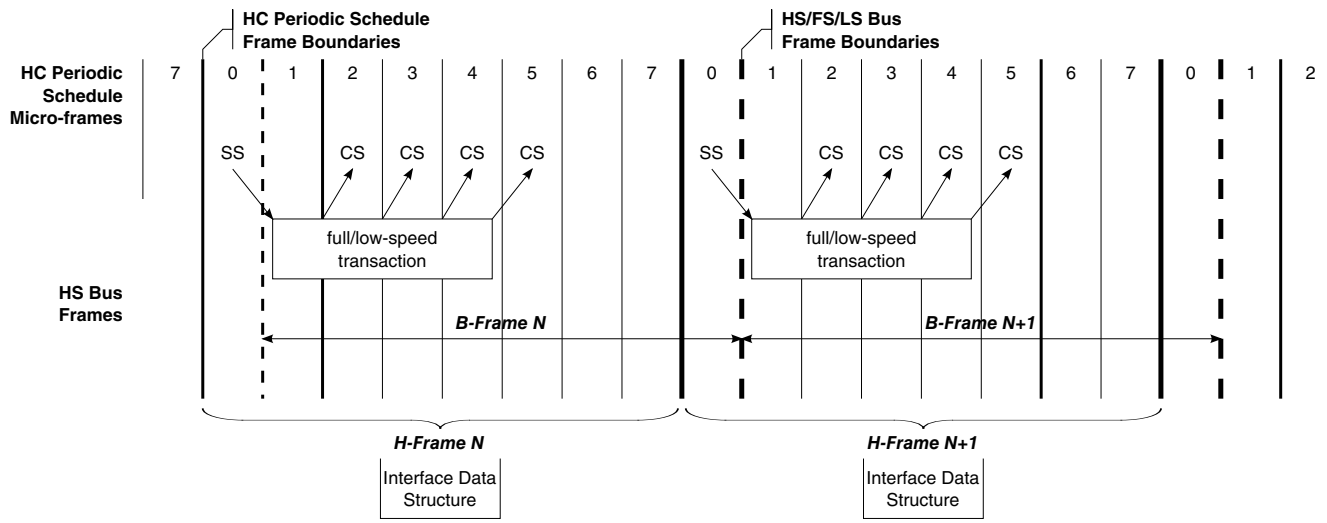


Figure 17-118. Relationship of periodic schedule frame boundaries to bus frame boundaries

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 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. FRINDEX, provides the requirements that software should adhere when writing a new value in FRINDEX.

Table 17-149. 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

17.8.7 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.

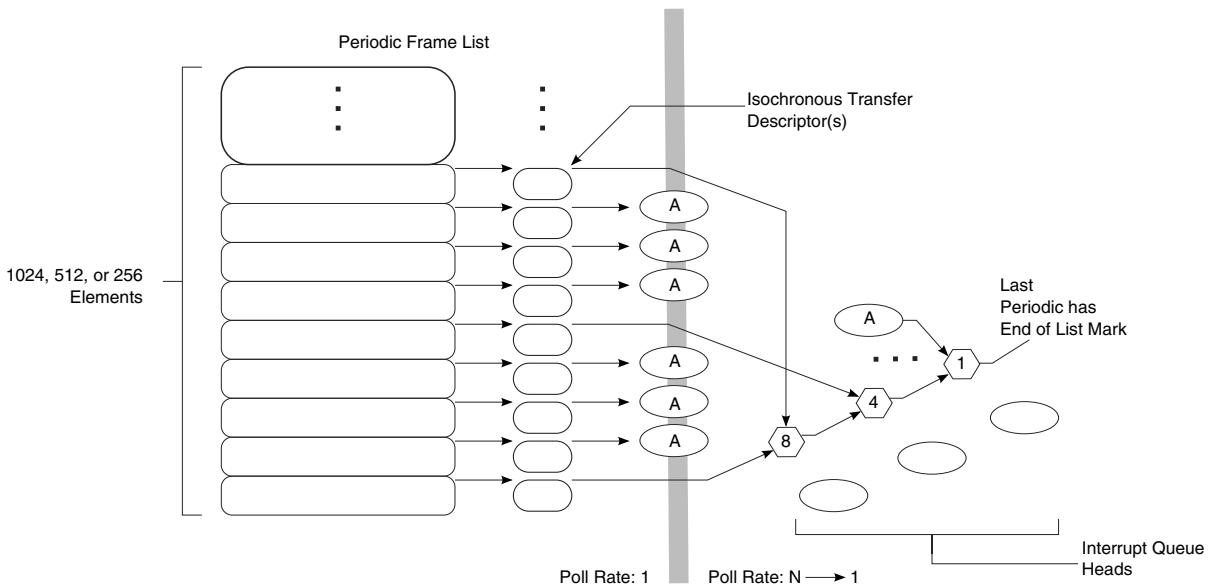


Figure 17-119. Example periodic schedule

17.8.8 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:

- The first field is the Next Link Pointer. This field is for schedule linkage purposes only.
- Transaction description array. This area is an eight-element array. Each element represents control and status information for one micro-frame's worth of transactions for a single high-speed isochronous endpoint.

- The buffer page pointer array is a 7-element array of physical memory pointers to data buffers. These are 4K aligned pointers to physical memory.
- Endpoint capabilities. This area utilizes the unused low-order 12 bits of the buffer page pointer array. The fields in this area are used across all transactions executed for this iTD, including endpoint addressing, transfer direction, maximum packet size and high-bandwidth multiplier.

17.8.8.1 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.

17.8.8.2 Software Operational Model for iTDs

A client buffer request to an isochronous endpoint may span 1 to N micro-frames. When N is larger than one, system software may have to use multiple iTDs to read or write data with the buffer (if N is larger than eight, it must use more than one iTD).

Figure 17-120 illustrates the simple model of how a client buffer is mapped by system software to the periodic schedule (that is, the periodic frame list and a set of iTDs). On the right is the client description of its request. The description includes a buffer base address plus additional annotations to identify which portions of the buffer should be used with each bus transaction. In the middle is the iTD data structures used by the system software to service the client request. Each iTD can be initialized to service up to 24 transactions, organized into eight groups of up to three transactions each. Each group maps to one micro-frame's worth of transactions. The EHCI controller does not provide per-transaction results within a micro-frame. It treats the per-micro-frame transactions as a single logical transfer. On the left is the host controller's frame list. System software establishes references from the appropriate locations in the frame list to each of the appropriate iTDs. If the buffer is large, then system software can use a small set of iTDs to service the entire buffer. System software can activate the transaction description records (contained in each iTD) in any pattern required for the particular data stream.

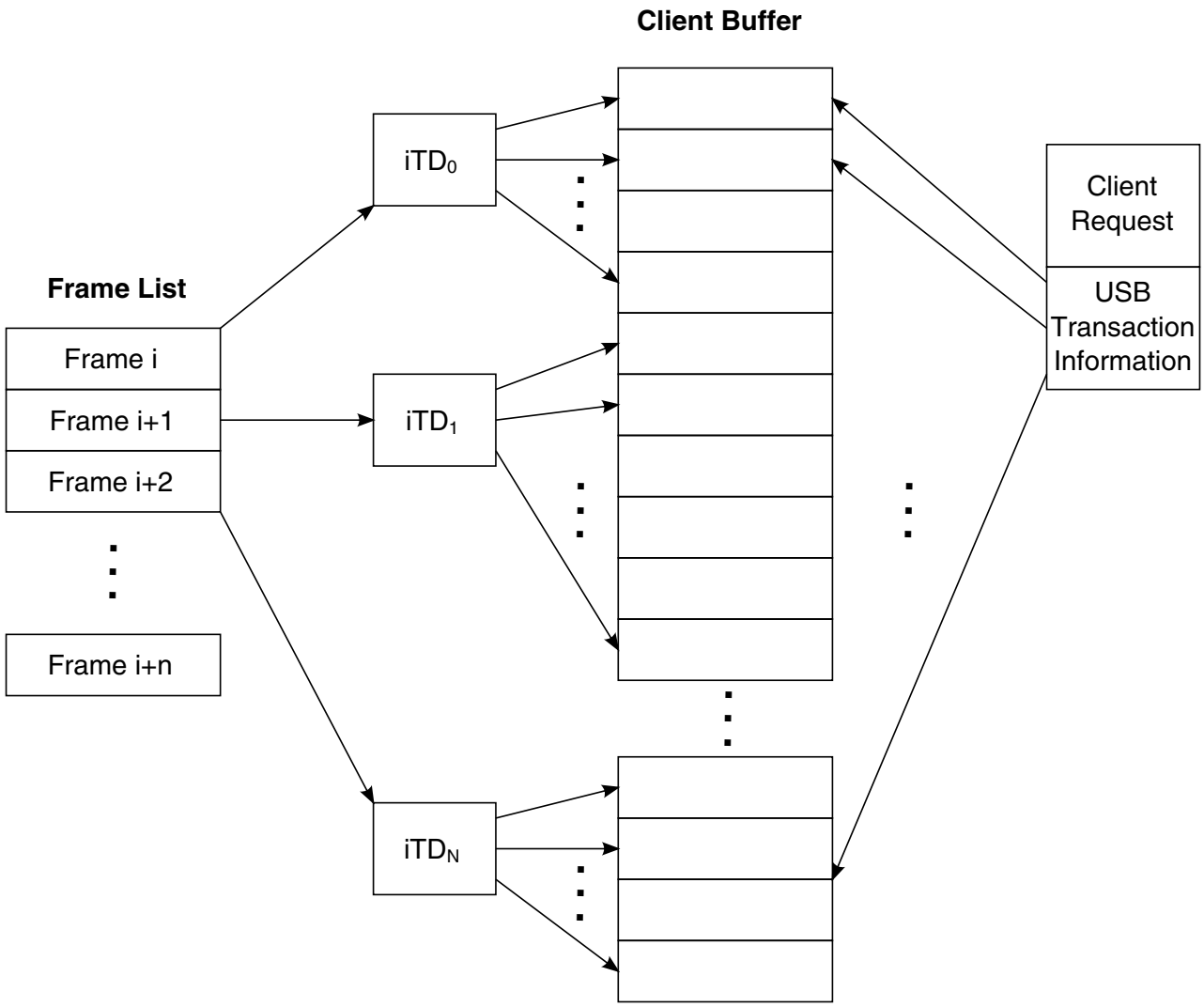


Figure 17-120. Example Association of iTDs to Client Request Buffer

As noted above, the client request includes a pointer to the base of the buffer and offsets into the buffer to annotate which buffer sections are to be used on each bus transaction that occurs on this endpoint. System software must initialize each transaction description in an iTD to ensure it uses the correct portion of the client buffer. For example, for each transaction description, the PG field is set to index the correct physical buffer page pointer and the Transaction Offset field is set relative to the correct buffer pointer page (for example, the same one referenced by the PG field). When the host controller executes a transaction it selects a transaction description record based on FRINDEX[2-0]. It then uses the current Page Buffer Pointer (as selected by the PG field) and concatenates to the transaction offset field. The result is a starting buffer address for the transaction. As the host controller moves data for the transaction, it must watch for a page wrap condition and properly advance to the next available Page Buffer Pointer. System software must not use the Page 6 buffer pointer in a transaction description where the length of the transfer

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.

17.8.8.2.1 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.

17.8.9 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 occur:

- 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 2. 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.

17.8.9.1 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
```

17.8.9.2 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, 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 its 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 its 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 its 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.

Figure 17-121 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).

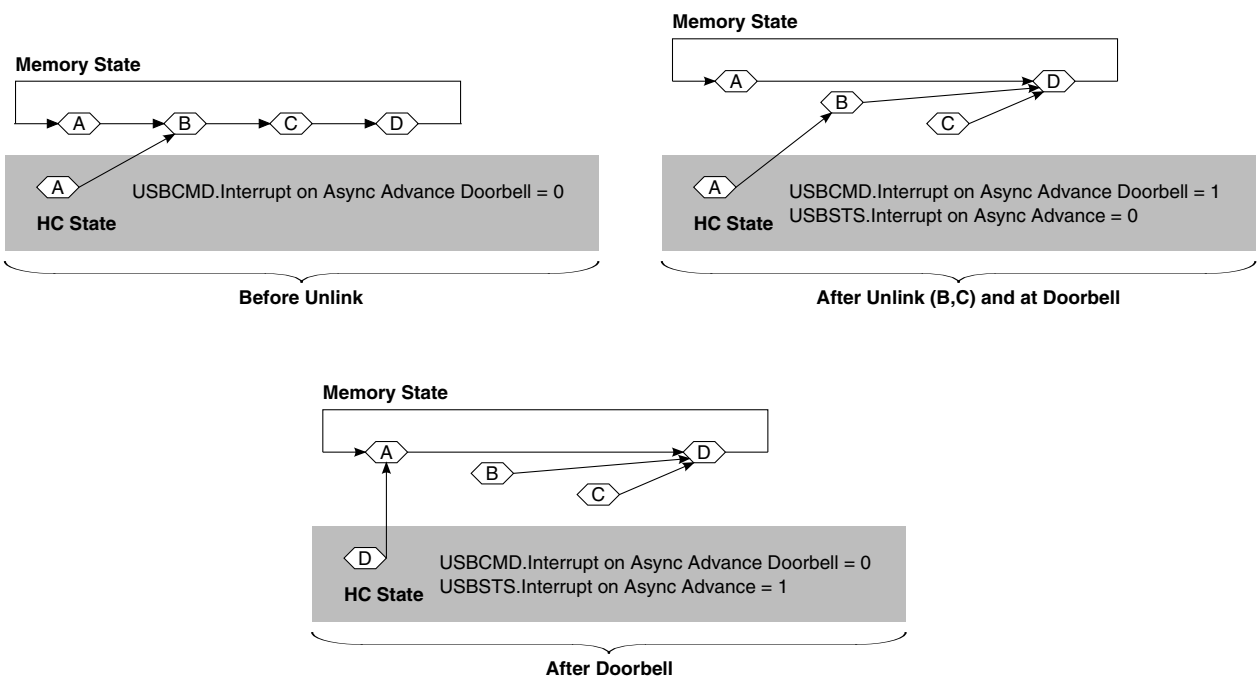


Figure 17-121. Generic Queue Head Unlink Scenario

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

17.8.9.3 Empty asynchronous schedule detection

EHCI uses two bits to detect when the asynchronous schedule is empty.

The queue head data structure (see Table 1) 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.

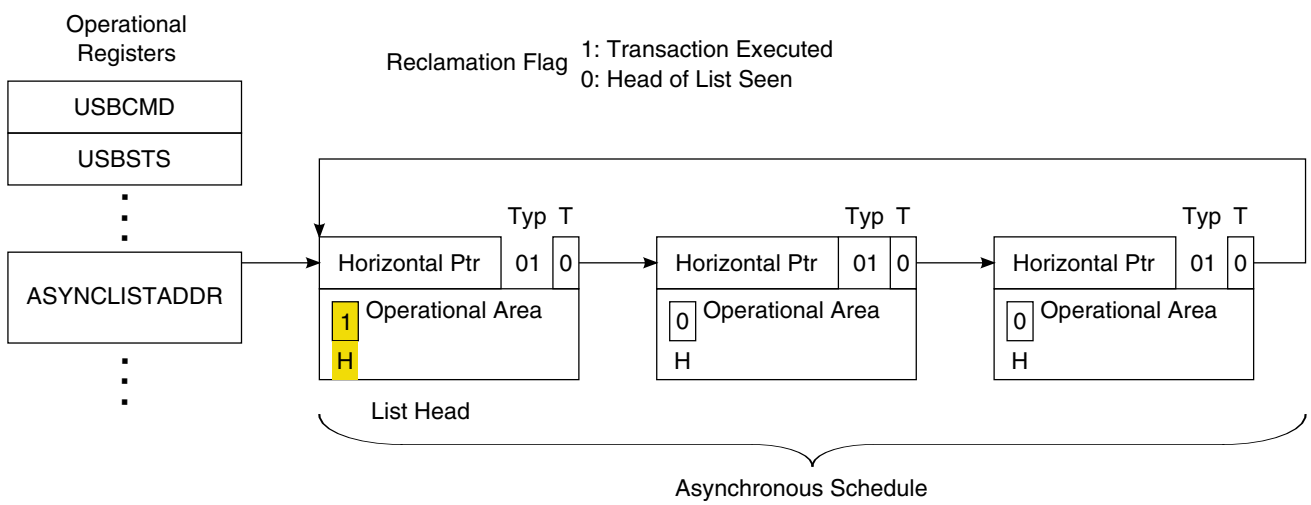


Figure 17-122. Asynchronous schedule list with annotation to mark head of list

17.8.9.4 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.

17.8.9.5 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.

17.8.10 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.

17.8.10.1 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.

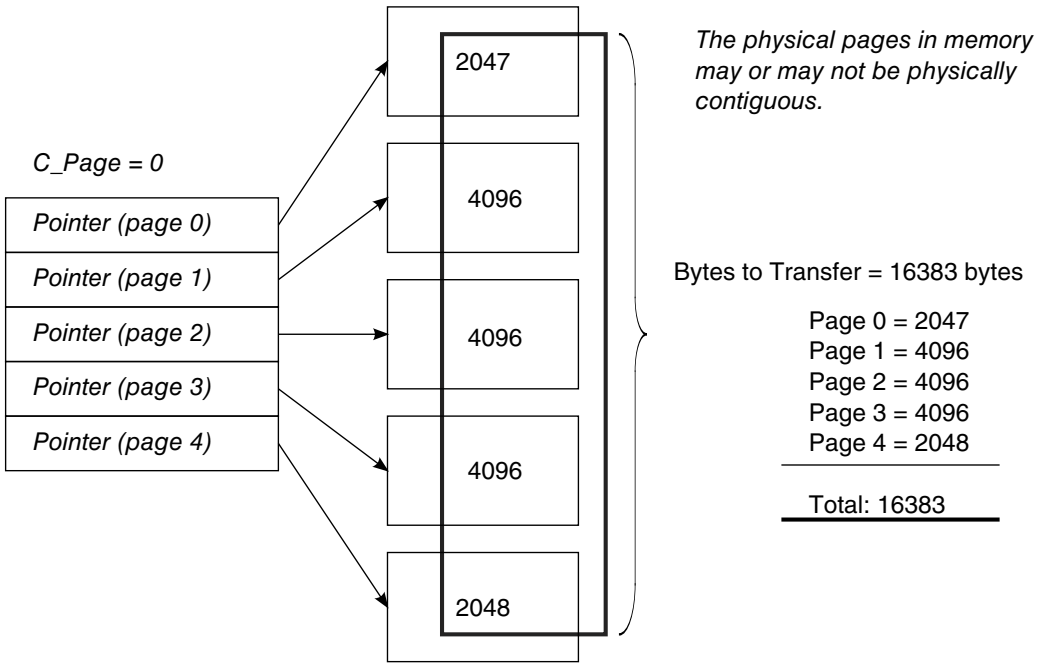


Figure 17-123. Example mapping of qTD buffer pointers to buffer pages

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.

17.8.10.2 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 17-150. 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.

17.8.10.3 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.

17.8.11 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 17-137). 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

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 17-151. 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 17-137. The defined ping protocol allows the host to be imprecise on the initialization of the ping protocol (that is, start in Do OUT when we don't know whether there is space on the device or not). The host controller manages the Ping State bit. System software sets the initial value in the queue head when it initializes a queue head. The host controller preserves the Ping State bit across all queue advancements. This means that when a new qTD is written into the queue head overlay area, the previous value of the Ping State bit is preserved.

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.

17.8.12 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.

17.8.12.1 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.

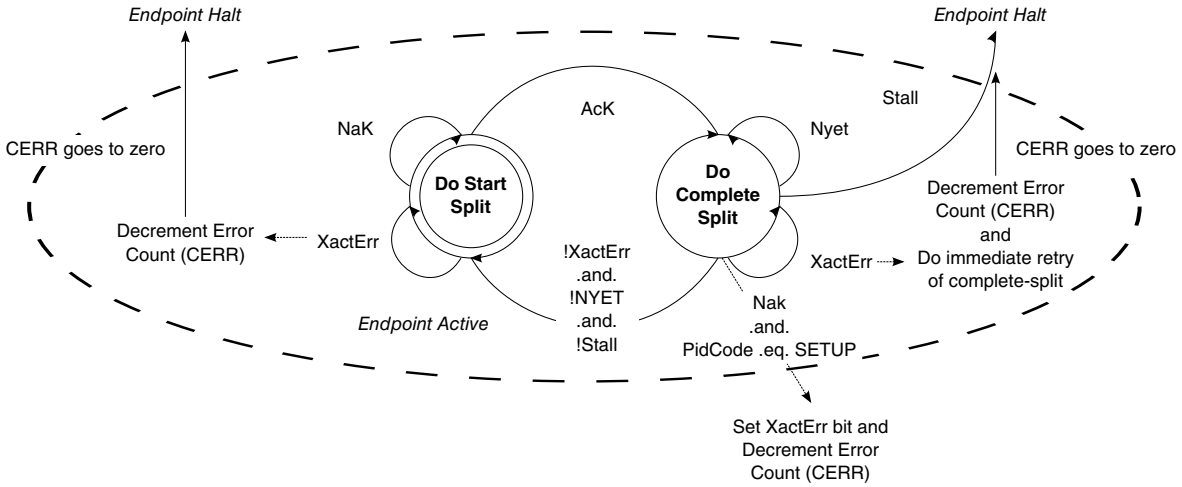


Figure 17-124. Host Controller Asynchronous Schedule Split-Transaction State Machine

17.8.12.1.1 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.

17.8.12.1.2 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 responses are expected:

- DATA0/1. On reception of data, the host controller ensures the PID matches the expected data toggle and checks CRC. If the packet is good, the host controller advances the state of the transfer (for example, 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 responses are expected:

- ACK. The target endpoint accepted the data, so the host controller must advance the state of the transfer. The Current Offset field is incremented by Maximum Packet

Length or Bytes to Transfer, whichever is less. The 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.

17.8.12.2 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.

17.8.12.2.1 Split Transaction Scheduling Mechanisms for Interrupt

Full- and low-speed Interrupt queue heads have an EPS field indicating full- or low-speed and have a non-zero S-mask field. The host controller can detect this combination of parameters and assume the endpoint is a periodic endpoint. Low- and full-speed interrupt queue heads require the use of the split transaction protocol. The host controller sets the Endpoint Type (ET) field in the split token to indicate the transaction is an interrupt. These transactions are managed through a transaction translator's periodic pipeline. Software should not set these fields to indicate the queue head is an interrupt unless the queue head is used in the periodic schedule.

System software manages the per/transaction translator periodic pipeline by budgeting and scheduling exactly during which micro-frames the start-splits and complete-splits for each endpoint will occur. The characteristics of the transaction translator are such that the high-speed transaction protocol must execute during explicit micro-frames, or the data or response information in the pipeline is lost. 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 micro-frames where software can schedule start-splits and complete splits (respectively).

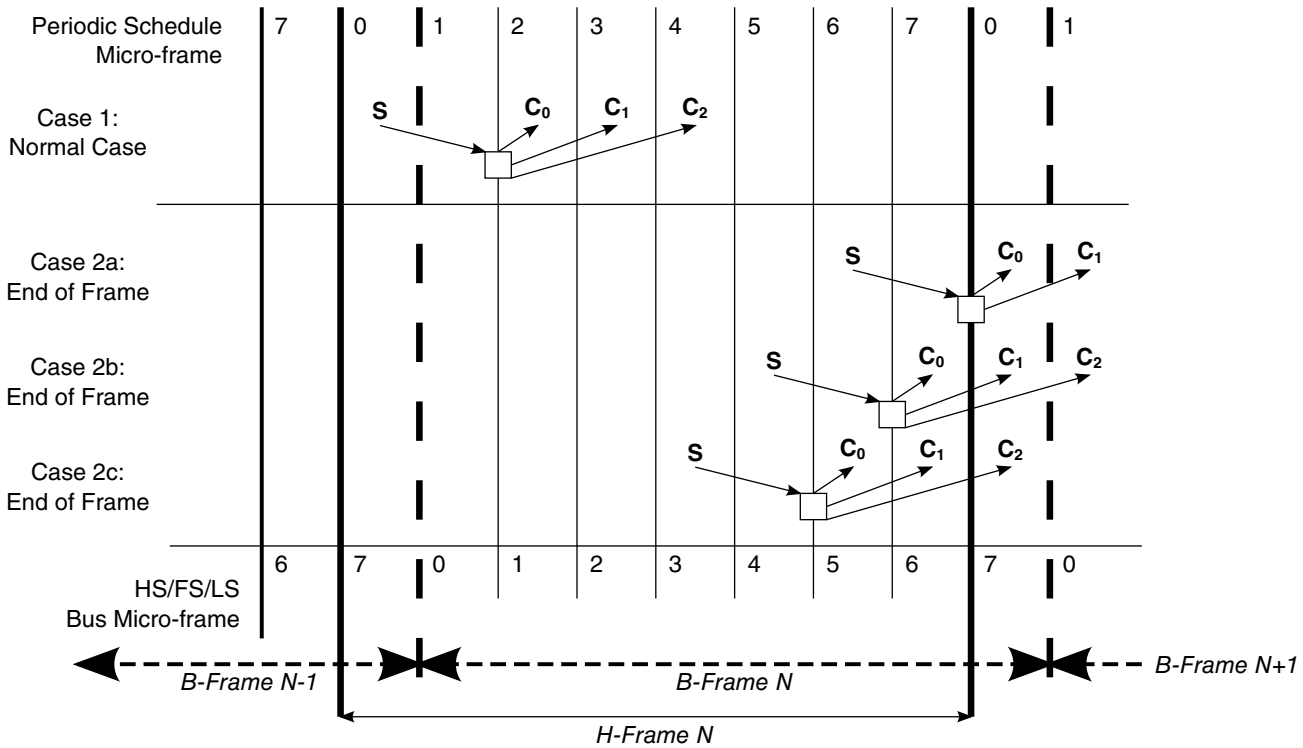


Figure 17-125. Split Transaction, Interrupt Scheduling Boundary Conditions

The scheduling cases are:

- Case 1: The normal scheduling case is where the entire split transaction is completely bounded by a frame (H-Frame in this case).
- Case 2a through Case 2c: The USB 2.0 hub pipeline rules states clearly, when and how many complete-splits must be scheduled to account for earliest to latest execution on the full/low-speed link. The complete-splits may span the H-Frame boundary when the start-split is in micro-frame 4 or later. When this occurs, the H-Frame to B-Frame alignment requires that the queue head be reachable from consecutive periodic frame list locations. System software cannot build an efficient schedule that satisfies this requirement unless it uses FSTNs. Figure below illustrates the general layout of the periodic schedule.

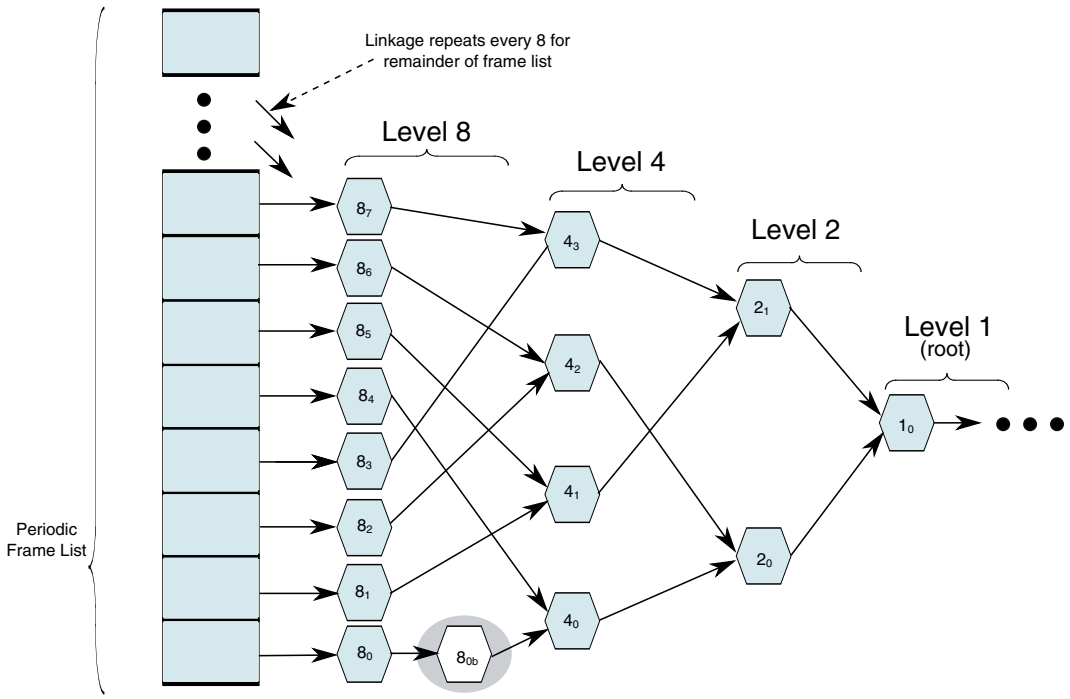


Figure 17-126. General Structure of EHCI Periodic Schedule Utilizing Interrupt Spreading

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 17-137). This bit is used to track the current state of the split transaction.

- Frame S-mask. This is a bit-field where-in system software sets a bit corresponding to the micro-frame (within an H-Frame) that the host controller should execute a start-split transaction. This is always qualified by the value of the SplitXState bit in the Status field of the queue head. For example, referring to Figure 17-125, 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 micro-frame as indicated by FRINDEX[2-0] is 0, then execute a start-split transaction.
- Frame C-mask. This is a bit-field where system software sets one or more bits corresponding to the micro-frames (within an H-Frame) that the host controller should execute complete-split transactions. The interpretation of this field is always qualified by the value of the SplitXState bit in the Status field of the queue head. For example, referring to Figure 17-125, 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 micro-frame as indicated by FRINDEX[2-0] is 2, 3, or 4, then execute a complete-split transaction. It is software's responsibility to ensure that the translation between H-Frames and B-Frames is correctly performed when setting bits in S-mask and C-mask.

17.8.12.2.2 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.

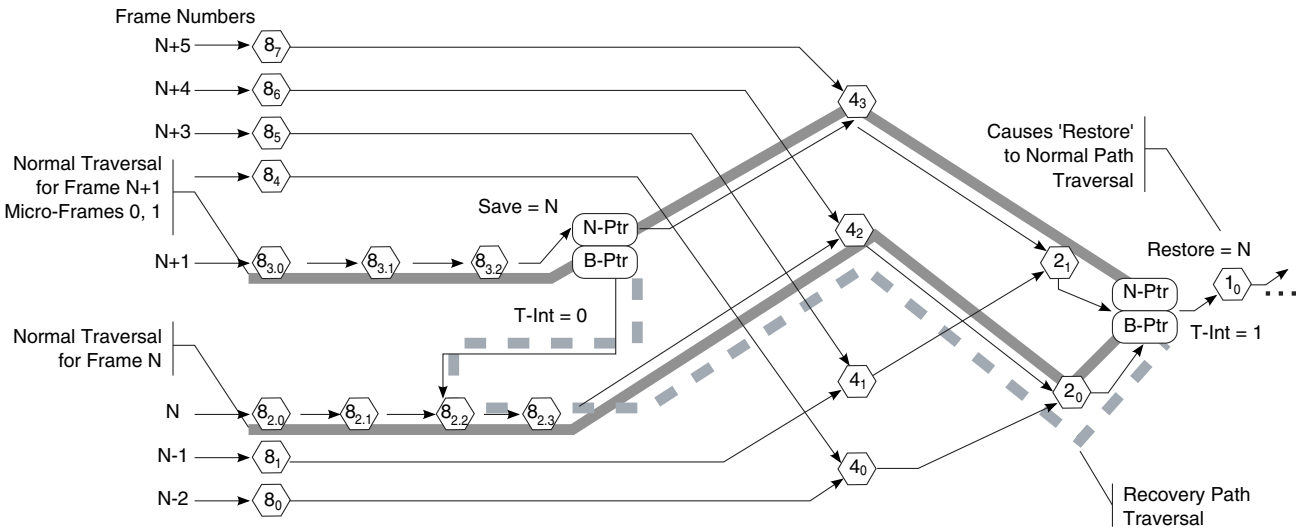


Figure 17-127. Example host controller traversal of recovery path via FSTNs

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 ...}.

17.8.12.2.3 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 will be use 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 will be 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.

17.8.12.2.4 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 micro-frame number in which the complete-split was executed. The host controller always checks C-prog-mask before executing a complete-split transaction. If the previous complete-splits have not been executed then it means one (or more) have been skipped and data has potentially been lost.
- FrameTag. This field is used by the host controller during the complete-split portion of the split transaction to tag the queue head with the frame number (H-Frame number) when the next complete split must be executed.
- S-bytes. This field can be used to store the number of data payload bytes sent during the start-split (if the transaction was an OUT). The S-bytes field must be used to accumulate the data payload bytes received during the complete-splits (for an IN).

17.8.12.2.5 Split Transaction Execution State Machine for Interrupt

In the following section, all references to micro-frame are in the context of a micro-frame within an H-Frame.

As with asynchronous Full- and Low-speed endpoints, a split-transaction state machine is used to manage the split transaction sequence. Aside from the fields defined in the queue head for scheduling and tracking the split transaction, the host controller calculates one internal mechanism that is also used to manage the split transaction. The internal calculated mechanism is:

- cMicroFrameBit. This is a single-bit encoding of the current micro-frame number. It is an eight-bit value calculated by the host controller at the beginning of every micro-frame. It is calculated from the three least significant bits of the FRINDEX register (that is, cMicroFrameBit = (1 shifted-left(FRINDEX[2-0]))). The cMicroFrameBit has at most one bit asserted, which always corresponds to the current micro-frame number. For example, if the current micro-frame is 0, then cMicroFrameBit will equal 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 micro-frame.

Figure 17-128 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 will be more than one complete-split transaction scheduled by software for the Do_Complete state. This translates simply to the fact that there are multiple bits set 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).

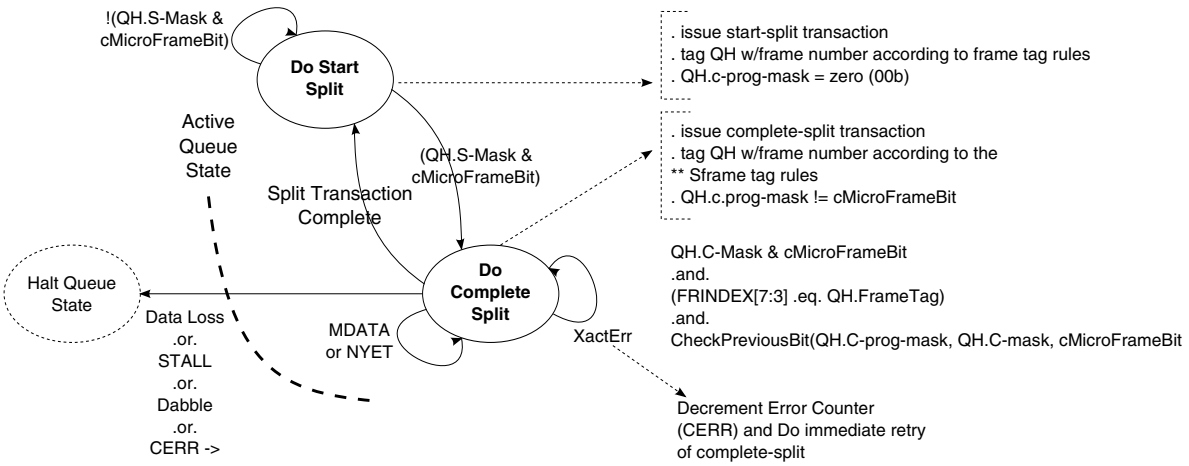


Figure 17-128. Split Transaction State Machine for Interrupt

17.8.12.2.6 Periodic Interrupt-Do-Start-Split

This is the state software must initialize a full- or low-speed interrupt queue head StartXState bit. This state is entered from the Do_Complete Split state only after the split transaction is complete. This occurs when one of the following events occur: The transaction translator responds to a complete-split transaction with one of the following:

- NAK. A NAK response is a propagation of the full- or low-speed endpoint's NAK response.
- ACK. An ACK response is a propagation of the full- or low-speed endpoint's ACK response. Only occurs on an OUT endpoint.
- DATA 0/1. Only occurs for INs. Indicates that this is the last of the data from the endpoint for this split transaction.
- ERR. The transaction on the low-/full-speed link below the transaction translator had a failure (for example, timeout, bad CRC, etc.).
- NYET (and Last). The host controller issued the last complete-split and the transaction translator responded with a NYET handshake. This means that the start-split was not correctly received by the transaction translator, so it never executed a transaction to the full- or low-speed endpoint, see Section Periodic 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.

17.8.12.2.7 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 17-152. 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 17-152. 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.

17.8.12.2.8 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 1).
- 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 1.
- 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 1.

17.8.12.2.9 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 micro-frame, 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.

17.8.12.3 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.

17.8.12.3.1 Split Transaction Scheduling Mechanisms for Isochronous

Full-speed isochronous transactions are managed through a transaction translator's periodic pipeline. As with full- and low-speed interrupt, system software manages each transaction translator's periodic pipeline by budgeting and scheduling exactly during which micro-frames the start-splits and complete-splits for each full-speed isochronous endpoint occur. The requirements described in Section Split Transaction Scheduling Mechanisms for Interrupt apply. Figure 17-129 illustrates the general scheduling boundary conditions that are supported by the EHCI periodic schedule. The S_n and C_n labels indicate micro-frames where software can schedule start- and complete-splits (respectively). The H-Frame boundaries are marked with a large, solid bold vertical line. The B-Frame boundaries are marked with a large, bold, dashed line. The figure below illustrates the relationship of an siTD to the H-Frame.

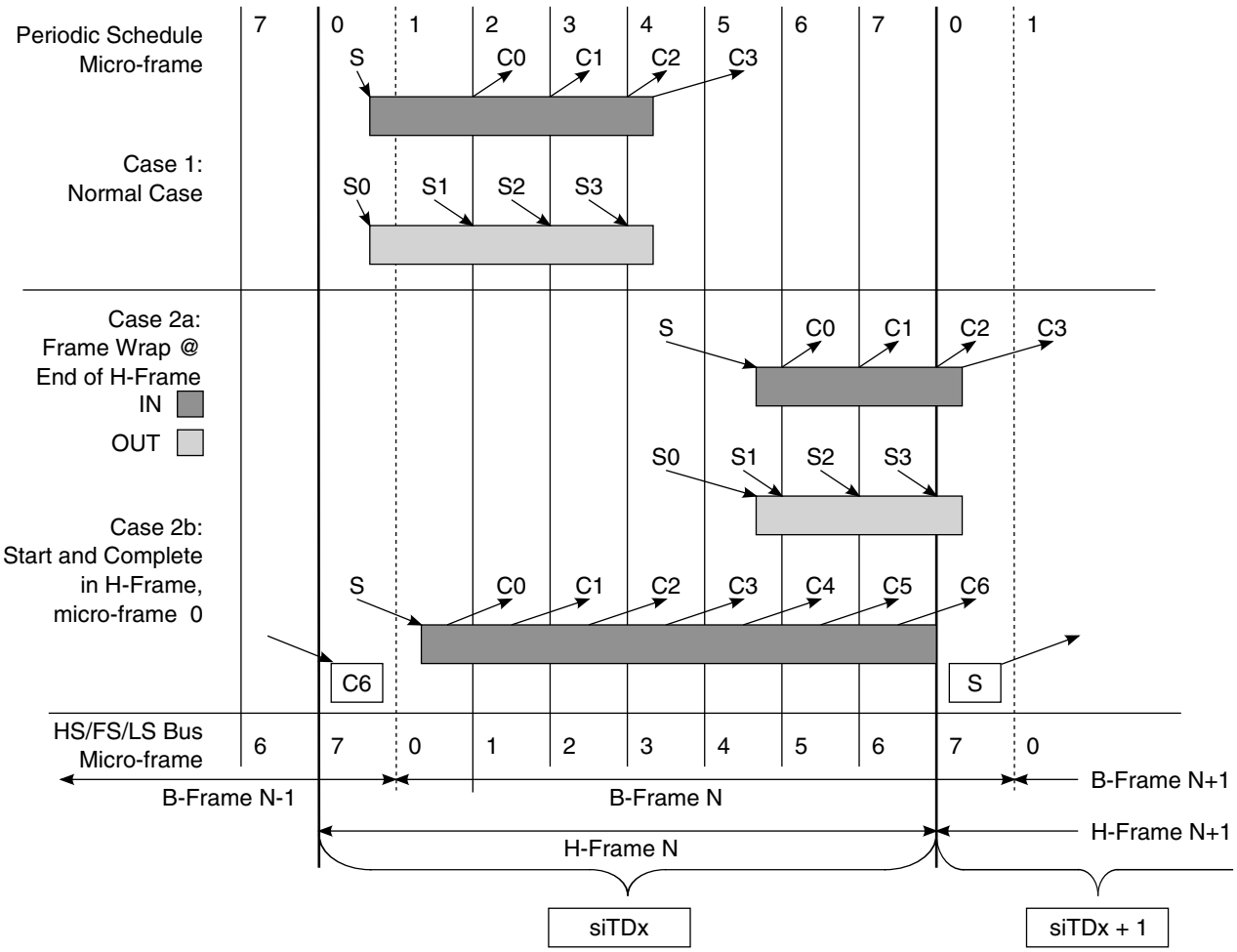


Figure 17-129. Split Transaction, Isochronous Scheduling Boundary Conditions

When the endpoint is an isochronous OUT, there are only start-splits, and no complete-splits. When the endpoint is an isochronous IN, there is at most one start-split and one to N complete-splits. The scheduling boundary cases are:

- Case 1: The entire split transaction is completely bounded by an H-Frame. For example, the start-splits and complete-splits are all scheduled to occur in the same H-Frame.
- Case 2a: This boundary case is where one or more (at most two) complete-splits of a split transaction IN are scheduled across an H-Frame boundary. This can only occur when the split transaction has the possibility of moving data in B-Frame, micro-frames 6 or 7 (H-Frame micro-frame 7 or 0). When an H-Frame boundary wrap condition occurs, the scheduling of the split transaction spans more than one location in the periodic list.(for example, it takes two siTDs in adjacent periodic frame list locations to fully describe the scheduling for the split transaction).

Although the scheduling of the split transaction may take two data structures, all of the complete-splits for each full-speed IN isochronous transaction must use only one data pointer. For this reason, siTDs contain a back pointer.

Software must never schedule full-speed isochronous OUTs across an H-Frame boundary.

- Case 2b: This case can only occur for a very large isochronous IN. It is the only allowed scenario where a start-split and complete-split for the same endpoint can occur in the same micro-frame. Software must enforce this rule by scheduling the large transaction first. Large is defined to be anything larger than 579 byte maximum packet size.

A subset of the same mechanisms employed by full- and low-speed interrupt queue heads are employed in siTDs to schedule and track the portions of isochronous split transactions. The following fields are initialized by system software to instruct the host controller when to execute portions of the split transaction protocol:

- SplitXState. This is a single bit residing in the Status field of an siTD (see Table 17-130). 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 micro-frame (within an H-Frame) that the host controller should execute a start-split transaction. This is always qualified by the value of the SplitXState bit. For example, referring to the IN example in Figure 17-129, 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 micro-frame as indicated by FRINDEX[2-0] is 0, then execute a start-split transaction.
- Frame C-mask. This is a bit-field where system software sets one or more bits corresponding to the micro-frames (within an H-Frame) that the host controller should execute complete-split transactions. The interpretation of this field is always qualified by the value of the SplitXState bit. For example, referring to the IN example in Figure 17-129, 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 micro-frame 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. Figure 17-130 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.

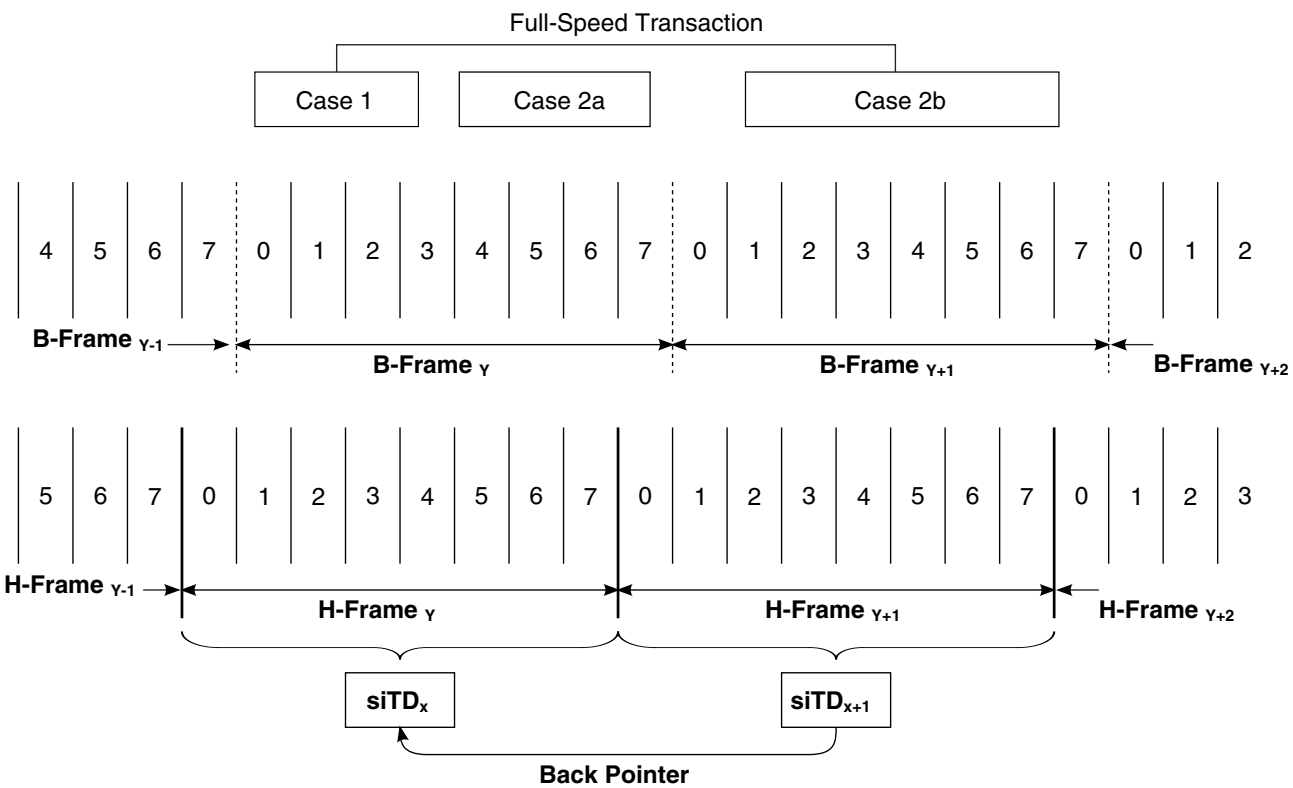


Figure 17-130. siTD Scheduling Boundary Examples

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 micro-frame 7 of H-Frame_{Y+1}, or micro-frame 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, micro-frame 1. This is mandated as a rule so that case 2a and case 2b can be discriminated. According to the core USB specification, the long isochronous transaction illustrated in Case 2b, could be scheduled so that the start-split was in micro-frame 1 of H-Frame N and the last complete-split would need to occur in micro-frame 1 of H-Frame N+1. However, it is impossible to discriminate between cases 2a and case 2b, which has significant impact on the complexity of the host controller.

17.8.12.3.2 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 micro-frames), timeouts and corrupted data received.

In similar kind to interrupt split-transactions, the portions of the split transaction protocol must execute in the micro-frames they are scheduled. The queue head data structure used to manage full- and low-speed interrupt has several mechanisms for tracking when portions of a transaction have occurred. Isochronous transfers use siTDs for their transfers and the data structures are only reachable using the schedule in the exact micro-frame in which they are required (so all the mechanism employed for tracking in queue heads is not required for siTDs). Software has the option of reusing siTD several times in

the complete periodic schedule. However, it must ensure that the results of split transaction N are consumed and the siTD re-initialized (activated) before the host controller gets back to the siTD (in a future micro-frame).

Split-transaction isochronous OUTs utilize a low-level protocol to indicate which portions of the split transaction data have arrived. Control over the low-level protocol is exposed in an siTD 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 micro-frame (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 micro-frame 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 micro-frame boundary, the transaction translator could have received the entire packet (including both CRC bytes) but not received the packet EOP. In the next micro-frame, the transaction translator 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.

17.8.12.3.3 Split Transaction Execution State Machine for Isochronous

In this section, all references to micro-frame are in the context of a micro-frame within an H-Frame.

If the Active bit in the Status byte is a zero, the host controller 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. Figure 17-131 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.

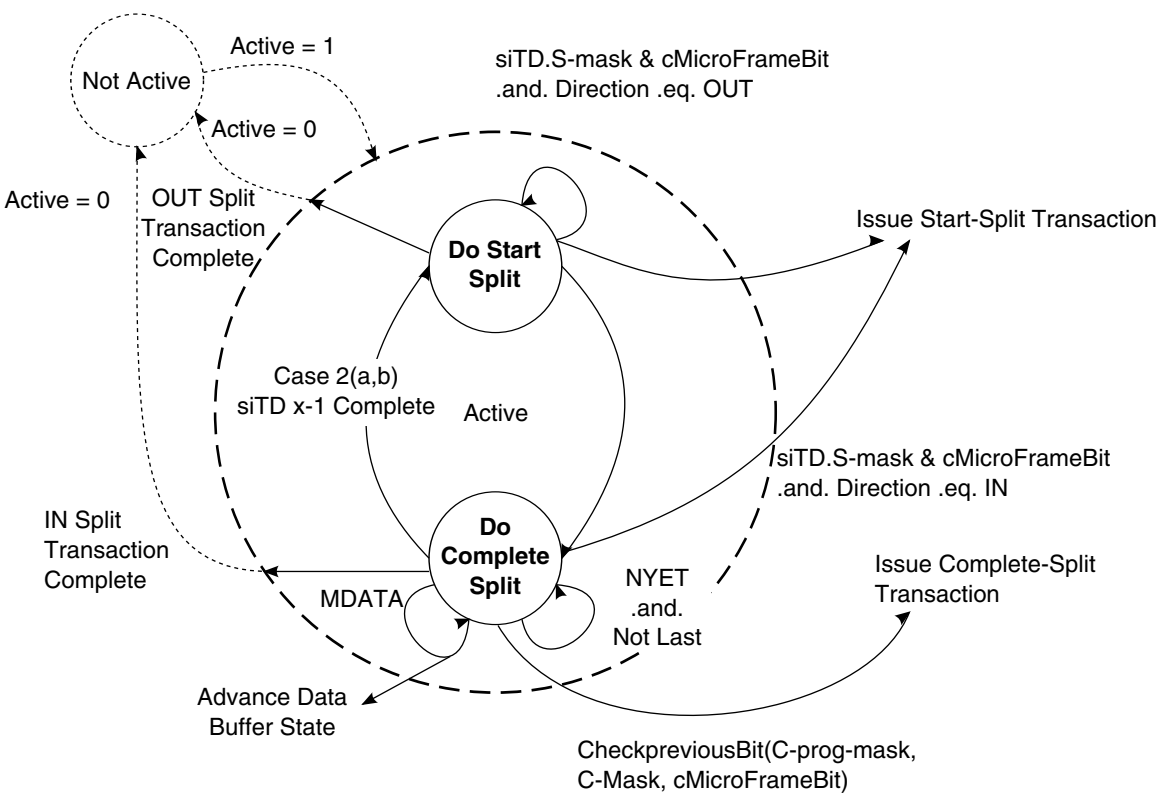


Figure 17-131. Split Transaction State Machine for Isochronous

17.8.12.3.4 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 2) is used to determine the initial value of TP. The initial cases are summarized in the table below.

Table 17-153. 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 17-154. 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 17-154.

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.

17.8.12.3.5 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 1). 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.

17.8.12.3.6 Complete-split for scheduling boundary cases 2a, 2b

Boundary cases 2a and 2b (INs only) (see Figure 1) 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 17-155. 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 17-155) 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.

17.8.12.3.7 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, table below illustrates a few frames worth of scheduling required to schedule a case 2a full-speed isochronous data stream.

Table 17-156. Example Case 2a-Software Scheduling siTDs for an IN Endpoint

siTDX		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	
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 micro-frame 4) and C-mask value of 0xC3 (one-bits in micro-frames 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 micro-frames 0 and 1 are ignored because the state is Do Start Split. During micro-frame 4, the host controller determines that it can run a start-split (and does) and changes SplitXState to Do Complete Split. During micro-frames 6 and 7, the host controller executes complete-splits. Notice the siTD for frame X+1 has 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 micro-frames 0 and 1 it will detect that it must execute complete-splits.

During H-Frame X+1, micro-frame 0, the host controller detects that $siTD_{X+1}$'s Back Pointer[T] bit is a zero, saves the state of $siTD_{X+1}$ and fetches $siTD_X$. It executes the complete split transaction using the transaction state of $siTD_X$. If the $siTD_X$ split transaction is complete, siTD's Active bit is 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 micro-frame 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, micro-frame 1.

During H-Frame X+2, micro-frame 0, the host controller detects that $siTD_{X+2}$'s Back Pointer[T] bit is 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, micro-frame 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 micro-frame 4.

17.8.13 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 is (assuming the CF-bit in the CONFIGFLAG register is set):

- Disable the periodic and asynchronous schedules by clearing the USBCMD[ASE] and USBCMD[PSE].
- Place all enabled root ports into the suspended state by setting the Suspend bit in the PORTSC register (PORTSC[SUSP]).

- Clear USBCMD[RS] (run/stop) and wait for USBSTS[HCH] to transition to a one. Note that an EHCI host controller implementation may optionally allow port testing with RS set. However, all host controllers must support port testing with RS cleared and HCH set.
- 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.
- 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].

17.8.14 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.

17.8.14.1 Transfer/Transaction Based Interrupts

These interrupt sources are associated with transfer and transaction progress. They are all dependent on the next interrupt threshold.

17.8.14.1.1 Transaction Error

A transaction error is any error that caused the host controller to think that the transfer did not complete successfully. Table 17-157 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 17-157. 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.

17.8.14.1.2 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 micro-frame EOF.

NOTE

When a host controller detects a data PID mismatch, it must either: disable the packet babble checking for the duration of the bus transaction or do packet babble checking based solely on Maximum Packet Size. The USB core specification defines the requirements on a data receiver when it receives a data PID mismatch (for example, expects a DATA0 and gets a DATA1 or visa-versa). In summary, it must ignore the received data and respond with an ACK handshake, in order to advance the transmitter's data sequence.The EHCI interface allows system software to provide buffers for a Control, Bulk or Interrupt IN endpoint that are not an even multiple of the maximum packet size specified by the device. Whenever a device misses an ACK for an IN endpoint, the host and device are out of

synchronization with respect to the progress of the data transfer. The host controller may have advanced the transfer to a buffer that is less than maximum packet size. The device 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.

17.8.14.1.3 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*.

17.8.14.1.4 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.

17.8.14.1.5 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.

17.8.14.2 Host controller event interrupts

These interrupt sources are independent of the interrupt threshold (with the one exception being the Interrupt on Async Advance.

17.8.14.2.1 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)

17.8.14.2.2 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.

17.8.14.2.3 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.

17.8.14.2.4 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 17-158. 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.

17.9 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.

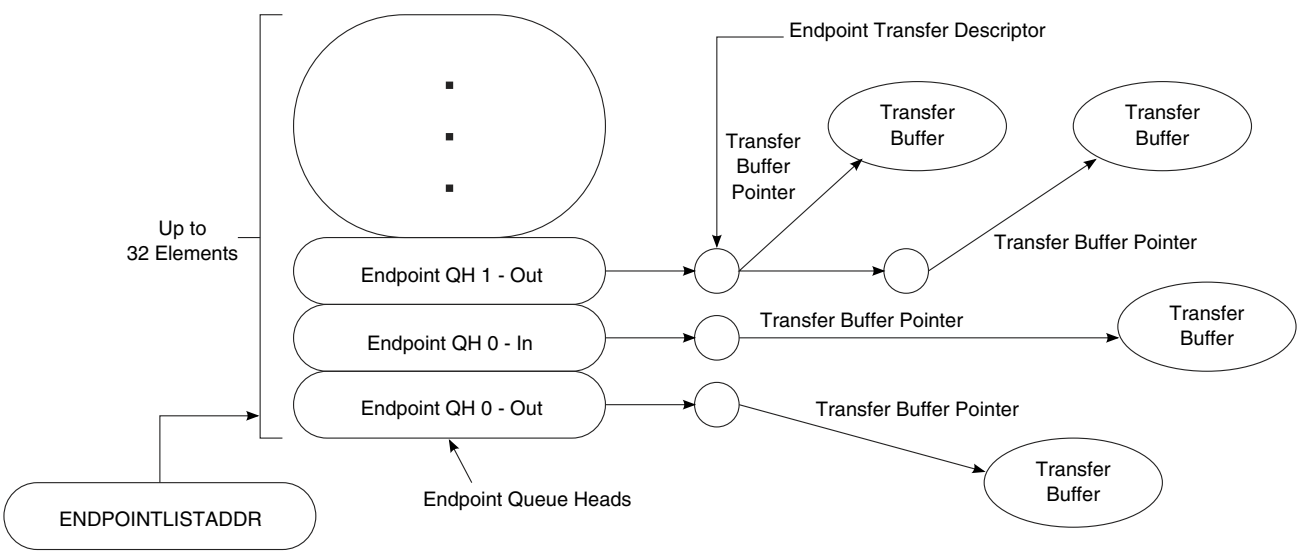


Figure 17-132. Endpoint queue head organization

17.9.1 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 17-159. 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.

17.9.1.1 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 17-160. 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:

Table continues on the next page...

Table 17-160. Endpoint capabilities/characteristics (continued)

Bits	Name	Description
		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. 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.

17.9.1.2 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 readied, the dTD will be copied into this queue head overlay area by the device controller. Until a transfer is expired, software must not write the queue head overlay area or the associated transfer descriptor. When the transfer is complete, the device controller will write the results back to the original transfer descriptor and advance the queue.

See dTD for a description of the overlay fields.

17.9.1.3 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 17-161. 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.

17.9.1.4 Set-up Buffer

The set-up buffer is dedicated storage for the 8-byte data that follows a set-up PID.

NOTE

Each endpoint has a TX and an RX dQH associated with it, and only the RX queue head is used for receiving setup data packets.

Table 17-162. 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.

17.9.2 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 17-163. 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

Table continues on the next page...

Table 17-163. Endpoint transfer descriptor (dTD) (continued)

Next Link Pointer							0000	T	0x00
0	Total Bytes[1]		ioc	000	MultO	00	Status[1]		0x04
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 17-164. 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 17-165. 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]

Table continues on the next page...

Table 17-165. dTD token (continued)

Bits	Description
	Three packets are sent: {Data2(8); Data1(7); Data0(0)} if QH.multiplier = 3; Maximum packet size = 8; Total bytes = 15; MultiO = 2 Two packets are sent: {Data1(8); Data0(7)} For maximal efficiency, software should compute MultO = greatest integer of (Total Bytes/Max. Packet Size) except for the case when Total bytes = 0; then MultO should be 1. **NOTE:** Non-ISO and non-TX endpoints must set MultO = 00.
9-8	Reserved, 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 17-166. 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 17-167. 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 17-168. 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

17.10 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.

17.10.1 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 internal UTMI PHY the following initialization sequence is required:

1. After power-on reset the UTMI PHY will be in disabled state and the PLL will be held reset.
2. Set the CONTROL[PHY_CLK_SEL] bits to select the UTMI PHY as the source of USB controller PHY clock.
3. Set the CONTROL[UTMI_PHY_EN] to enable the UTMI PHY and release the PLL.
4. Wait (approx 10 ms) for PHY clock to become valid.

Once the PHY clock is valid the user can proceed to the host controller initialization phase.

Once the PHY clock is valid the user can proceed to the device controller initialization phase.

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 UTMI 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.

17.10.2 Port State and Control

From a chip or system reset, the USB 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 following state diagram depicts the state of a USB 2.0 device.

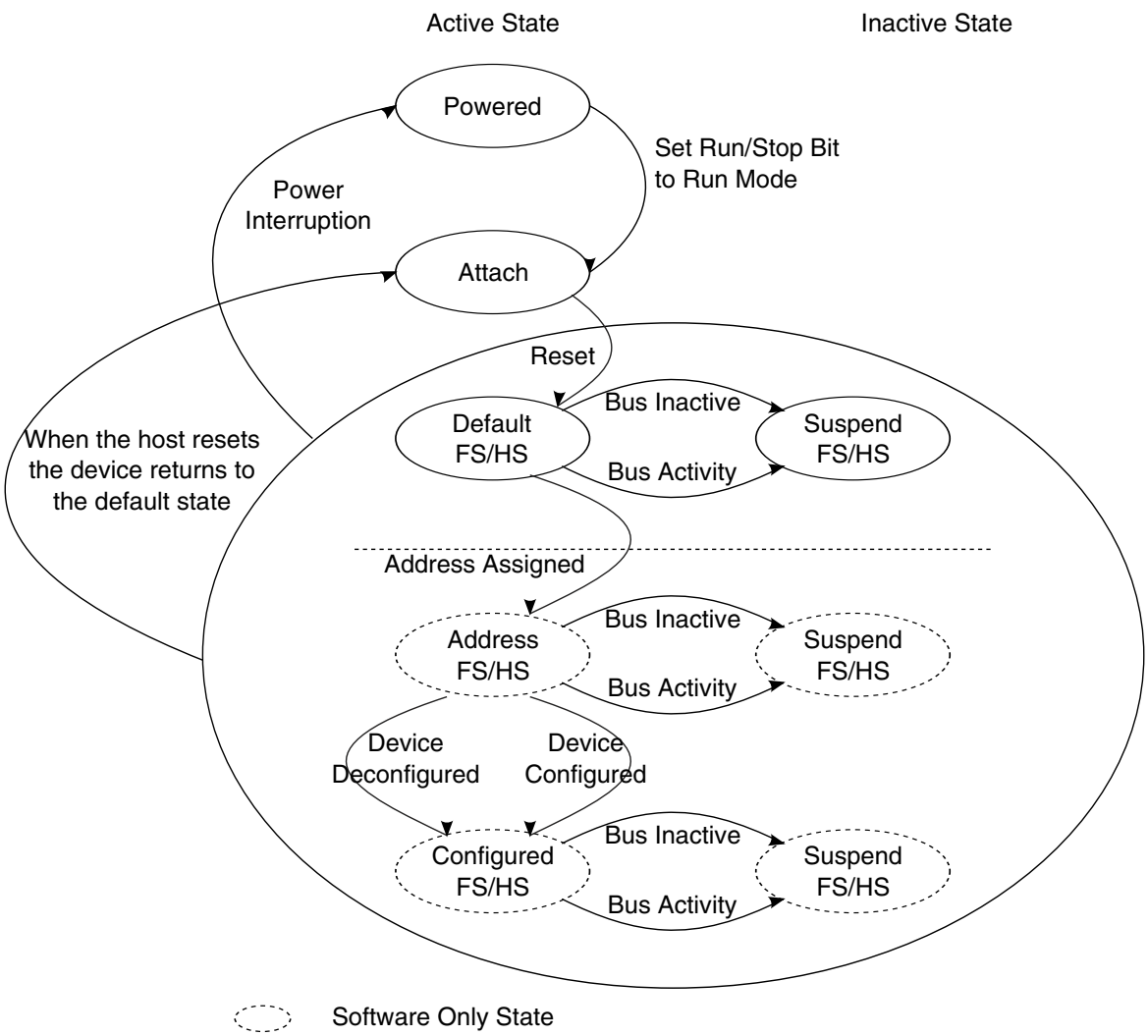


Figure 17-133. USB 2.0 Device States

States powered, attach, defaultFS/HS, suspendFS/HS are implemented in the USB controller and are communicated to the DCD using the following status bits:

Table 17-169. 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.

17.10.2.1 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.

17.10.2.2 Suspend/Resume-Operational Model

17.10.2.2.1 Suspend Description

In order to conserve power, USB controller automatically enters the suspended state when no bus traffic has been observed for a specified period. When suspended, the USB 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 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 controller is capable of remote wake-up signaling. When the USB controller is reset, remote wake-up signaling must be disabled.

17.10.2.2.2 Suspend Operational Model

The USB 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.

17.10.2.2.3 Resume

If the USB controller is suspended, its operation is resumed when any non-idle signaling is received on its upstream facing port. In addition, the USB 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 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.

17.10.3 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 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 is, 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.

17.10.3.1 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 17-170. 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

17.10.3.1.1 Stalling

There are two occasions where the USB 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).

Table 17-171. 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

17.10.3.2 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*.

17.10.3.2.1 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.

17.10.3.2.2 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.

17.10.3.3 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 can not be precisely predicted as a single pipeline, so it is not possible to prepare a single packet for the device controller to execute. However, the order of packet requests is predictable when the endpoint number and direction is considered. For example, if endpoint 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.

17.10.3.3.1 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 will be stored in the dQH until the device controller completes the transfer described by the dTD. Storing the dTD in the dQH allows the device controller to fetch the operating context needed to handle a request from the host without the need to follow the linked list, starting at the dQH when the host request is received.

After the device has loaded the dTD, the leading data in the packet is stored in a FIFO in the device controller. This FIFO is split into virtual channels so that the leading data can be stored for any endpoint up to the maximum number of endpoints configured at device synthesis time.

After a priming request is complete, an endpoint state of primed is indicated in the 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 must be sized to account for the maximum latency that can be incurred by the system memory bus.

17.10.3.3.2 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.

17.10.3.4 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 17-172. 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 17-173. 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.

17.10.3.4.1 Interrupt/bulk endpoint bus response matrix

The table below shows the interrupt/bulk endpoint bus response matrix.

Table 17-174. 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.

17.10.3.5 Control endpoint operation model

This section discusses the control endpoint operation model.

17.10.3.5.1 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 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 be an incoming setup packet. This is the preferred behavior because ignoring repeated setup packets due to long software interrupt latency would be a compliance issue.

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.

17.10.3.5.2 Data Phase

Following the setup phase, the DCD must create a device transfer descriptor for the data phase and prime the transfer.

After priming the packet, the DCD must verify a new setup packet has not been received by reading the 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.

17.10.3.5.3 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.

17.10.3.5.4 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 17-175. 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.

17.10.3.6 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 controller will is accomplished by the following:

- Exactly MULT Packets per (micro)Frame are transmitted/received.

NOTE

MULT is a 2-bit field in the device Queue Head. The variable length packet protocol is not used on isochronous endpoints.

- NAK responses are not used. Instead, zero length packets and sent in response to an IN request to an unprimed endpoints. For unprimed RX endpoints, the response to an OUT transaction is to ignore the packet within the device controller.
- Prime requests always schedule the transfer described in the dTD for the next (micro)frame. If the ISO-dTD is still active after that frame, then the ISO-dTD will be held ready until executed or canceled by the DCD.

The USB 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 will be cleared as usual to indicate to software that the device controller completed a priming the dTD for transfer. Internal to the design, the device controller hardware masks that prime start until the next frame boundary. This behavior is hidden from the DCD but occurs so that the device controller can match the dTD to a specific (micro) frame.

Another difference with isochronous endpoints is that the transaction must wholly complete in a (micro)frame. Once an ISO transaction is started in a (micro)frame it will retire the corresponding dTD when MULT transactions occur or the device controller finds a fulfillment condition.

The transaction error bit set in the status field indicates a fulfillment error condition. When a fulfillment error occurs, the frame after the transfer failed to complete wholly, the device controller will force retire the ISO-dTD and move to the next ISO-dTD.

It is important to note that fulfillment errors are only caused due to partially completed packets. If no activity occurs to a primed ISO-dTD, the transaction will stay primed indefinitely. This means it is up to software discard transmit ISO-dTDs that pile up from a failure of the host to move the data.

Finally, the last difference with ISO packets is in the data level error handling. When a CRC error occurs on a received packet, the packet is not retried similar to bulk and control endpoints. Instead, the CRC is noted by setting the Transaction Error bit and the data is stored as usual for the application software to sort out.

- TX Packet Retired
  - MULT counter reaches zero.
  - Fulfillment Error [Transaction Error bit is set]
  - #Packets Occurred > 0 AND # Packets Occurred < MULT

NOTE

For TX-ISO, MULT Counter can be loaded with a lesser value in the dTD Multiplier Override field. 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.

17.10.3.6.1 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.

17.10.3.6.2 Isochronous endpoint bus response matrix

The table below shows the isochronous endpoint bus response matrix.

Table 17-176. 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.

17.10.4 Managing queue heads

The figure below shows the endpoint queue head diagram.

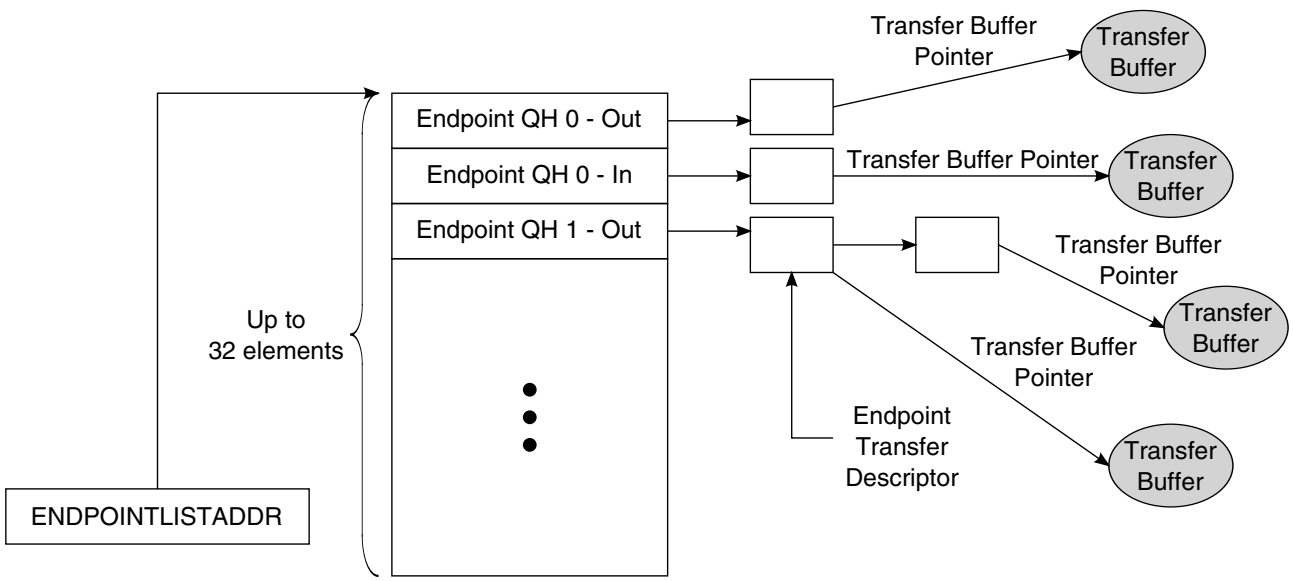


Figure 17-134. Endpoint queue head diagram

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 17-134. 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.

17.10.4.1 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: 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.

17.10.4.2 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.

17.10.5 Managing transfers with transfer descriptors

This section describes software link pointers, transfer descriptors, transfer completion and the device error matrix.

17.10.5.1 Software Link Pointers

It is necessary for the DCD software to maintain head and tail pointers to the for the linked list of dTDs for each respective queue head. This is necessary because the dQH only maintains pointers to the current working dTD and the next dTD to be executed. The operations described in next section for managing dTD will assume the DCD can use reference the head and tail of the dTD linked list.

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.

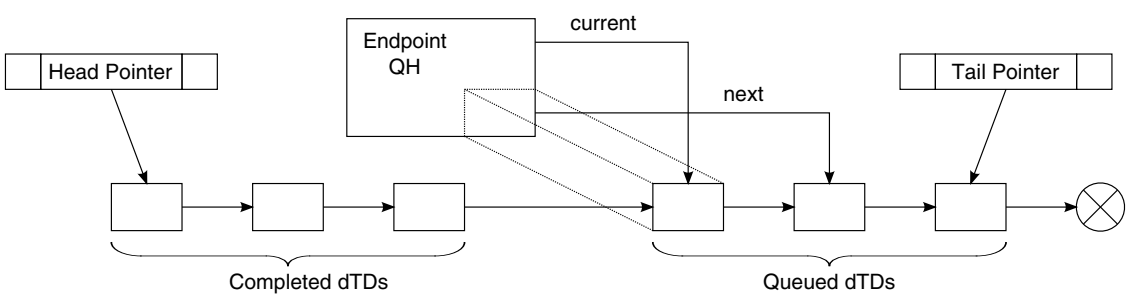


Figure 17-135. Software Link Pointers

17.10.5.2 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.

17.10.5.3 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.

17.10.5.4 Transfer Completion

After a dTD has been initialized and the associated endpoint primed the device controller will execute the transfer upon the host-initiated request. The DCD will be notified with a USB interrupt if the Interrupt On Complete bit was set or alternately, the DCD can poll the endpoint complete register to find when the dTD had been executed. After a dTD has been executed, DCD can check the status bits to determine success or failure.

NOTE

Multiple dTD can be completed in a single endpoint complete notification. After clearing the notification, DCD must search the dTD linked list and retire all dTDs that have finished (Active bit cleared).

By reading the status fields of the completed dTDs, the DCD can determine if the transfers completed successfully. Success is determined with the following combination of status bits:

- Active = 0
- Halted = 0
- Transaction Error = 0
- Data Buffer Error = 0

Should any combination other than the one shown above exist, the DCD must take proper action. Transfer failure mechanisms are indicated in the Device Error Matrix.

In addition to checking the status bit the DCD must read the Transfer Bytes field to determine the actual bytes transferred. When a transfer is complete, the Total Bytes transferred is by decremented by the actual bytes transferred. For Transmit packets, a packet is only complete after the actual bytes reaches zero, but for receive packets, the host may send fewer bytes in the transfer according the USB variable length packet protocol.

17.10.5.5 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.

17.10.5.6 Device error matrix

The table below summarizes packet errors that are not automatically handled by the USB controller.

Table 17-177. 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 17-178. 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.

17.10.6 Servicing Interrupts

The interrupt service routine must consider that there are high-frequency, low-frequency operations, and error operations and order accordingly.

17.10.6.1 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 17-179. 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.

17.10.6.2 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 17-180. 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.

17.10.6.3 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 17-181. 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.

17.11 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.

17.11.1 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.

17.11.1.1 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 HCSPARAMS, for usage information.

17.11.1.2 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.

17.11.1.3 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 17-182. 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)]

17.11.1.4 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.

17.11.1.5 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.

17.11.1.5.1 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.

It is important to note that when programming the S-mask and C-masks in the EHCI data structures to schedule periodic transfers for the embedded transaction translator, the EHCI host controller driver must follow the same rules specified in EHCI for programming the S-mask and C-mask for downstream Hub-based transaction translators.

Once periodic transfers are exhausted, any stored asynchronous transfer will be moved. Asynchronous transfers are opportunistic in that they shall execute whenever possible and their operation is not tied to H-frame and B-frame boundaries with the exception that an asynchronous transfer can not babble through the SOF (start of B-frame 0).

17.11.1.5.2 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. Table below summarizes the conditions where handshakes are emulated from internal state instead of actual handshakes to HS split bus traffic.

Table 17-183. 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]

17.11.1.5.3 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

17.11.1.5.4 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.

17.11.1.5.5 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 HCSPARAMS, for more information.

17.11.2 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.

17.11.3 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).

17.11.4 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. Note that the free running interrupt is shared with the device-mode start-of-frame interrupt. See USB status (USB_USBSTS), and USB interrupt enable (USB_USBINTR), for more information.

17.11.5 Embedded design

This is an Embedded USB Host Controller as defined by the EHCI specification and thus does not implement the PCI configuration registers.

17.11.5.1 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.

17.11.6 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.

17.11.6.1 Discovery

This section discusses port reset and port speed detection.

17.11.6.1.1 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.

17.11.6.1.2 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 18
DUART

18.1 DUART 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 DUART registers and features.

The DUART consists of two universal asynchronous receiver/transmitters (UARTs). Note that this device implements two DUART modules. DUART1 contains UART1 and UART2; DUART2 contains UART3 and UART4. The UARTs act independently; all references to UART refer to one of these receiver/transmitters. The DUART programming model is compatible with the PC16552D.

The UART interface is point to point, meaning that only two UART devices are attached to the connecting signals. As shown in Figure 18-1, each UART module consists of the following:

- Receive and transmit buffers
- Clear to send (CTS_B) input port and request to send (RTS_B) output port for data flow control
- 16-bit counter for baud rate generation
- Interrupt control logic

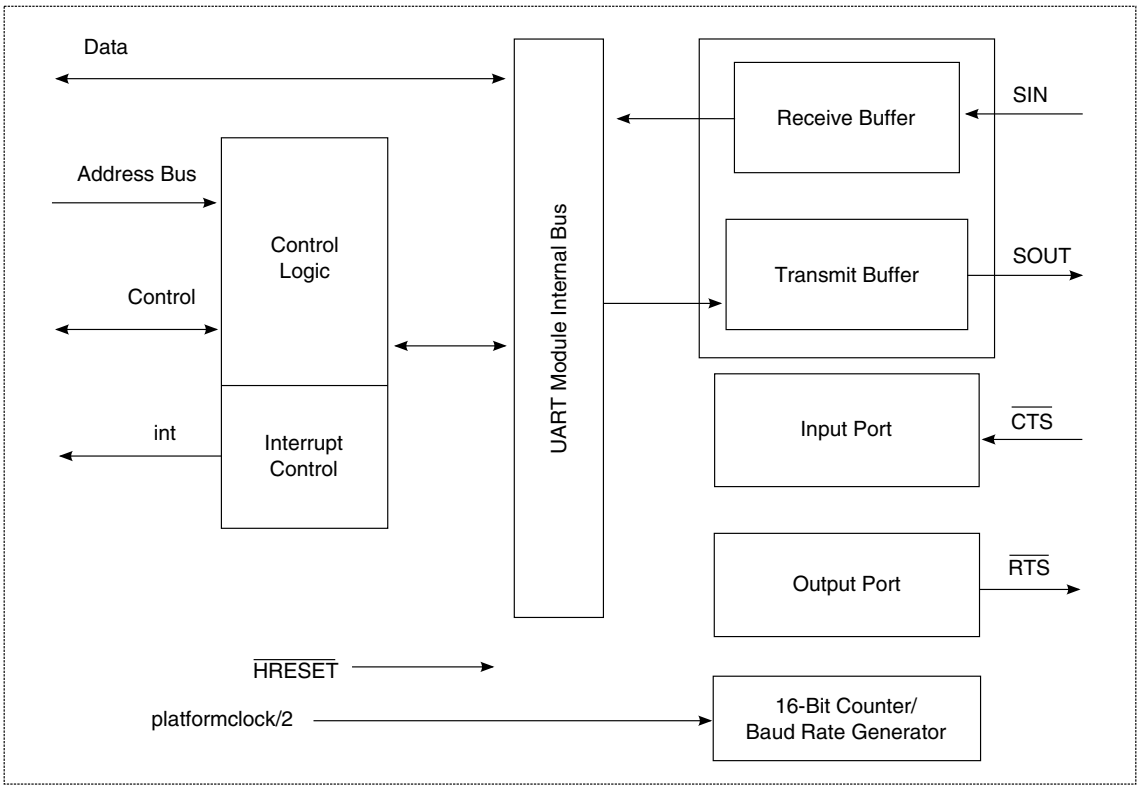


Figure 18-1. UART Block Diagram

18.2 DUART Features Summary

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
- FIFO mode for both transmitter and receiver, providing 16-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 CCB clock (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 request 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

18.3 DUART Modes of Operation

The communication channel provides a full-duplex asynchronous receiver and transmitter using an operating frequency derived from the platform clock/2.

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.

18.3.1 DUART Signal Mode Selection

The UART signals are multiplexed with other functions on the device. The following relationships apply:

- UART1 shares signals with GPIO and UART3
- UART2 shares signals with GPIO and UART4

The functionality of these signals is determined by the UART field in the reset configuration word (RCW[UART_BASE]). Note that RCW[UART_BASE] = 0x0 defaults to all GPIO functionality on the UART signal pins; the RCW must be initialized to the desired UART signaling for proper UART operation.

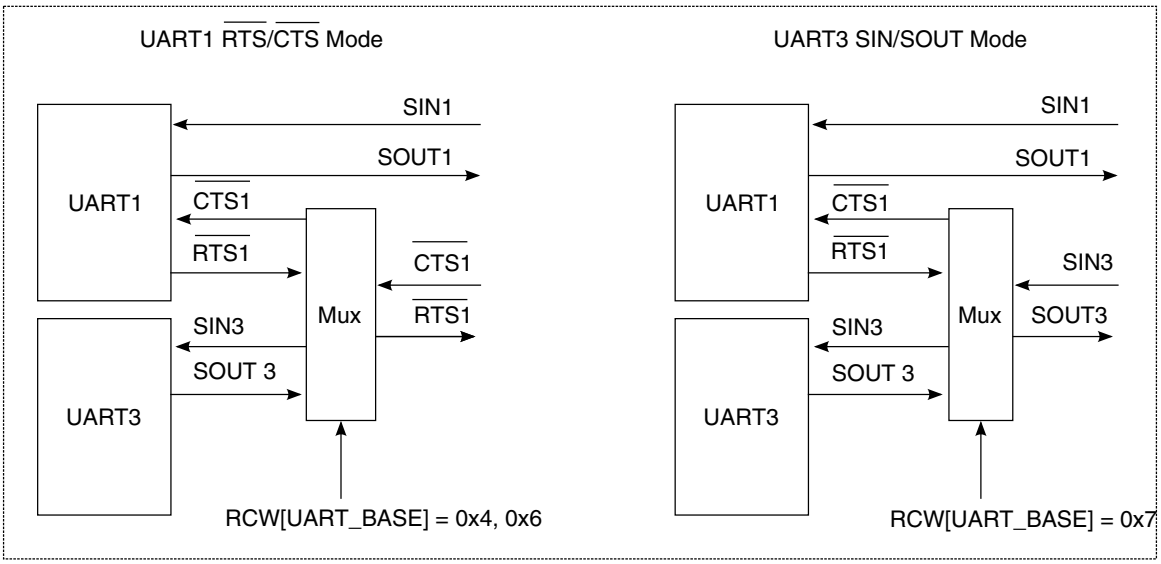


Figure 18-2. UART1/UART3 Signal Multiplexing

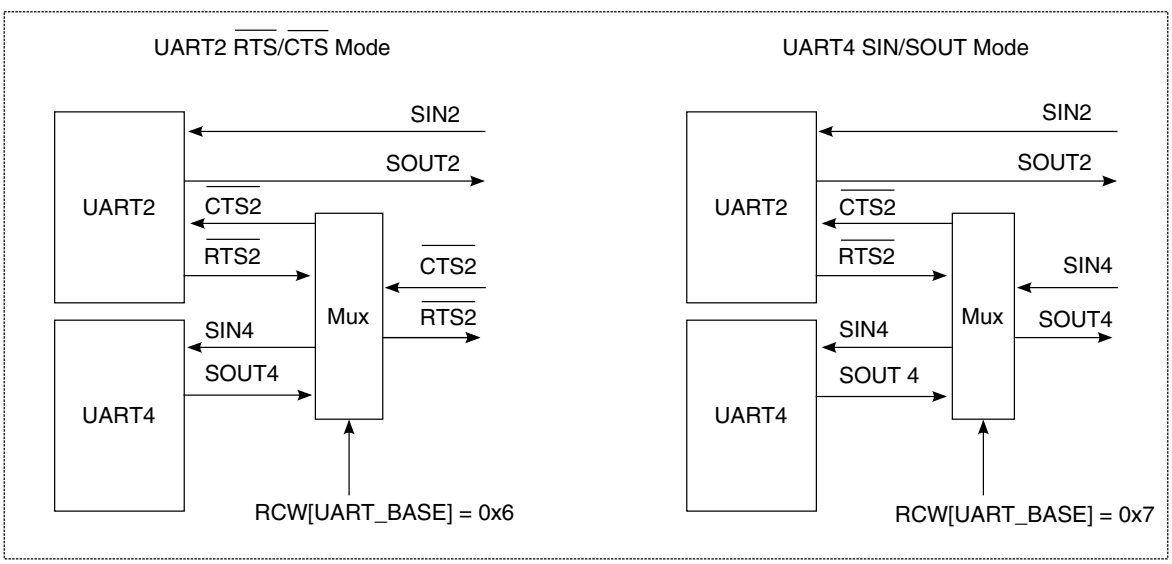


Figure 18-3. UART2/UART4 Signal Multiplexing

18.4 DUART External Signal Descriptions

The DUART signals are described in Table 18-1. Note that although the actual device signal names are prepended with the UART_ prefix as shown in the table, the functional (abbreviated) signal names are often used throughout this chapter.

Table 18-1. DUART Signals-Detailed Signal Descriptions

Signal	I/O	Description	
UART*n*_SIN	I	Serial data in. Data is received on the receivers of UART1 , UART2, UART3, and UART4 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 UART1 , UART2, UART3, and UART4 are set ('mark' condition) when the transmitter is disabled, idle, or operating in the local loopback mode. 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.
UART*n*_CTS_B	I	Clear to send. These active-low inputs are the clear-to-send inputs. They are connected to the respective RTS_B outputs of the other UART devices on the bus. They can be programmed to generate an interrupt on change-of-state of the signal. Note that UART1_CTS is not available when UART3 is configured for SIN/SOUT mode and UART2_CTS is not available when UART4 is configured for SIN/SOUT mode. **NOTE:** The DUARTs are set during reset to be in either 2-pin or 4-pin mode (or disabled if the GPIO functionality is selected). For 2-pin mode, the CTS_B inputs are asserted internally.	
		State Meaning	Asserted/Negated-Represent the clear to send condition for their respective UART.
		Timing	Assertion/Negation-Sampled at the rising edge of every input clock (platform clock/2).
UART*n*_RTS_B	O	Request to send. These active-low output signals can be programmed to be automatically negated and asserted by either the receiver or transmitter. When connected to the clear-to-send (CTS_B) input of a transmitter, this signal can be used to control serial data flow.	
		State Meaning	Asserted/Negated-Represents the data being transmitted on the respective UART interface.
		Timing	Assertion/Negation-Updated and driven at the rising edge of every input clock (platform clock/2).

18.5 DUART Memory Map/Register Definition

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. Note that the full register address is comprised of CCSRBAR together with the block base address and offset.

There are four complete sets of registers (one for each UART). The four 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 either UART1, UART2, UART3, or UART4.

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 (DUART_ULCR*n*) 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.

NOTE

UART2 has the same memory-mapped registers that are described for UART1 from 0x500 to 0x510 except the offsets range from 0x600 to 0x610. Similarly, the registers for UART3 are located at offsets0x500--0X510 from the DUART2 block base address and the registers for UART4 are located at offsets 0x600--0x610 from the DUART2 block base address.

DUART memory map

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
11_C500	UART receiver buffer register (DUART1_URBR1)	8	R	00h	18.5.1/1132
11_C500	UART transmitter holding register (DUART1_UTHR1)	8	W	00h	18.5.2/1133
11_C500	UART divisor least significant byte register (DUART1_UDLB256)	8	R/W	00h	18.5.3/1133
11_C501	UART divisor most significant byte register (DUART1_UDMB1)	8	R/W	00h	18.5.4/1135
11_C501	UART interrupt enable register (DUART1_UIER1)	8	R/W	00h	18.5.5/1135
11_C502	UART interrupt ID register (DUART1_UIIR1)	8	R	01h	18.5.6/1136
11_C502	UART FIFO control register (DUART1_UFCR1)	8	W	00h	18.5.7/1137
11_C502	UART alternate function register (DUART1_UAFR1)	8	R/W	00h	18.5.8/1139
11_C503	UART line control register (DUART1_ULCR1)	8	R/W	00h	18.5.9/1139

Table continues on the next page...

DUART memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
11_C504	UART modem control register (DUART1_UMCR1)	8	R/W	00h	18.5.10/ 1141
11_C505	UART line status register (DUART1_ULSR1)	8	R	60h	18.5.11/ 1142
11_C506	UART modem status register (DUART1_UMSR1)	8	R	00h	18.5.12/ 1143
11_C507	UART scratch register (DUART1_USCR1)	8	R/W	00h	18.5.13/ 1144
11_C510	UART DMA status register (DUART1_UDSR1)	8	R	01h	18.5.14/ 1144
11_C600	UART receiver buffer register (DUART1_URBR2)	8	R	00h	18.5.1/1132
11_C600	UART transmitter holding register (DUART1_UTHR2)	8	W	00h	18.5.2/1133
11_C600	UART divisor least significant byte register (DUART1_UDLB)	8	R/W	00h	18.5.3/1133
11_C601	UART divisor most significant byte register (DUART1_UDMB2)	8	R/W	00h	18.5.4/1135
11_C601	UART interrupt enable register (DUART1_UIER2)	8	R/W	00h	18.5.5/1135
11_C602	UART interrupt ID register (DUART1_UIIR2)	8	R	01h	18.5.6/1136
11_C602	UART FIFO control register (DUART1_UFCR2)	8	W	00h	18.5.7/1137
11_C602	UART alternate function register (DUART1_UAFR2)	8	R/W	00h	18.5.8/1139
11_C603	UART line control register (DUART1_ULCR2)	8	R/W	00h	18.5.9/1139
11_C604	UART modem control register (DUART1_UMCR2)	8	R/W	00h	18.5.10/ 1141
11_C605	UART line status register (DUART1_ULSR2)	8	R	60h	18.5.11/ 1142
11_C606	UART modem status register (DUART1_UMSR2)	8	R	00h	18.5.12/ 1143
11_C607	UART scratch register (DUART1_USCR2)	8	R/W	00h	18.5.13/ 1144
11_C610	UART DMA status register (DUART1_UDSR2)	8	R	01h	18.5.14/ 1144
11_D500	UART receiver buffer register (DUART2_URBR1)	8	R	00h	18.5.1/1132
11_D500	UART transmitter holding register (DUART2_UTHR1)	8	W	00h	18.5.2/1133
11_D500	UART divisor least significant byte register (DUART2_UDLB256)	8	R/W	00h	18.5.3/1133
11_D501	UART divisor most significant byte register (DUART2_UDMB1)	8	R/W	00h	18.5.4/1135
11_D501	UART interrupt enable register (DUART2_UIER1)	8	R/W	00h	18.5.5/1135
11_D502	UART interrupt ID register (DUART2_UIIR1)	8	R	01h	18.5.6/1136
11_D502	UART FIFO control register (DUART2_UFCR1)	8	W	00h	18.5.7/1137
11_D502	UART alternate function register (DUART2_UAFR1)	8	R/W	00h	18.5.8/1139
11_D503	UART line control register (DUART2_ULCR1)	8	R/W	00h	18.5.9/1139

Table continues on the next page...

DUART memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
11_D504	UART modem control register (DUART2_UMCR1)	8	R/W	00h	18.5.10/ 1141
11_D505	UART line status register (DUART2_ULSR1)	8	R	60h	18.5.11/ 1142
11_D506	UART modem status register (DUART2_UMSR1)	8	R	00h	18.5.12/ 1143
11_D507	UART scratch register (DUART2_USCR1)	8	R/W	00h	18.5.13/ 1144
11_D510	UART DMA status register (DUART2_UDSR1)	8	R	01h	18.5.14/ 1144
11_D600	UART receiver buffer register (DUART2_URBR2)	8	R	00h	18.5.1/1132
11_D600	UART transmitter holding register (DUART2_UTHR2)	8	W	00h	18.5.2/1133
11_D600	UART divisor least significant byte register (DUART2_UDLB)	8	R/W	00h	18.5.3/1133
11_D601	UART divisor most significant byte register (DUART2_UDMB2)	8	R/W	00h	18.5.4/1135
11_D601	UART interrupt enable register (DUART2_UIER2)	8	R/W	00h	18.5.5/1135
11_D602	UART interrupt ID register (DUART2_UIIR2)	8	R	01h	18.5.6/1136
11_D602	UART FIFO control register (DUART2_UFCR2)	8	W	00h	18.5.7/1137
11_D602	UART alternate function register (DUART2_UAFR2)	8	R/W	00h	18.5.8/1139
11_D603	UART line control register (DUART2_ULCR2)	8	R/W	00h	18.5.9/1139
11_D604	UART modem control register (DUART2_UMCR2)	8	R/W	00h	18.5.10/ 1141
11_D605	UART line status register (DUART2_ULSR2)	8	R	60h	18.5.11/ 1142
11_D606	UART modem status register (DUART2_UMSR2)	8	R	00h	18.5.12/ 1143
11_D607	UART scratch register (DUART2_USCR2)	8	R/W	00h	18.5.13/ 1144
11_D610	UART DMA status register (DUART2_UDSR2)	8	R	01h	18.5.14/ 1144

18.5.1 UART receiver buffer register (DUARTx_URBR*n*)

(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 (DUART_ULSR*n*). Note that these registers have same offset as the UTHRs.

Address: Base address + 500h offset + (256d × i), where i=0d to 1d

Bit	0	1	2	3	4	5	6	7
Read	DATA							
Write								
Reset	0	0	0	0	0	0	0	0

DUARTx_URBR*n* field descriptions

Field	Description
0–7 DATA	Data received from the transmitter on the UART bus (read only)

18.5.2 UART transmitter holding register (DUARTx_UTHR*n*)

(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] indicates when the FIFO is full. Refer to UART DMA status register (DUART_UDSR*n*) for more details.

Address: Base address + 500h offset + (256d × i), where i=0d to 1d

Bit	0	1	2	3	4	5	6	7
Read								
Write	DATA							
Reset	0	0	0	0	0	0	0	0

DUARTx_UTHR*n* field descriptions

Field	Description
0–7 DATA	Data that is written to UTHR (write only)

18.5.3 UART divisor least significant byte register (DUARTx_UDLB*n*)

(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 = input clock frequency/(16 x [UDMB||UDLB]). Equivalently, [UDMB||UDLB:0b0000] = input clock frequency/desired baud rate. Baud rates that can be generated by specific input clock frequencies are shown in the table below.

The table below shows examples of baud rate generation based on common input clock frequencies. Many other target baud rates are also possible. Note that 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 18-72. Baud Rate Examples

Target Baud Rate (Decimal)	Divisor		DUART Input Clock (Platform Clock/2) Frequency (MHz)	Actual Baud Rate (Decimal)	Percent Error (Decimal)
	Decimal	Hex			
9,600	1953	07A1	300	9600.61444	0.0064
19,200	977	03D1	300	19,191.40225	0.0448
38,400	488	01E8	300	38,422.13115	0.0576
57,600	326	0146	300	57,515.33742	0.1470
115,200	163	00A3	300	115,030.67485	0.1470
230,400	81	0051	300	231,481.48148	0.4694
9,600	2168	0878	333	9599.86162	0.0014
19,200	1084	043C	333	19,199.72325	0.0014
38,400	542	021E	333	38,399.44649	0.0014
57,600	361	0169	333	57,652.35457	0.0909
115,200	181	00B5	333	114,986.18785	0.1856
230,400	90	005A	333	231,250.00000	0.3689

Address: Base address + 500h offset + (256d × i), where i=0d to 1d

Bit	0	1	2	3	4	5	6	7
Read Write	UDLB							
Reset	0	0	0	0	0	0	0	0

DUARTx_UDLB*n* field descriptions

Field	Description
0–7 UDLB	Divisor least significant byte. This is concatenated with UDMB.

18.5.4 UART divisor most significant byte register (DUARTx_UDMB*n*)

(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 = input clock frequency/(16 x [UDMB||UDLB]). Equivalently, [UDMB||UDLB:0b0000] = input clock frequency/desired baud rate. Baud rates that can be generated by specific input clock frequencies are shown in Table 18-7.

Address: Base address + 501h offset + (256d × i), where i=0d to 1d

Bit	0	1	2	3	4	5	6	7
Read Write	UDMB							
Reset	0	0	0	0	0	0	0	0

DUARTx_UDMB*n* field descriptions

Field	Description
0–7 UDMB	Divisor most significant byte

18.5.5 UART interrupt enable register (DUARTx_UIER*n*)

(ULCR[DLAB] = 0)

The UIER gives the user the ability to mask specific UART interrupts to the programmable interrupt controller (PIC).

Address: Base address + 501h offset + (256d × i), where i=0d to 1d

Bit	0	1	2	3	4	5	6	7
Read Write	Reserved				EMSI	ERLSI	ETHREI	ERDAI
Reset	0	0	0	0	0	0	0	0

DUARTx_UIER*n* field descriptions

Field	Description
0–3 -	This field is reserved. Reserved
4 EMSI	Enable modem status interrupt. 0 Mask interrupts caused by UMSR[DCTS] being set 1 Enable and assert interrupts when the clear-to-send bit in the UART modem status register (UMSR) changes state

Table continues on the next page...

DUARTx_UIER*n* field descriptions (continued)

Field	Description
5 ERLSI	Enable receiver line status interrupt. 0 Mask interrupts when ULSR's overrun, parity error, framing error or break interrupt bits are set 1 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. 0 Mask interrupt when ULSR[THRE] is set 1 Enable and assert interrupts when ULSR[THRE] is set
7 ERDAI	Enable received data available interrupt. 0 Mask interrupt when new receive data is available or receive data time out has occurred 1 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

18.5.6 UART interrupt ID register (DUARTx_UIIR*n*)

(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

See the table below for more details.

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 18-80. 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.

Table continues on the next page...

Table 18-80. UIIR IID Bits Summary (continued)

IID Bits IID[3-0]	Priority Level	Interrupt Type	Interrupt Description	How To Reset Interrupt
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.

Address: Base address + 502h offset + (256d × i), where i=0d to 1d

Bit	0	1	2	3	4	5	6	7
Read	FE		Reserved		IID3	IID2	IID1	IID0
Write								
Reset	0	0	0	0	0	0	0	1

DUARTx_UIIR*n* field descriptions

Field	Description
0–1 FE	FIFOs enabled. Reflects the setting of UFCR[FEN]
2–3 -	This field is reserved. 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. 0 The UART has an active interrupt ready to be serviced. 1 No interrupt is pending.

18.5.7 UART FIFO control register (DUARTx_UFCR*n*)

(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.

Address: Base address + 502h offset + (256d × i), where i=0d to 1d

Bit	0	1	2	3	4	5	6	7
Read			Reserved					
Write	RTL				DMS	TFR	RFR	FEN
Reset	0	0	0	0	0	0	0	0

DUARTx_UFCR*n* field descriptions

Field	Description
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: 00 1 byte 01 4 bytes 10 8 bytes 11 14 bytes
2–3 -	This field is reserved. Reserved
4 DMS	DMA mode select. See DMA Mode Select for more information. 0 UDSR[RXRDY] and UDSR[TXRDY] bits are in mode 0. 1 UDSR[RXRDY] and UDSR[TXRDY] bits are in mode 1 if UFCR[FEN] = 1.
5 TFR	Transmitter FIFO reset 0 No action 1 Clears all bytes in the transmitter FIFO and resets the FIFO counter/pointer to 0
6 RFR	Receiver FIFO reset 0 No action 1 Clears all bytes in the receiver FIFO and resets the FIFO counter/pointer to 0
7 FEN	FIFO enable 0 FIFOs are disabled and cleared 1 Enables the transmitter and receiver FIFOs

18.5.8 UART alternate function register (DUARTx_UAFRn)

(ULCR[DLAB] = 1)

The UAFRs give software the ability to gate off the baud clock and write to both UART1/UART2 registers or both UART3/UART4 registers simultaneously with the same write operation.

Address: Base address + 502h offset + (256d × i), where i=0d to 1d

Bit	0	1	2	3	4	5	6	7
Read Write	Reserved						BO	CW
Reset	0	0	0	0	0	0	0	0

DUARTx_UAFRn field descriptions

Field	Description
0–5 -	This field is reserved. Reserved
6 BO	Baud clock select. 0 The baud clock is not gated off. 1 The baud clock is gated off.
7 CW	Concurrent write enable. 0 Disables concurrent writing to both UART1 and UART2 (DUART1) or UART3 and UART4 (DUART2) 1 Enables concurrent writes to corresponding UART registers. For DUART1, a write to a register in UART1 is also a write to the corresponding register in UART2 and vice versa; for DUART2, a write to a register in UART3 is also a write to the corresponding register in UART4 and vice versa. The user needs to ensure that the LCR[DLAB] of both UARTs are in the same state before executing a concurrent write to register addresses 0x*n*00, 0x*n*01 and 0x*n*02, where *n* is the offset of the corresponding UART.

18.5.9 UART line control register (DUARTx_ULCRn)

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 18-88. 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

Address: Base address + 503h offset + (256d × i), where i=0d to 1d

Bit	0	1	2	3	4	5	6	7
Read Write	DLAB	SB	SP	EPS	PEN	NSTB	WLS	
Reset	0	0	0	0	0	0	0	0

DUARTx_ULCR*n* field descriptions

Field	Description
0 DLAB	Divisor latch access bit. 0 Access to all registers except UDLB, UAFR, and UDMB 1 Ability to access divisor latch least and most significant byte registers and alternate function register (UAFR)
1 SB	Set break. 0 Send normal UTHR data onto the serial output (SOUT) signal 1 Force logic 0 to be on the SOUT signal. Data in the UTHR is not affected
2 SP	Stick parity. 0 Stick parity is disabled. 1 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. 0 If PEN = 1 and SP = 0, odd parity is selected. 1 If PEN = 1 and SP = 0, even parity is selected.
4 PEN	Parity enable.

Table continues on the next page...

DUARTx_ULCR*n* field descriptions (continued)

Field	Description
	0 No parity generation and checking 1 Generate parity bit as a transmitter, and check parity as a receiver
5 NSTB	Number of STOP bits. 0 One STOP bit is generated in the transmitted data. 1 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: 00 5 bits 01 6 bits 10 7 bits 11 8 bits

18.5.10 UART modem control register (DUARTx_UMCR*n*)

The UMCRs control the interface with the external peripheral device on the UART bus.

.

Address: Base address + 504h offset + (256d × i), where i=0d to 1d

Bit	0	1	2	3	4	5	6	7
Read Write	Reserved			LOOP	Reserved		RTS	Reserved
Reset	0	0	0	0	0	0	0	0

DUARTx_UMCR*n* field descriptions

Field	Description
0–2 -	This field is reserved. Reserved
3 LOOP	Local loopback mode. 0 Normal operation 1 Functionally, the data written to UTHR can be read from URBR of the same UART, and UMCR[RTS] is tied to UMSR[CTS].
4–5 -	This field is reserved. Reserved
6 RTS	Ready to send. 0 Negates corresponding UART_RTS_B output 1 Assert corresponding UART_RTS_B output. Informs external modem or peripheral that the UART is ready for sending/receiving data
7 -	This field is reserved. Reserved

18.5.11 UART line status register (DUARTx_ULSR*n*)

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.

Address: Base address + 505h offset + (256d × i), where i=0d to 1d

Bit	0	1	2	3	4	5	6	7
Read	RFE	TEMT	THRE	BI	FE	PE	OE	DR
Write								
Reset	0	1	1	0	0	0	0	0

DUARTx_ULSR*n* field descriptions

Field	Description
0 RFE	Receiver FIFO error. 0 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. 1 Set to one when one of the characters in the receiver FIFO encounters an error (framing, parity, or break interrupt)
1 TEMT	Transmitter empty. 0 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. 1 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. 0 The UTHR is not empty. 1 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. 0 This bit is cleared when the ULSR is read or when a valid data transfer is detected (that is, STOP bit is received). 1 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.

Table continues on the next page...

DUARTx_ULSR*n* field descriptions (continued)

Field	Description
4 FE	Framing error. 0 This bit is cleared when ULSR is read or when a new character is loaded into the URBR from the receiver shift register. 1 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 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. 0 This bit is cleared when ULSR is read or when a new character is loaded into the URBR. 1 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. 0 This bit is cleared when ULSR is read. 1 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. 0 This bit is cleared when URBR is read or when all of the data in the receiver FIFO is read. 1 A character has been received in the URBR or the receiver FIFO.

18.5.12 UART modem status register (DUART*x*_UMSR*n*)

The UMSRs track the status of the modem (or external peripheral device) clear to send (CTS_B) signal for the corresponding UART.

Address: Base address + 506h offset + (256d × i), where i=0d to 1d

Bit	0	1	2	3	4	5	6	7
Read	Reserved			CTS	Reserved			DCTS
Write								
Reset	0	0	0	0	0	0	0	0

DUARTx_UMSR*n* field descriptions

Field	Description
0–2 -	This field is reserved. Reserved

Table continues on the next page...

DUARTx_UMSR*n* field descriptions (continued)

Field	Description
3 CTS	Clear to send. Represents the inverted value of the CTS_B input pin from the external peripheral device 0 Corresponding CTS_B is negated 1 Corresponding CTS_B is asserted. The modem or peripheral device is ready for data transfers.
4–6 -	This field is reserved. Reserved
7 DCTS	Clear to send. 0 No change on the corresponding CTS_B signal since the last read of UMSR[CTS] 1 The CTS_B value has changed, since the last read of UMSR[CTS]. Causes an interrupt if UIER[EMSI] is set to detect this condition

18.5.13 UART scratch register (DUARTx_USCR*n*)

The USCR registers are for debugging software or the DUART hardware. The USCRs do not affect the operation of the DUART.

Address: Base address + 507h offset + (256d × i), where i=0d to 1d

Bit	0	1	2	3	4	5	6	7
Read Write	DATA							
Reset	0	0	0	0	0	0	0	0

DUARTx_USCR*n* field descriptions

Field	Description
0–7 DATA	Data

18.5.14 UART DMA status register (DUARTx_UDSR*n*)

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 18-103. 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	

Table continues on the next page...

Table 18-103. UDSR[TXRDY] Set Conditions (continued)

DMS	FEN	DMA Mode	Meaning
1	1	1	TXRDY is set when the transmitter FIFO is full.

Table 18-104. 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.

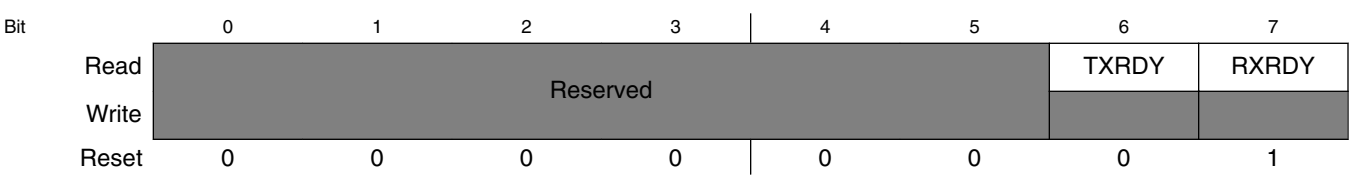
Table 18-105. UDSR[RXRDY] Set Conditions

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 18-106. 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.

Address: Base address + 510h offset + (256d × i), where i=0d to 1d



DUARTx_UDSR*n* field descriptions

Field	Description
0–5 -	This field is reserved. Reserved

Table continues on the next page...

DUARTx_UDSR*n* field descriptions (continued)

Field	Description
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. 0 The bit is cleared, as shown above in Table 18-36. 1 This bit is set, as shown above in Table 18-35.
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. 0 The bit is cleared, as shown above in Table 18-38. 1 This bit is set, as shown above in Table 18-37.

18.6 DUART Functional Description

The communication channel provides a full-duplex asynchronous receiver and transmitter using an operating frequency derived from one-half of the platform clock.

The transmitter accepts parallel data with a write access to the transmitter holding register (UTHR). In FIFO mode, the data is placed directly into an internal transmitter shift register, or into the transmitter FIFO-see FIFO Mode. The transmitting registers convert the data to a serial bit stream, by inserting the appropriate START, STOP, and optional parity bits. Finally, the registers output a composite serial data stream on the channel transmitter serial data output (SOUT). The transmitter status may be polled or interrupt-driven.

The receiver accepts serial data on the channel receiver serial data input (SIN), converts the data into parallel format, and checks for START, STOP, and parity bits. In FIFO mode, the receiver removes the START, STOP, and parity bits and then transfers the assembled character from the receiver buffer, or receiver FIFO. This transfer occurs in response to a read of the UART receiver buffer register (URBR). The receiver status may be polled or interrupt driven.

18.6.1 Serial Interface

The UART bus is a serial, full-duplex, point-to-point bus as shown in Figure 18-130. Therefore, only two devices are attached to the same signals and there is no need for address or arbitration bus cycles.

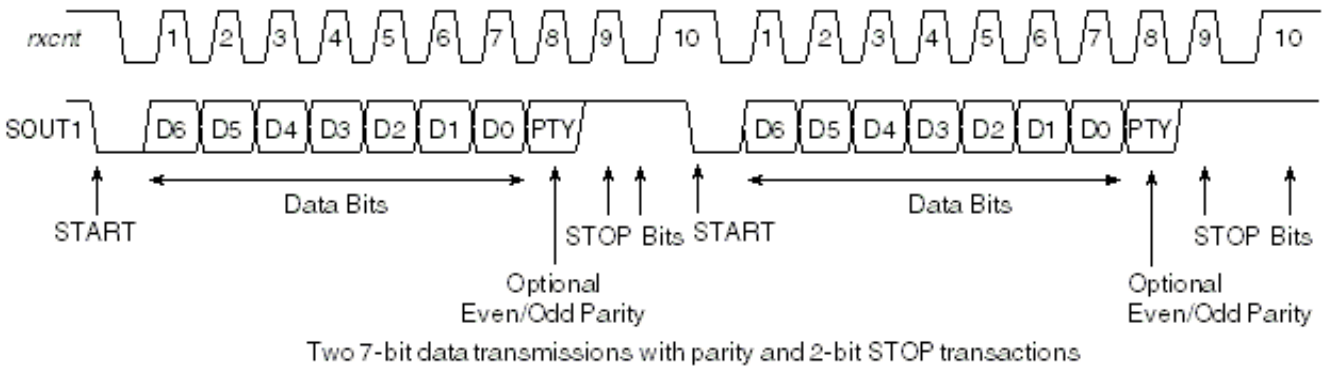


Figure 18-130. UART Bus Interface Transaction Protocol Example

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.

18.6.1.1 START Bit

A write to the transmitter holding register (UTHR) generates a START bit on the SOUT signal. Figure 18-130 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.

18.6.1.2 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.

18.6.1.3 Parity Bit

The user has the option of using even, odd, no parity, or stick parity (see UART line control register (DUART_ULCR*n*)). 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 (DUART_ULSR*n*).)

18.6.1.4 STOP Bit

The transmitter device ends the write transfer by generating a STOP bit. The STOP bit is always high. The user can program the length of the STOP bit(s) 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.

18.6.2 Baud-Rate Generator Logic

Each UART contains an independent programmable baud-rate generator, that is capable of taking the DUART module input clock (platform clock/2) and dividing it 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 (DUART module input clock 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.

18.6.3 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 (input signal) is disconnected, RTS is internally connected to CTS, and the RTS (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 that in local loopback mode, the transmit/receive interrupts are fully operational and can be controlled by the interrupt enable register (UIER).

18.6.4 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.

18.6.4.1 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.

18.6.4.2 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.

18.6.4.3 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.

18.6.5 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.

18.6.5.1 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 (DUART_UIER*n*) for more details on interrupt enables.

The interrupt ID register (UIIR) indicates if the FIFOs are enabled. Interrupt ID3 UIIR[IID3] bit 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 (DUART_UIIR*n*) for more information.

The UIIR[FE] bits indicate if FIFO mode is enabled.

18.6.5.2 DMA Mode Select

The UDSR[RXRDY] bit reflects the status of the receiver FIFO or URBR. In mode 0 (UFCR[DMS] is cleared), UDSR[RXRDY] is cleared when there is at least one character in the receiver FIFO or URBR and it is set when there are no more characters in the receiver FIFO or URBR. This occurs regardless of the setting of the UFCR[FEN] bit. In mode 1 (UFCR[DMS] and UFCR[FEN] are set), UDSR[RXRDY] is cleared when the trigger level or a time-out has been reached and it is set when there are no more characters in the receiver FIFO.

The UDSR[TXRDY] bit reflects the status of the transmitter FIFO or UTHR. In mode 0 (UFCR[DMS] is cleared), UDSR[TXRDY] is cleared when there are no characters in the transmitter FIFO or UTHR and it is set after the first character is loaded into the transmitter FIFO or UTHR. This occurs regardless of the setting of the UFCR[FEN] bit. In mode 1 (UFCR[DMS] and UFCR[FEN] are set), UDSR[TXRDY] is cleared when there are no characters in the transmitter FIFO or UTHR and it is set when the transmitter FIFO is full.

See UART DMA status register (DUART_UDSR*n*) for a complete description of the USDR[RXRDY] and USDR[TXRDY] bits.

18.6.5.3 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. For more details refer to the description of UIER in UART interrupt enable register (DUART_UIER*n*).

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 (UMSR) registers. UIIR[IID0] can be used for polling if the interrupts are enabled in UIER.

18.7 DUART Initialization/Application Information

The following requirements must be met for DUART accesses:

- All DUART registers must be mapped to a cache-inhibited and guarded area. (That is, the WIMG setting in the MMU needs to be 0b01*n*1.)
- 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. Update the DUART configuration register (DCR).
3. Set data attributes and control bits in the ULCR, UFCR, UAFR, UMCR, UDLB, and UDMB.
4. Set the data attributes and control bits of the external modem or peripheral device.
5. Set the interrupt enable register (UIER).
6. To start a write transfer, write to the UTHR.
7. Poll UIIR if the interrupts generated by the DUART are masked.

Chapter 19
SerDes Module

19.1 Overview

The SerDes module implements link serialization/deserialization and PCS functions for high speed serial interfaces from 1.25 Gbaud to 10.3125 Gbaud.

19.1.1 Features

The SerDes module includes eight data lanes and two PLLs, and supports the following protocols:

- PCI Express 3.0 (November, 2010) @ 2.5, 5 Gbaud and x1, x2, x4 @ 8 Gbaud
- Serial RapidIO^{TM} 2.1 (August, 2009) @ 2.5, 3.125, 5 Gbaud
- Aurora SP002, Revision 1.3 (June, 2004) @ 2.5, 3.125, 5 Gbaud
- SGMII ENG-46158, Revision 1.7 (July, 2001) @ 1.25, 3.125 Gbaud
- IEEE 802.3ae - 2002 XAUI (August, 2002) @ 3.125 Gbaud
- HiGig @3.125 Gbaud
- HiGig2 @ 3.75 Gbaud
- XFI/10GBASE-R and 10GBASE-KR @ 10.3125 Gbaud
  - IEEE 802.3-2007
  - IEEE 802.3ae-2007
  - IEEE 802.3ap-2007
  - IEEE 802.3aq-2007
  - IEEE 802.3an-2006
- Serial ATA Revision 2.6 (February, 2007) @ 1.5, 3 Gbaud

19.2 Modes of Operation

The SerDes supports several combinations of protocols and frequencies. See Section 19.4, "SerDes Lane Assignments and Multiplexing," for details on the combinations. The supported protocols are:

- x1, x2, x4 or x8 PCI Express @ 2.5, 5, 8 Gbaud
  - Note: 8 Gbaud is not supported with x8 link width, and is restricted in other configurations. See Section 19.4, "SerDes Lane Assignments and Multiplexing," for details on the supported configurations for PCIe gen3.
- x2 or x4 Serial RapidIO @ 2.5, 3.125, 5 Gbaud: SerDes 2
- Aurora @ 2.5, 3.125, 5 Gbaud: SerDes 2

  - x2 Rx and x2 Tx
  - x2 Rx and x4 Tx, with the Rx lanes 1:0 same as Tx lanes 1: 0
- SGMII @ 1.25, 3.125 Gbaud: SerDes 1
- XAUI @ 3.125 Gbaud: SerDes 1
- HiGig/HiGig2 @ 3.125, 3.75 Gbaud: SerDes 1
- XFI @ 10.3125 Gbaud: SerDes 1
- SATA @ 1.5, 3 Gbaud: SerDes 2

19.3 External Signals Description

The table below lists the external signals for the SerDes.

Table 19-1. SerDes Interface Signals

Pin Name	Description	No. of Signals	I/O
SDn_TX[0:7]_P	Transmitter serial output, positive data	8	O
SDn_TX[0:7]_N	Transmitter serial output, negative dat	8	O
SDn_RX[0:7]_P	Receiver serial output, positive data	8	I
SDn_RX[0:7]_N	Receiver serial output, negative da	8	I
x=1, 2			
SDn_REFx_CLK_P	Reference clock input to PLL***x***	2	I
SDn_REFx_CLK_N	Reference clock-bar input to PLL***x***	2	I

19.4 SerDes Lane Assignments and Multiplexing

19.4.1 SerDes Protocols (SerDes 1 and SerDes 2)

The following table shows the supported protocol options for SerDes modules 1 and 2. The following notation conventions are used in the table:

- XAUIm indicates XAUI (4 lanes @ 3.125 Gbps), “m” is the MAC on the Frame Manager. For example, XAUI9 indicates that MAC9 runs XAUI (4 lanes @ 3.125 Gbps).
- XFIm indicates XFI (1 lane @10.3125 Gbps), “m” indicates MAC on the Frame Manager. For example, “XFI9” indicates XFI using MAC 9.
- HiGig notation :
  - HiGigm indicates HiGig (4 lanes @3.125 Gbps), “m” indicates MAC on the Frame Manager. For example, “Higig9” indicates HiGig using MAC 9.
  - *HiGigm* : Higig (4 lanes @ 3.75 Gbps), “m” indicates MAC on the Frame Manager. For example, “*Higig9*” indicates HiGig using MAC 9.
- SGMII notation :

  - sgm means SGMII @ 1.25 Gbps where “m” indicates which MAC on the Frame Manager. For example, “SG3” indicates SGMII for MAC 3 at 1.25 Gbps.
  - *sgm* means SGMII @3.125Gbps where “m” indicates which MAC on the Frame Manager. For example, “*sg3*” indicates SGMII for MAC 3 at 3.25 Gbps.
- PCIe notation :
  - PCIem is PCIe @ 5/2.5 Gbps, m indicates the PCIe controller number.
  - *PCIem* is PCIe @ 8/5/2.5 Gbps, m indicates the PCIe controller number.
- SRIO notation :
  - SRIOm, m is SRIO1 or SRIO2 @ 5/2.5 Gbps
  - *SRIOm*, m is SRIO1 or SRIO2 @ 3.125 Gbps
- Aurora notation : Aurora is Aurora @ 5/2.5 Gbps
- SATA notation : SATAm, where m is SATA1 or SATA2
- Per lane PLL mapping: 1-PLL1, 2-PLL2

NOTE

RGMII may use MAC3 and/or MAC4/10. When RGMII is enabled for a particular MAC (using EC1/EC2 RCW fields), it always takes precedence over SerDes-based interfaces.

If SRIO2 is chosen in SRDS_PRTCL_S2 then DMA3 is not functional. If SRIO2 is not chosen DMA3 is functional

Table 19-2. SerDes Lanes Assignments and Multiplexing

SERDES1									
SRDS_PRTCL_S1	A	B	C	D	E	F	G	H	Per[1] Lane PLL Mapping
1C	SG9	SG10	SG1	SG2	SG3	SG4	SG5	SG6	1111 1111
95	SG9	SG10	*SG1*	*SG2*	SG3	SG4	SG5	SG6	1122 1111
A2	SG9	SG10	SG1	SG2	SG3	SG4	SG5	SG6	1111 1111
94	SG9	SG10	*SG1*	*SG2*	SG3	SG4	SG5	SG6	1122 1111
51	XAUI9				PCIe4	SG4	SG5	SG6	1111 2222
5F	Higig9				PCIe4	SG4	SG5	SG6	1111 2222
65	*Higig9*				PCIe4	SG4	SG5	SG6	1111 2222
6B	XFI9	XFI10	XFI1	XFI2	PCIe4	SG4	SG5	SG6	1111 2222

SERDES2									
SRDS_PRTCL_S2	A	B	C	D	E	F	G	H	Per[1] Lane PLL Mapping
1F	*PCIe1*				PCIe2				1111 2222
1F	*PCIe1*				PCIe2				1111 2222
16	PCIe1				PCIe2		SATA1	SATA2	1111 1122
16	PCIe1				PCIe2		SATA1	SATA2	1111 1122
1F	*PCIe1*				PCIe2				1111 2222
1F	*PCIe1*				PCIe2				1111 2222
1F	*PCIe1*				PCIe2				1111 2222
1F	*PCIe1*				PCIe2				1111 2222

Table 19-2. SerDes Lanes Assignments and Multiplexing

SERDES1										SERDES2									
SRDS_PRTCL_S1	A	B	C	D	E	F	G	H	Per[1] Lane PLL Mapping	SRDS_PRTCL_S2	A	B	C	D	E	F	G	H	Per[1] Lane PLL Mapping
6C	XFI9	XFI10	SG1	SG2	PCIe4				1122 2222	16	PCIe1				PCIe2		SATA1	SATA2	1111 1122
6D	XFI9	XFI10	SG1	SG2	PCIe4				1122 2222	1F	*PCIe1*				PCIe2				1111 2222
6D	XFI9	XFI10	SG1	SG2	PCIe4				1122 2222	1	PCIe1								1111 1111
6C	XFI9	XFI10	SG1	SG2	PCIe4				1122 2222	29	*SRIO2*				*SRIO1*				1111 1111
6D	XFI9	XFI10	SG1	SG2	PCIe4				1122 2222	2D	SRIO2				SRIO1				1111 1111
71	XFI9	XFI10	SG1	SG2	PCIe4	SG4	SG5	SG6	1122 2222	1F	*PCIe1*				PCIe2				1111 2222
71	XFI9	XFI10	SG1	SG2	PCIe4	SG4	SG5	SG6	1122 2222	15	PCIe1				PCIe2		SATA1	SATA2	1111 1122
6C	XFI9	XFI10	SG1	SG2	PCIe4				1122 2222	18	PCIe1				Aurora		SATA1	SATA2	1111 1122
A6	SG9	SG10	SG1	SG2	*PCIe4*		SG5	SG6	1111 2211	29	*SRIO2*				*SRIO1*				1111 1111
A6	SG9	SG10	SG1	SG2	*PCIe4*		SG5	SG6	1111 2211	2E	SRIO2				SRIO1				1111 1111
8E	SG9	SG10	*SG1*	*SG2*	PCIe4		SG5	SG6	1122 1111	29	*SRIO2*				*SRIO1*				1111 1111
8F	SG9	SG10	*SG1*	*SG2*	PCIe4		SG5	SG6	1122 1111	2D	SRIO2				SRIO1				1111 1111
82	*SG9*	*SG10*	*SG1*	*SG2*	PCIe4		SG5	SG6	1111 2222	29	*SRIO2*				*SRIO1*				1111 1111
83	*SG9*	*SG10*	*SG1*	*SG2*	PCIe4		SG5	SG6	1111 2222	2D	SRIO2				SRIO1				1111 1111
A4	SG9	SG10	SG1	SG2	*PCIe4*				1111 1111	16	PCIe1				PCIe2		SATA1	SATA2	1111 1122
96	SG9	SG10	*SG1*	SG2	PCIe4				1121 1111	16	PCIe1				PCIe2		SATA1	SATA2	1111 1122
8A	SG9	SG10	*SG1*	*SG2*	PCIe4				1122 1111	16	PCIe1				PCIe2		SATA1	SATA2	1111 1122
AB	PCIe3				PCIe4				1111 1111	1F	*PCIe1*				PCIe2				1111 2222
DA	PCIe3								1111 1111	02	PCIe1								1111 1111
D9	PCIe3	SG10	SG1	SG2	PCIe4	SG4	SG5	SG6	1111 2111	1F	*PCIe1*				PCIe2				1111 2222
D3	PCIe3	SG10	*SG1*	SG2	PCIe4	SG4	SG5	SG6	1121 1111	1F	*PCIe1*				PCIe2				1111 2222
CB	PCIe3	SG10	*SG1*	*SG2*	PCIe4	SG4	SG5	SG6	1122 1111	1F	*PCIe1*				PCIe2				1111 2222

Table 19-2. SerDes Lanes Assignments and Multiplexing

SERDES1										SERDES2									
SRDS_PRTCL_S1	A	B	C	D	E	F	G	H	Per[1] Lane PLL Mapping	SRDS_PRTCL_S2	A	B	C	D	E	F	G	H	Per[1] Lane PLL Mapping
D8	PCIe3	SG10	SG1	SG2	*PCIe4*	SG4	SG5	SG6	1111 2111	16	PCIe1				PCIe2	PCIe2	SATA1	SATA2	1111 1122
6E	XFI9	XFI10	SG1	SG2	PCIe4		SG5	SG6	1122 2222	18	PCIe1				Aurora		SATA1	SATA2	1111 1122
BC	PCIe3		SG1	SG2	*PCIe4*				1111 1111	18	PCIe1				Aurora		SATA1	SATA2	1111 1122
C8	PCIe3	SG10	*SG1*	*SG2*	PCIe4		SG5	SG6	1122 1111	18	PCIe1				Aurora		SATA1	SATA2	1111 1122
D6	PCIe3	SG10	SG1	SG2	*PCIe4*		SG5	SG6	1111 2211	18	PCIe1				Aurora		SATA1	SATA2	1111 1122
DE	PCIe3				*PCIe4*	PCIe1	PCIe2	SG6	1111 1111	36	SRIO2				Aurora		SATA1	SATA2	1111 1122
E0	PCIe3				*PCIe4*	PCIe1	SG5	SG6	1111 1111	36	SRIO2				Aurora		SATA1	SATA2	1111 1122
F2	PCIe3	SG10	SG1	SG2	*PCIe4*	PCIe1	PCIe2	SG6	1111 1111	36	SRIO2				Aurora		SATA1	SATA2	1111 1122
F8	PCIe3	SG10	*SG1*	*SG2*	PCIe4	PCIe1	PCIe2	SG6	1122 1111	36	SRIO2				Aurora		SATA1	SATA2	1111 1122
FA	PCIe3	SG10	*SG1*	*SG2*	PCIe4	PCIe1	SG5	SG6	1122 1111	36	SRIO2				Aurora		SATA1	SATA2	1111 1122
CB	PCIe3	SG10	*SG1*	*SG2*	PCIe4	SG4	SG5	SG6	1122 1111	1F	*PCIe1*				PCIe2				1111 2222
67	XFI9	XFI10	XFI1	XFI2	*PCIe4*				1111 2222	1F	*PCIe1*				PCIe2				1111 2222
AB	PCIe3				*PCIe4*				1111 1111	02	PCIe1								1111 1111
AB	PCIe3				*PCIe4*				1111 1111	29	*SRIO2*				*SRIO1*				1111 1111
DB	PCIe3								1111 1111	1F	*PCIe1*				PCIe2				1111 2222
DB	PCIe3								1111 1111	2D	SRIO2				SRIO1				1111 1111
66	XFI9	XFI10	XFI1	XFI2	*PCIe4*				1111 2222	16	PCIe1				PCIe2	PCIe2	SATA1	SATA2	1111 1122

1 PLL1 can drive each of the lanes A to H; PLL2 can drive each of the lanes C to H

Maximum Speed 8G

Maximum Speed 5G

19.4.2 Rules for disabling a SerDes module

In order to disable a serdes the following should be configured (x is 1 or 2):

SRDS_PLL_PD_Sx = 2'b11 , SRDS_PLL_REF_CLK_SEL_Sx = 2'b00 , SRDS_PRTCL_Sx=2

Powerdown of each PLL is done through RCW. Powerdown of specific lanes can be achived by software. PLL mapping column defines which PLL is related to which lane

19.4.3 SerDes 1 Lane Assignments

Table 19-3. SerDes 1 Lane/Signal Assignments

SerDes 1 Lane	Signal	Controller Lane Assignments			
		PCI Express			
		PCIe1	PCIe2	PCIe3	PCIe4
A	SD1_TX[0]/SD1_TX[0]_B SD1_RX[0]/SD1_RX[0]_B			0	
B	SD1_TX[1]/SD1_TX[1]_B SD1_RX[1]/SD1_RX[1]_B			1	
C	SD1_TX[2]/SD1_TX[2]_B SD1_RX[2]/SD1_RX[2]_B			2	
D	SD1_TX[3]/SD1_TX[3]_B SD1_RX[3]/SD1_RX[3]_B			3	
E	SD1_TX[4]/SD1_TX[4]_B SD1_RX[4]/SD1_RX[4]_B			4	0
F	SD1_TX[5]/SD1_TX[5]_B SD1_RX[5]/SD1_RX[5]_B	0		5	1
G	SD1_TX[6]/SD1_TX[6]_B SD1_RX[6]/SD1_RX[6]_B		0	6	2
H	SD1_TX[7]/SD1_TX[7]_B SD1_RX[7]/SD1_RX[7]_B			7	3

19.4.4 SerDes 2 Lane Assignments

Table 19-4. SerDes 2 Lane/Signal Assignments

SerDes 2 Lane	Signal	Controller Lane Assignments								
		PCI Express				SATA		SRIO		Aurora
		PCIe1	PCIe2	PCIe3	PCIe4	SATA1	SATA2	SRIO1	SRIO2	
A	SD2_TX[0]/SD2_TX[0]_B SD2_RX[0]/SD2_RX[0]_B	0							3	
B	SD2_TX[1]/SD2_TX[1]_B SD2_RX[1]/SD2_RX[1]_B	1							2	
C	SD2_TX[2]/SD2_TX[2]_B SD2_RX[2]/SD2_RX[2]_B	2							1	
D	SD2_TX[3]/SD2_TX[3]_B SD2_RX[3]/SD2_RX[3]_B	3							0	

Table 19-4. SerDes 2 Lane/Signal Assignments (continued)

SerDes 2 Lane	Signal	Controller Lane Assignments								
		PCI Express				SATA		SRIO		Aurora
		PCIe1	PCIe2	PCIe3	PCIe4	SATA1	SATA2	SRIO1	SRIO2	
E	SD2_TX[4]/SD2_TX[4]_B SD2_RX[4]/SD2_RX[4]_B	4	0					3		1
F	SD2_TX[5]/SD2_TX[5]_B SD2_RX[5]/SD2_RX[5]_B	5	1					2		0
G	SD2_TX[6]/SD2_TX[6]_B SD2_RX[6]/SD2_RX[6]_B	6	2			0		1		
H	SD2_TX[7]/SD2_TX[7]_B SD2_RX[7]/SD2_RX[7]_B	7	3				0	0		

19.5 Reference Clocks for SerDes Protocols

Each SerDes protocol allows for a finite set of valid SerDes-related RCW fields and reference clock frequencies, as shown in Table 19-5.

Table 19-5. Valid SerDes Reference Clocks and RCW Encodings

SerDes Protocol (given lane)	Valid reference clock frequency	Valid setting as determined by SRDS_PRTCL_S*n*	Valid setting as determined by SRDS_PLL_REF_CLK_SEL_S*n*	Valid setting as determined by SRDS_DIV_[prot]_S*n*
SGMII (1.25 Gbps)	100 MHz	SGMII @ 1.25 Gbps	0: 100 MHz	Don't Care
	125 MHz		1: 125 MHz	
2.5x SGMII (3.125 Gbps)	125 MHz	SGMII @ 3.125 Gbps	0: 125 MHz	Don't Care
	156.25 MHz		1: 156.25 MHz	
XAUI (3.125 Gbps)	125 MHz	XAUI @ 3.125 Gbps	0: 125 MHz	Don't Care
	156.25 MHz		1: 156.25 MHz	
HiGig (3.125 Gbps)	125 MHz	HiGig @ 3.125 Gbps	0: 125 MHz	Don't Care
	156.25 MHz		1: 156.25 MHz	
HiGig2 (3.75 Gbps)	125 MHz	HiGig2 @ 3.75 Gbps	0: 125 MHz	Don't Care
	156.25 MHz		1: 156.25 MHz	
XFI (10.3125 Gbps)	156.25 Mhz	XFI @ 10.3125 Gbps	0: 156.25 MHz	Don't Care
PCI Express 2.5 Gbps (doesn't negotiate upwards)	100 MHz (1)	Any PCIe	0: 100 MHz	2'b10:2.5 G
	125 MHz (1)		1: 125 MHz	
PCI Express 5 Gbps (can negotiate up to 5 Gbps)	100 MHz (1)	Any PCIe	0: 100 MHz	2'b01:5.0 G
	125 MHz (1)		1: 125 MHz	

Table 19-5. Valid SerDes Reference Clocks and RCW Encodings (continued)

SerDes Protocol (given lane)	Valid reference clock frequency	Valid setting as determined by SRDS_PRTCL_S*n*	Valid setting as determined by SRDS_PLL_REF_CLK_SEL_S*n*	Valid setting as determined by SRDS_DIV_[prot]_S*n*
PCI Express 8 Gbps (can negotiate up to 8 Gbps)	100 MHz (1)	Any PCIe	0: 100 MHz	2'b00:8.0 G
	125 MHz (1)		1: 125 MHz	
Serial RapidIO 2.5 Gbps	100 MHz	SRIO @ 2.5/5 Gbps	0: 100 MHz	1: 2.5 G
	125 MHz		1: 125 MHz	
Serial RapidIO 3.125 Gbps	125 MHz	SRIO @ 3.125 Gbps	0: 125 MHz	Don't Care
	156.25 MHz		1: 156.25 MHz	
Serial RapidIO 5 Gbps	100 MHz	SRIO @ 2.5/5 Gbps	0: 100 MHz	0: 5.0 G
	125 MHz		1: 125 MHz	
SATA (1.5 or 3 Gbps)	100 MHz	Any SATA	0: 100 MHz	Don't Care (2)
	125 MHz		1: 125 MHz	
Debug (2.5 Gbps)	100 MHz	Aurora @ 2.5/5 Gbps	0: 100 MHz	1: 2.5 G
	125 MHz		1: 125 MHz	
Debug (3.125 Gbps)	125 MHz	Aurora @ 3.125 Gbps	0: 125 MHz	Don't Care
	156.25 MHz		1: 156.25 MHz	
Debug (5 Gbps)	100 MHz	Aurora @ 2.5/5 Gbps	0: 100 MHz	0: 5.0 G
	125 MHz		1: 125 MHz	

Notes:
1. A spread-spectrum reference clock is permitted for PCI Express. However, if any other high speed interface such as SRIO, SATA, or Debug is used concurrently on the same SerDes, spread-spectrum clocking is not permitted.
2. SerDes lanes configured as SATA initially operate at 3.0 Gbps. 1.5 Gbps operation may be enabled later through the SATA IP itself. It is possible for software to set each SATA at different rates.

19.6 Memory Map/Register Definition

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.

19.6.1 Detailed Memory Map

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 0x000 to 0xFFF (4 KB).

Table 19-6. SerDes Memory Map

Offset	Register	Access	Reset	Section/Page
General SerDes Control Registers				
0x000	SerDes PLL1 Reset Control Register (SRDSxPLL1RSTCTL)	Special	0x0n47_4500	19.6.2.1.1/19-12
0x004	SerDes PLL1 Control Register 0 (SRDSxPLL1CR0)	R/W	0xn0nn_0000	19.6.2.1.2/19-13
0x008	SerDes PLL1 Control Register 1 (SRDSxPLL1CR1)	R/W	0x0n00_4100	19.6.2.1.3/19-15
0x018-0x01F	Reserved	—	—	—
0x020	SerDes PLL 2 Reset Control Register (SRDSxPLL2RSTCTL)	R/W	0x0n47_4500	19.6.2.1.1/19-12
0x024	SerDes PLL 2 PLL Control Register 0 (SRDSxPLL2CR0)	R/W	0xn0nn_0000	19.6.2.1.2/19-13
0x028	SerDes PLL 2 PLL Control Register 1 (SRDSxPLL2CR1)	R/W	0x0n00_4100	19.6.2.1.3/19-15
0x038 - 0x08F	Reserved	—	—	—
0x090	SerDes Transmit Calibration Control Register (SRDSxTCALCR)	R/W	0xn000_0000	19.6.2.2.1/19-16
0x094 - 0x09F	Reserved	—	—	—
0x0A0	SRDSx Receive Calibration Control Register (SRDSxRCALCR)	R/W	0xn000_0000	19.6.2.2.2/19-17
0x0A4 - 0x0AF	Reserved	—	—	—
0x0B0	SRDSx General Control Register 0 (SRDSxGR0)	R/W	0x0nnn_n000	19.6.2.2.3/19-17
0x0B4 - 0x0DF	Reserved	—	—	—
0x0E0	SRDSx Protocol Configuration Register 0 (SRDSxPCCR0)	R/W	0xnnnn_nnnn	19.6.2.3.1/19-18
0x0E4	SRDSx Protocol Configuration Register 1 (SRDSxPCCR1)	R/W	0xnnnn_nnnn	19.6.2.3.2/19-20
0x0E8	SRDSx Protocol Configuration Register 2 (SRDSxPCCR2)	R/W	0xnnnn_nnnn	19.6.2.3.3/19-22
0x0F0	SRDSx Protocol Configuration Register 4 (SRDSxPCCR4)	R/W	0xnnnn_nnnn	19.6.2.3.4/19-24
0x0F4 - 0x0FF	Reserved	—	—	—
Protocol Global Registers				
0x100	SRDSx Lane 0 Protocol Select Status Register 0 (SRDSxLN0PSSR0)	R	0xnnnn_nnnn	19.6.2.3.5/19-25
0x120	SRDSx Lane 1 Protocol Select Status Register 0 (SRDSxLN1PSSR0)	R	0xnnnn_nnnn	19.6.2.3.5/19-25
0x140	SRDSx Lane 2 Protocol Select Status Register 0 (SRDSxLN2PSSR0)	R	0xnnnn_nnnn	19.6.2.3.5/19-25
0x160	SRDSx Lane 3 Protocol Select Status Register 0 (SRDSxLN3PSSR0)	R	0xnnnn_nnnn	19.6.2.3.5/19-25
0x180	SRDSx Lane 4 Protocol Select Status Register 0 (SRDSxLN4PSSR0)	R	0xnnnn_nnnn	19.6.2.3.5/19-25
0x1a0	SRDSx Lane 5 Protocol Select Status Register 0 (SRDSxLN5PSSR0)	R	0xnnnn_nnnn	19.6.2.3.5/19-25

Table 19-6. SerDes Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x1c0	SRDSx Lane 6 Protocol Select Status Register 0 (SRDSxLN6PSSR0)	R	0xnnnn_nnnn	19.6.2.3.5/19-25
0x1e0	SRDSx Lane 7 Protocol Select Status Register 0 (SRDSxLN7PSSR0)	R	0xnnnn_nnnn	19.6.2.3.5/19-25
0x1e4-0x2FF	Reserved	—	N/A	N/A
0x300	SRDSx PCIe Equalization Configuration Register (SRDSxPEXEQCR)	R/W	0x2000_0C10	19.6.2.3.6/19-27
0x304	SRDSx PCIe Equalization Preset 0 Register (SRDSxPEXEQP0CR)	R/W	0x0000_C000	19.6.2.3.7/19-27
0x308	SRDSx PCIe Equalization Preset 1 Register (SRDSxPEXEQP1CR)	R/W	0x0000_0000	19.6.2.3.7/19-27
0x30C	SRDSx PCIe Equalization Preset 2 Register (SRDSxPEXEQP2CR)	R/W	0x0000_A000	19.6.2.3.7/19-27
0x310	SRDSx PCIe Equalization Preset 3 Register (SRDSxPEXEQP3CR)	R/W	0x0000_6000	19.6.2.3.7/19-27
0x314	SRDSx PCIe Equalization Preset 4 Register (SRDSxPEXEQP4CR)	R/W	0x0000_0000	19.6.2.3.7/19-27
0x318	SRDSx PCIe Equalization Preset 5 Register (SRDSxPEXEQP5CR)	R/W	0x0000_0005	19.6.2.3.7/19-27
0x31C	SRDSx PCIe Equalization Preset 6 Register (SRDSxPEXEQP6CR)	R/W	0x0000_0006	19.6.2.3.7/19-27
0x320	SRDSx PCIe Equalization Preset 7 Register (SRDSxPEXEQP7CR)	R/W	0x0000_9005	19.6.2.3.7/19-27
0x324	SRDSx PCIe Equalization Preset 8 Register (SRDSxPEXEQP8CR)	R/W	0x0000_6006	19.6.2.3.7/19-27
0x328	SRDSx PCIe Equalization Preset 9 Register (SRDSxPEXEQP9CR)	R/W	0x0000_0008	19.6.2.3.7/19-27
0x32C	SRDSx PCIe Equalization Preset 10 Register (SRDSxPEXEQP10CR)	R/W	0x0000_A006	19.6.2.3.7/19-27
Protocol Control Registers				
0x400	SRDSx PEXa Protocol Control Register 0	R/W	0x2000_0014	19.6.2.4.1/19-29
0x440-0x47F	SRDSx PEXb Protocol Control Registers	R/W	0x2000_0014	19.6.2.4/19-28
0x480-0x4BF	SRDSx PEXc Protocol Control Registers	R/W	0x2000_0014	19.6.2.4/19-28
0x4C0-0x4FF	SRDSx PEXd Protocol Control Registers	R/W	0x2000_0014	19.6.2.4/19-28
0x600	SGMIIA Protocol Control Register 0	R/W		19.6.2.5.1/19-30
0x610	SGMIIB Protocol Control Register 0	R/W		19.6.2.5.1/19-30
0x620	SGMIIC Protocol Control Register 0	R/W		19.6.2.5.1/19-30
0x630	SGMIID Protocol Control Register 0	R/W		19.6.2.5.1/19-30

Table 19-6. SerDes Memory Map (continued)

Offset	Register	Access	Reset	Section/Page
0x640	SGMIIE Protocol Control Register 0	R/W		19.6.2.5.1/19-30
0x650	SGMIIF Protocol Control Register 0	R/W		19.6.2.5.1/19-30
0x660	SGMIIG Protocol Control Register 0	R/W		19.6.2.5.1/19-30
0x670	SGMIIH Protocol Control Register 0	R/W		19.6.2.5.1/19-30
0x604	SGMIIA Protocol Control Register 1	R/W		19.6.2.5.2/19-30
0x614	SGMIIB Protocol Control Register 1	R/W		19.6.2.5.2/19-30
0x624	SGMIIC Protocol Control Register 1	R/W		19.6.2.5.2/19-30
0x634	SGMIID Protocol Control Register 1	R/W		19.6.2.5.2/19-30
0x644	SGMIIE Protocol Control Register 1	R/W		19.6.2.5.2/19-30
0x654	SGMIIF Protocol Control Register 1	R/W		19.6.2.5.2/19-30
0x664	SGMIIG Protocol Control Register 1	R/W		19.6.2.5.2/19-30
0x674	SGMIIH Protocol Control Register 1	R/W		19.6.2.5.2/19-30
Per-Lane Control and Status Registers				
0x800	General Control Register 0 - Lane 0 (SRDSxLN0GCR0)	R/W	0xnnnn_nn00	19.6.2.7.1/19-32
0x804	General Control Register 1 - Lane 0 (SRDSxLN0GCR1)	R/W	0xnnnn_00n0	19.6.2.7.2/19-36
0x80C	Speed Switch Control Register 0 - Lane 0 (SRDSxLN0SSCR0)	R/W	0xnnnn_nnnn	19.6.2.7.3/19-40
0x0810	Receive Equalization Control Register 0 - Lane 0 (SRDSxLN0RECR0)	R/W	0xnn0n_n01F	19.6.2.7.4/19-44
0x814	Reserved	—	—	—
0x814	Receive Equalization Control Register 1- Lane 0 (SRDSxLN0RECR1)	R/W	0xnn0n_nn0n	19.6.2.7.5/19-47
0x818	Transmit Equalization Control Register 0 - Lane 0 (SRDSxLN0TECR0)	R/W	0xn00n_40nn	19.6.2.7.6/19-48
0x81C	Speed Switch Control Register 1- Lane 0 (SRDSxLN0SSCR1)	R/W	0xnnnn_nnnn	19.6.2.7.7/19-52
0x820	TTL Control Register 0 - Lane 0 (SRDSxLN0TTLCR0)	R/W	0x1B00_n000	19.6.2.7.8/19-56
0x824-0x83F	Reserved	—	—	—
0x83C	Test Control/Status Register 3 - Lane 0 (SRDSxLN0TCSR3)	R/W	0x0n0n_nn00	19.6.2.7.9/19-57
0x840-0x87F	Registers for Lane 1	R/W	—	19.6.2.7/19-32
0x880-0x8BF	Registers for Lane 2	R/W	—	19.6.2.7/19-32
0x8C0-0x8FF	Registers for Lane 3	R/W	—	19.6.2.7/19-32
0x900-0x93F	Registers for Lane 4	R/W	—	19.6.2.7/19-32
0x940-0x97F	Registers for Lane 5	R/W	—	19.6.2.7/19-32
0x980-0x9BF	Registers for Lane 6	R/W	—	19.6.2.7/19-32
0x9C0-0x9F0	Registers for Lane 7	R/W	—	19.6.2.7/19-32

19.6.2 Memory-Mapped Register Descriptions

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).

19.6.2.1 Group/PLL Configuration Control and Status Registers

19.6.2.1.1 SRDS Reset Control Register (SRDSxPLL*n*RSTCTL)

SRDSxPLL*n*RSTCTL contains control and status bits for the SerDes PLL *n* reset state machine, as shown in Figure 19-1.

This register should not be modified while RST_DONE=0 and RST_ERR=0.

Figure 19-1. SRDS Reset Control Register (SRDSxPLL*n*RSTCTL)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	RSTREQ	RST_DONE	RST_ERR	—												
Reset	0000_nnn0_0100_0111															
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	—								PLLRST_B	SDRST_B	SDEN	—				
Reset	0100_0101_0000_0000															
Offset	SRDSxPLL1RSTCTL: 0x000 SRDSxPLL2RSTCTL: 0x020;															

Table 19-7. SRDSxPLL*n*RSTCTL Field Descriptions

Bits	Name	Description
0	RSTREQ	PLL Reset Request - write 1 self-clearing. 0: Not requesting PLL soft reset. 1: Requesting PLL soft reset. 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: Clearing this bit during the reset sequence has no effect.
1	RST_DONE	PLL Reset Done from Control Block State Machine 0: PLL reset sequence in progress. 1: PLL reset sequence complete.
2	RST_ERR	No PLL Lock before counter time_out 0: Normal function 1: PLL lock didn't happen in the expected time period
3-23	—	Reserved

Table 19-7. SRDSxPLL*n*RSTCTL Field Descriptions

24	PLLRST_B	PLL reset. Resets PLLn . 0: Force PLL reset 1: Application Mode, unless RSTREQ=1 Default value: 1 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.
25	SDRST_B	SRDS group reset. Resets all logic in the group of lanes operating from PLLn 0: Force SerDes group reset 1: Application Mode, unless RSTREQ=1 The SerDes reset state machine overrides this field to 0 in some states.
26	SDEN	SerDes enable. Enable the section of the SerDes module clocked by PLLn. 0: Force SerDes power down 1: Application Mode, unless RSTREQ=1 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.
27-31	—	Reserved

19.6.2.1.2 SRDSx PLL Control Register 0 (SRDSxPLL*n*CR0)

SRDS*x*PLLnCR0 contains control and status bits for SerDes PLL *n*, as shown in Figure 19-2.

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.

Figure 19-2. SRDSx PLL Control Register 0 (SRDSxPLL*n*CR0)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	POFF	RFCLK_SEL			RFCLK_EN	—			PLL_LCK	—			FRATE_SEL			
W																
Reset	nnnn_0000_0n00_nnnn															
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	—														DLYDIV_SEL	
W																
Reset	0000_0000_0000_0*nnn*															
Offset	SRDSxPLL1CR0: 0x004 SRDSxPLL2CR0: 0x024															

Table 19-8. SRDSxPLL*n*CR0 Field Descriptions

Bits	Name	Description
0	POFF	Power down an unused PLL 0: PLL on 1: PLL off. All lanes referencing this PLL must also be disabled. Default value: 0
1-3	RFCLK_SEL	Reference clock frequency select. 000: 100 MHz 001: 125 MHz 010: 156.25MHz 011: 150 MHz 100: 161.1328125 MHz All others reserved Default value set by RCW configuration. Recommended setting per protocol: • SATA: 000, 001 or 011 • PCIe: 000 or 001 • S-RIO @ 3.125: 001 or 010 • S-RIO @ , 2.5, 5: 000, 001 or 100 • Aurora @ 3.125: 001 or 010 • Aurora @ 2.5, 5: 000, 001 or 010 • SGMII @ 3.125: 001 or 010 • SGMII @ 1.25: 000, 001, or 010 • HiGig: 001 or 010 • HiGig2: 001 or 010 • XFI 010 or 100
4-7	—	Reserved
5-7	—	Reserved
8	PLL_LCK	Indicates PLL(n) has calibrated and locked 0: PLL is not locked 1: PLL is locked Read-only status bit.
9-11	—	Reserved

Table 19-8. SRDSxPLL*n*CR0 Field Descriptions

12-15	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). 0000: 5.00 GHz 0101: 3.75 GHz 0110: 5.15625 GHz 0111: 4 GHz 1001: 3.125 GHz 1010: 3.0 GHz All others are reserved Default value set by RCW configuration. Recommended settings per protocol: • PCIe: 0000 or 0111 (see Section 19.8.3, “Frequency Negotiation for details on PCIe speed negotiation) • SATA: 1010 • S-RIO: 0000 or 1001 • Aurora: 0000 or 1001 • SGMII: 0000 or 1001 • XAUI: 1001 • XFI: 0110 • HiGig: 1001 • HiGig2: 1001 or 0101
16-29	—	Reserved
30-31	DLYDIV_SEL	Select pllN_ex_dly_clk divider value: 00: pllN_ex_dly_clk off 01: pllN_ex_dly_clk is FRATE_SEL/16 (e.g. 312.5 MHz for 5.0 GHz FRATE_SEL) All other values reserved This feature is used for 1000Base-KX. When those modes are not active, it should be off.

19.6.2.1.3 SRDSx PLL Control Register 1 (SRDSxPLL*n*CR1)

SRDS*x*PLLnCR1 contains control bits for SerDes PLL *n*, as shown in Figure 19-3.

Figure 19-3. SRDSx PLL Control Register 1 (SRDSxPLL*n*CR1)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	—				PLLBW_SEL	—					VCO_EN	—				
Reset	0000_n000_00n0_0000															
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	—															
Offset	SRDSxPLL1CR1: 0x008; SRDSxPLL2CR1: 0x028;															

Table 19-9. SRDSxPLL*n*CR1 Field Descriptions

Bits	Name	Description
0-3	—	Reserved
4	PLLBW_SEL	Select Higher pll(n) bandwidth. 0: Nominal PLL Bandwidth 1: PLL Bandwidth Setting Recommended setting per PLL: 1
5-9	—	Reserved
10	VCO_EN	Select VCO for use in PLLn 0: Use LC VCO 1: Use ring VCO Default value set by RCW configuration. Recommended settings per SRDSxPLLnCR0[FRATE_SEL] and SRDSxLNnGCR0[T/RRAT_SEL]: • 5.00 GHz: full/half/quarter speed: 0 or 1 • 3.75 GHz full speed: 1 • 5.15625 GHz double speed: 0 • 4 GHz double speed: 0 • 3.125 GHz full speed: 1 • 3.0 GHz full/half speed: 1 All others reserved Note: running two LC VCOs at the same frequency is not supported on the same SERDES, or across Serdes 1 and 2.
11-31	—	Reserved

19.6.2.2 SerDes General Control and Status Registers

19.6.2.2.1 SRDSx Transmit Calibration Control Register (SRDSxTCALCR)

SRDSxTCALCR contains the control bits used for Transmit Calibration, as shown in Figure 19-4.

Figure 19-4. SRDSx Transmit Calibration Control Register (SRDSxTCALCR)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	—				CALR ST_B	—										
Reset	0nn0_0000_0000_0000															
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	—															
Reset	0000_0000_0000_0000															
Offset	0x090															

Table 19-10. SRDSxTCALCR Field Descriptions

Bits	Name	Description
0-3	—	Reserved
4	CALRST_B	Reset to the transmit calibration logic in the right endcap cell 0: Reset 1: Application Mode During POR (warm reset sequence) it is active and is released when first PLL comes out of reset
5-31	—	Reserved

19.6.2.2.2 SRDSx Receive Calibration Control Register (SRDSxRCALCR)

SRDSxRCALCR contains the control bits for Receive Calibration, as shown in Figure 19-5.

Figure 19-5. SRDSx Receive Calibration Control Register (SRDSxRCALCR)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	—				CALRST_B	—										
Reset	0nn0_0000_0000_nn00															
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	—															
Reset	0000_0000_0000_0000															
Offset	0x0A0															

Table 19-11. SRDSxRCALCR Field Descriptions

Bits	Name	Description
0-3	—	Reserved
4	CALRST_B	Reset to the receiver calibration logic in the left endcap cell 0 = Reset 1 = Application Mode During POR (warm reset sequence) it is active and is released when first PLL comes out of reset
5-31	—	Reserved

19.6.2.2.3 SRDSx General Register 0 (SRDSxGR0)

SRDSxGR0 contains general control/status bits for the SerDes, as shown in Figure 19-6.

Figure 19-6. SRDSx General Register 0 (SRDSxGR0)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	—															
Reset	0000_00nn_nnnn_nnnn															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	—	XPAD_SEL	—													
Reset	0n00_0000_0000_0000															
Offset	0x0B0															

Table 19-12. SRDSxGR0 Field Descriptions

Bits	Name	Description
0-16	—	Reserved
17	XPAD_SEL	Describes to SerDes module the value of the power supply being used by the Serdes I/Os: 0 = High xpadvdd supply (nominal 1.5V - see H/W spec for details) 1 = Low xpadvdd supply (nominal 1.35V - see H/W spec for details) 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.
18-31	—	Reserved

19.6.2.3 Protocol Configuration Control and Status Registers

19.6.2.3.1 SRDSx Protocol Configuration Register 0 (SRDSxPCCR0)

SRDSxPCCR0 contains controls for protocol configuration for PCIe and S-RIO, as shown in Figure 19-7.

Figure 19-7. SRDSx Protocol Configuration Register 0 (SRDSxPCCR0)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	—	PEXA_CFG			—	PEXB_CFG			—							
Reset	0nnn_0nnn_0000_0000															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	SRIOA_LRV	SRIOA_CFG			SRIOB_LRV	SRIOB_CFG			—							
Reset	0nnn_0nnn_0000_0000															
Offset	0x0E0															

Table 19-13. SRDSxPCCR0 Field Descriptions

Bits	Name	Description
0	—	Reserved
1-3	PEXA_CFG	PEXa Configuration: 000: disabled 001: x1 on Lane 0 (Serdes #1 only) 010: x2 [1:0] on lanes [1:0] 100: x4 [3:0] on lanes [3:0] 111: x8 [7:0] on lanes [7:0] All others reserved Default value set by RCW configuration.
4	—	Reserved
5-7	PEXB_CFG	PEXb Configuration: 000: disabled 001: x1 on lane 4 (Serdes #1 only) 010: x2 [1:0] on lanes [5:4] 011: x4 [3:0] on lanes [7:4] All others reserved Default value set by RCW configuration.
8-15	—	Reserved
16	SRIOA_LRV	SRIOa Lane Reversal 0: Lanes mapped as in SRIOA_CFG x2 [0:1] on lanes [7:6] or x4 [0:3] on lanes [7:4] 1: Lanes mapped in the reverse order as SRIOA_CFG, if SRIOa enabled as x2 or x4. x2 [0:1] on lanes [6:7] or x4 [0:3] on lanes [4:7] Note: the lane marked as lane 0 in SRIOA_CFG must remain powered up regardless of lane reversal. Default value set by RCW configuration. This field must be 0 for Serdes #1
17-19	SRIOA_CFG	SRIOa Configuration: 000: disabled 110: x2 [0:1] on lanes [7:6] 111: x4 [0:3] on lanes [7:4] All others reserved Default value set by RCW configuration. This field must be 0 for Serdes #1

Table 19-13. SRDSxPCCR0 Field Descriptions

20	SRIOB_LRV	SRIOa Lane reversal 0: Lanes mapped as in SRIOB_CFG x2 [0:1] on lanes [3:2 or x4 [0:3] on lanes [3:0] 1: Lanes mapped in the reverse order as SRIOB_CFG, if SRIOb enabled as x2 or x4. x2 [0:1] on lanes [2:3] or x4 [0:3] on lanes [0:3] Note: the lane marked as lane 0 in SRIOB_CFG must remain powered up regardless of lane reversal. Default value set by RCW configuration. This field must be 0 for Serdes #1
21-23	SRIOB_CFG	SRIOb Configuration: 000: disabled 101: x4 [0:3] on lanes [3:0] 111: x2 [0:1] on lanes [3:2] All others reserved Default value set by RCW configuration. This field must be 0 for Serdes #1
24-31	—	Reserved

19.6.2.3.2 SRDSx Protocol Configuration Register 1 (SRDSxPCCR1)

SRDSxPCCR1 contains the Protocol Configuration for SGMII , as shown in Figure 19-8.

This register is only valid for SerDes 1only.All fields are reserved for the other SerDes instances.

Figure 19-8. SRDSx Protocol Configuration Register 1 (SRDSxPCCR1)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	SGMIIA_CFG		SGMIIB_CFG		SGMIIC_CFG		SGMIID_CFG		SGMIIE_CFG		SGMIIF_CFG		SGMIIG_CFG		SGMIIH_CFG	
Reset	nnnn_nnnn_nnnn_nnnn															
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	SGMII A_KX	SGMII B_KX	SGMII C_KX	SGMII D_KX	SGMII E_KX	SGMII F_KX	SGMII G_KX	SGMII H_KX	—							
Reset	0000_0000_0000_0000															
Offset	0x0E4															

Table 19-14. SRDSxPCCR1 Field Descriptions

Bits	Name	Description
0-1	SGMIIA_CFG	SGMIIa Configuration: 00: disabled 10: x1 on Lane 4 to Mac 3 Any others reserved Default value set by RCW configuration. This field must be 0 for Serdes #2

Table 19-14. SRDSxPCCR1 Field Descriptions

2-3	SGMIIB_CFG	SGMIIb Configuration: 00: disabled 10: x1 on Lane 5 to Mac 4 Any others reserved Default value set by RCW configuration. This field must be 0 for Serdes #2
4-5	SGMIIC_CFG	SGMIIc Configuration: 00:disabled 10:x1 on Lane 6 to Mac 5 All others reserved Default value set by RCW configuration. This field must be 0 for Serdes #2
6-7	SGMIID_CFG	SGMIId Configuration: 00: disabled 10: x1 on Lane 7 to Mac 6 All others reserved Default value set by RCW configuration. This field must be 0 for Serdes #2
8-9	SGMIIE_CFG	SGMIIe Configuration: 00: disabled 01: x1 on Lane 0 to Mac 9 All others reserved Default value set by RCW configuration. This field must be 0 for Serdes #2
10-11	SGMIIF_CFG	SGMIIf Configuration: 00: disabled 01: x1 on Lane 1 to Mac 10 All others reserved Default value set by RCW configuration. This field must be 0 for serdes #2
12-13	SGMIIG_CFG	SGMIIg Configuration: 00: disabled 01: x1 on Lane 3 to Mac 1 All others reserved Default value set by RCW configuration. This field must be 0 for Serdes #2
14-15	SGMIIH_CFG	SGMIIh Configuration: 00: disabled 01: x1 on Lane 2 to Mac 2 All others reserved Default value set by RCW configuration. This field must be 0 for Serdes #2

Table 19-14. SRDSxPCCR1 Field Descriptions

16	SGMIIA_KX	SGMIIb 1000Base-KX Configuration: 0: SGMII mode 1: 1000Base-KX mode Note: this configuration bit must be set before performing any MDIO initialization of the PCS.
17	SGMIIB_KX	SGMIIc 1000Base-KX Configuration: 0: SGMII mode 1: 1000Base-KX mode Note: this configuration bit must be set before performing any MDIO initialization of the PCS.
18	SGMIIC_KX	SGMIId 1000Base-KX Configuration: 0: SGMII mode 1: 1000Base-KX mode Note: this configuration bit must be set before performing any MDIO initialization of the PCS.
19	SGMIID_KX	SGMIIe 1000Base-KX Configuration: 0: SGMII mode 1: 1000Base-KX mode Note: this configuration bit must be set before performing any MDIO initialization of the PCS.
20	SGMIIE_KX	SGMIIf 1000Base-KX Configuration: 0: SGMII mode 1: 1000Base-KX mode Note: this configuration bit must be set before performing any MDIO initialization of the PCS.
21	SGMIIF_KX	SGMIIf 1000Base-KX Configuration: 0: SGMII mode 1: 1000Base-KX mode Note: this configuration bit must be set before performing any MDIO initialization of the PCS.
22	SGMIIG_KX	SGMIIg 1000Base-KX Configuration: 0: SGMII mode 1: 1000Base-KX mode Note: this configuration bit must be set before performing any MDIO initialization of the PCS.
23	SGMIIH_KX	SGMIIh 1000Base-KX Configuration: 0: SGMII mode 1: 1000Base-KX mode Note: this configuration bit must be set before performing any MDIO initialization of the PCS.
24-31	—	Reserved

19.6.2.3.3 SRDSx Protocol Configuration Register 2 (SRDSxPCCR2)

SRDSxPCCR2 contains the Protocol Configuration for XAUI and XFI, as shown in Figure 19-9. Note that HiGig and HiGig2 use XAUI configuration.

This register is only valid for SerDes 1. All fields are reserved for the other SerDes instance.

Figure 19-9. SRDSx Protocol Configuration Register 2 (SRDSxPCCR2)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	XAUIA_LRV	XAUIA_XCF	—		XAUIB_LRV	XAUIB_CFG	—									
Reset	0n00_0n00_0000_0000															
Reset	0n00_0n00_0000_0000															
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	XFIA_CFG		—		XFIB_CFG		—		XFIC_CFG		—		XFID_CFG		—	
Offset	0x0E8															

Table 19-15. SRDSxPCCR2 Field Descriptions

Bits	Name	Description
0	XAUIA_LRV	XAUIa Lane Reversal 0: Lanes mapped as in XAUIA_CFG x4 [3:0] on lanes [3:0] 1: Lanes mapped in the reverse order as XAUIA_CFG, if XAUIa enabled. x4 [3:0] on lanes [0:3] Note: the lane marked as lane 0 in XAUIA_CFG must remain powered up regardless of lane reversal.
1	XAUIA_CFG	XAUIa Configuration: 0: Disabled 1: x4 [3:0] on lanes [3:0] to MAC 9 Default value set by RCW configuration. This field must be 0 for Serdes #2
2-3	—	Reserved
4	XAUIB_LRV	XAUIb Lane Reversal 0: Lanes mapped as in XAUIB_CFG x4 [3:0] on lanes [7:4] 1: Lanes mapped in the reverse order as XAUIB_CFG, if XAUIb enabled. x4 [3:0] on lanes [4:7] Note: the lane marked as lane 0 in XAUIB_CFG must remain powered up regardless of lane reversal.
5	XAUIB_CFG	XAUIb Configuration: 0: Disabled 1: x4 [3:0] on lanes [7:4] to MAC 10 Default value set by RCW configuration. This field must be 0 for Serdes #2
6-15	—	Reserved
16-17	XFIA_CFG	XFIa Configuration: 00: Disabled 01: x1 on Lane 3 to MAC 2 All others reserved Default value set by RCW configuration. This field must be 0 for Serdes #2
18-19	—	Reserved

Table 19-15. SRDSxPCCR2 Field Descriptions

20-21	XFIB_CFG	XFIb Configuration: 00: Disabled 01: x1 on Lane 2 to MAC 1 All others reserved Default value set by RCW configuration. This field must be 0 for serdes #2
22-23	—	Reserved
24-25	XFIC_CFG	XFIc Configuration: 00: Disabled 01: x1 on Lane 0 to MAC 9 All others reserved Default value set by RCW configuration. This field must be 0 for Serdes #2
26-27	—	Reserved
28-29	XFID_CFG	XFId Configuration: 00: Disabled 01: x1 on Lane 1 to MAC 10 All others reserved Default value set by RCW configuration. This field must be 0 for Serdes #2
30-31	—	Reserved

19.6.2.3.4 SRDSx Protocol Configuration Register 4 (SRDSxPCCR4)

SRDSxPCCR4 contains the Protocol Configuration forSATA, and Aurora, as shown in Figure 19-10.

This is relevant for SerDes 2 only.

Figure 19-10. SRDSx Protocol Converter Configuration Register 4 (SRDSxPCCR4)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	SATAA_CFG	—			SATAB_CFG	—										
Reset	n000_n000_0000_0000															
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	—												AUR_CFG			
Reset	0000_0000_0000_nnnn															
Offset	0x0F0															

Table 19-16. SRDSxPCCR4 Field Descriptions

Bits	Name	Description
0	SATAA_CFG	SATAa Configuration: 0: Disabled 1: x1 on Lane 6 Note: this field must be 0 for SerDes #1.

Table 19-16. SRDSxPCCR4 Field Descriptions

1-3	—	Reserved
4	SATAB_CFG	SATAb Configuration: 0: Disabled 1: x1 on Lane 7 Note: this field must be 0 for SerDes #1.
527	—	Reserved
28:31	AUR_CFG	AURORA Configuration 0000 = disabled 1001 = x4 [3:0] on lanes [4:7] All others reserved Note: this field must be 0 for SerDes #1.

19.6.2.3.5 SRDSx Lane m Protocol Select Status Register 0 (SRDSxLNmPSSR0)

SRDSxLNmPSSR0 (m=0-7) contains decoded protocol select information per lane, as shown in Figure 19-11.

The primary usage of this register is to map an Ethernet link to its corresponding MAC and MDIO address space. This register applies to SerDes 1 only.

Figure 19-11. SerDes Lane M Protocol Select Status Register 0 (SRDSxLNmPSSR0)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	—	TYPE							—			CN-TRL	—	MAC		
W																
Reset	0nnn_nnnn_000n_0nnn															
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	—					PCS			—				LANE			
W																
Reset	0000_0nnn_0000_nnnn															
Offset	SRDSxLN0PSSR0: 0x100 SRDSxLN1PSSR0: 0x120 SRDSxLN2PSSR0: 0x140 SRDSxLN3PSSR0: 0x160 SRDSxLN4PSSR0: 0x180 SRDSxLN5PSSR0: 0x1a0 SRDSxLN6PSSR0: 0x1c0 SRDSxLN7PSSR0: 0x1e0															

Table 19-17. SRDSxLN*n*PSSR0 Field Descriptions

Bits	Name	Description
0	—	Reserved
1-7	TYPE	Protocol Type: Bits 1-5 are same as SRDSxLNmGCR0[PROTS]. Bit 6-7: • 00
8-10	—	Reserved

Table 19-17. SRDSxLN*n*PSSR0 Field Descriptions

11	CNTRL	Controller Group Instance: 0: FM or non-ethernet
12	—	Reserved
13-15	MAC	MAC instance within CNTRL 000: MAC1 001: MAC2 010: MAC3 011: MAC4 100: MAC5 101: MAC6 110: MAC9 111: MAC10
16-20	—	Reserved
21-23	PCS	PCS instance of TYPE within PHY 000: PCSa/1 001: PCSb/2 010: PCSc/3 011: PCSd/4 100: PCSe/5 101: PCSf/6 110: PCSg/7 111: PCSh/8
24:26	—	Reserved
28:31	LANE	Lane number within PCS 0000: lane 0 0001: lane 1 0010: lane 2 0011: lane 3 0100: lane 4 0101: lane 5 0110: lane 6 0111: lane 7 All others reserved Example 1: for x4 PCIe [3:0] on lanes [3:0], LANE=0 for n=0, LANE=1 for n=1, LANE=2 for n=2 and LANE=3 for n=3 Note that the lane number within PCS does not take into account lane reversal, either via auto-negotiate, or by S/W control in SerDes registers.

19.6.2.3.6 SRDSx PCI Express Equalization Configuration Register (SRDSxPEXEQCR)

SRDSxPEXEQCR (y=0-10) contains the PCIe gen3 Tx equalization configuration values, as shown in Figure 19-11.

Figure 19-12. SerDes PCI Express Equalization Configuration Register (SRDSxPEXEQCR)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	—															
Reset	0000_0000_0000_0000															
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	—				FS						LF					
Reset	0000_1100_0001_0000															
Offset	SRDSxPEXEQCR: 0x300															

Table 19-18. SRDSxPEXEQCR Field Descriptions

Bits	Name	Description
0-19	—	Reserved
20-25	FS	PCI Express FS value. This field determines the FS value advertised during 8.0 GT/s link equalization phase 1, and is also used to calculate the preset C(0). Default value = 0x30 This value applies to all PCIe links on SRDSx
26-31	LF	PCI Express LF value. This field determines the LF value advertised during 8.0 GT/s link equalization phase 1. Default value =0x10 This value applies to all PCIe links on SRDSx

19.6.2.3.7 SRDSx PCI Express Equalization Preset Y Control Register (SRDSxPEXEQPyCR)

SRDSxPEXEQPyCR (y=0-10) contains the PCIe gen3 Tx equalization preset values, as shown in Figure 19-11. Note that C(0) is automatically calculated from C(-1), C(+1), and SRDSxPEXEQCR[FS].

Figure 19-13. SerDes PCI Express Preset Y Control Register (SRDSxPEXEQPyCR)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	—														CP1	
Reset	0000_0000_0000_00nn															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CP1				—						CM1					
Reset	nnnn_0000_00nn_nnnn															
Offset	SRDSxPEXEQP0CR: 0x304 SRDSxPEXEQP1CR: 0x308 SRDSxPEXEQP2CR: 0x30C SRDSxPEXEQP3CR: 0x310 SRDSxPEXEQP4CR: 0x314 SRDSxPEXEQP5CR: 0x318 SRDSxPEXEQP6CR: 0x31C SRDSxPEXEQP7CR: 0x320 SRDSxPEXEQP8CR: 0x324 SRDSxPEXEQP9CR: 0x328 SRDSxPEXEQP10CR: 0x32C															

Table 19-19. SRDSxPEXEQPyCR Field Descriptions

Bits	Name	Description
0-13	—	Reserved
14-19	CP1	C(+1) preset value. Default values for preset y= 0000: 6'b00_1100 0001: 6'b00_1000 0010: 6'b00_1010 0011: 6'b00_0110 0100: 6'b00_0000 0101: 6'b00_0000 0110: 6'b00_0000 0111: 6'b00_1001 1000: 6'b00_0110 1001: 6'b00_0000 1010: 6'b00_1010
20-25	—	Reserved
26-31	CM1	C(-1) preset value. Default values for preset y= 0000: 6'b00_0000 0001: 6'b00_0000 0010: 6'b00_0000 0011: 6'b00_0000 0100: 6'b00_0000 0101: 6'b00_0101 0110: 6'b00_0110 0111: 6'b00_0101 1000: 6'b00_0110 1001: 6'b00_1000 1010: 6'b00_0110

19.6.2.4 PCIe Protocol Control and Status Registers

The PCIe Protocol Control and Status Registers support the control and status bits related to the PCIe PCS layers and control logic.

19.6.2.4.1 SRDSx PEXn Protocol Control Register 0 (SRDSxPEXnCR0)

SRDSxPEXnCR0 (n=A,B,C,D) contains control bits used for the PEXn protocol logic, as shown in Figure 19-14.

Figure 19-14. SRDSx PEX*n* Protocol Control Register 0 (SRDSxPEXnCR0)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	—				RD_SW	—										
Reset	0010_0000_0000_0000															
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	—															
Reset	0000_0000_0000_0000															
Offset	SRDSxPEXACR0: 0x400 SRDSxPEXBCR0: 0x440 SRDSxPEXCCR0: 0x480 SRDSxPEXDCR0: 0x4C0															

19.6.2.4.2 SRDSx PEX*n* Protocol Control Register 7 (SRDSxPEXnCR7)

SRDSPEXnCR7 (n=A,B,C,D) contains status bits used for the PEXn protocol logic, as shown in Figure 19-15.

Figure 19-15. SRDSx PEX*n* Protocol Control Register 7 (SRDSxPEXnCR7)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	EQ-DIS	—														
Reset	0000_0000_0000_0000															
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	—															
Reset	0000_0000_0000_0000															
Offset	SRDSxPEXACR7: 0x420 SRDSxPEXBCR7: 0x460 SRDSxPEXCCR7: 0x4A0 SRDSxPEXDCR7: 0x4E0															

Table 19-20. SRDSx PEX*n* Protocol Control Register 7 (SRDSxPEXnCR7)

Bits	Name	Description
0	EQ_DIS	Rx EQ disable
1	—	Reserved

19.6.2.5 SGMII Protocol Configuration, Control and Status Registers

These registers apply only to Serdes 1.

19.6.2.5.1 SRDSx SGMII*n* Protocol Control Register 0 (SRDSxSGMII*n*CR0)

SRDSxSGMIInCR0 **(*n*=A,B,C,D,E,F,G,H)** contains control bits used for the SGMIIn protocol logic, as shown in Figure 19-16.

Figure 19-16. SRDSxSGMII*n*CR0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RST_SGM	PD_SGM	0	0		0	0	0	0	0	0	0	0	0	0	0
W																
Reset	n000_0000_0000_0000															
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	LPI_RST
Reset	0000_0000_0000_0000															
Offset	SRDSxSGMIIACR0: 0x600 SRDSxSGMIIBCR0: 0x610 SRDSxSGMIICCR0: 0x620 SRDSxSGMIIDCR0: 0x630 SRDSxSGMIIECR0: 0x640 SRDSxSGMIIFCR0: 0x650 SRDSxSGMIIGCR0: 0x660 SRDSxSGMIIHCR0: 0x670															

19.6.2.5.2 SRDSx SGMII*n* Protocol Control Register 1 (SRDSxSGMIInCR1)

SRDSxSGMIInCR1 **(*n*=A,B,C,D,E,F,G,H)** contains control bits used for the SGMIIn protocol logic, as shown in Figure 19-17.

Figure 19-17. SRDSxSGMIInCR1

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MDEV_PORT					—										
W																
Reset	0000_0000_0000_0000															
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	—				SGPCS_EN	—										
W																
R	0	0	0	0	SGPCS_EN	0	0	0	SPD		0	0	HD	0	0	0
W																
Reset	0000_n000_0000_0000															
Offset	SRDSxSGMIIACR1: 0x604 SRDSxSGMIIBCR1: 0x614 SRDSxSGMIICCR1: 0x624 SRDSxSGMIIDCR1: 0x634 SRDSxSGMIIECR1: 0x644 SRDSxSGMIIFCR1: 0x654 SRDSxSGMIIGCR1: 0x664 SRDSxSGMIIHCR1: 0x674															

Table 19-21. SRDSxSGMIInCR1 Field Descriptions

Bits	Name	Descriptions
0-4	MDEV_PORT	MDIO bus port address. When operating in SGMII mode, compared to MDIO_COMMAND[PHY Address] (Clause 22) to accept a command. When operating in 1000Base-KX mode, compared to the MDIO_COMMAND[Device Address] to accept a command. Recommended value: 0 Note: software must wait at least 3 platform clocks after changing this value before performing any MDIO accesses to the SGMIIn PCS.
5-19	—	Reserved
20	SGPCS_EN	SGMII PCS enable Default value: 1 if corresponding SGMII is enabled in SRDSxPCCSR1, else 0.
21-31	—	Reserved

19.6.2.5.3 SRDSx XAUI*n* Protocol Control Register 1 (SRDSxXAUInCR1)

SRDSxXAUI*n*CR1 **(*n*=A,B)** contains control bits used for the XAUIn protocol logic, as shown in Figure 19-18.

Figure 19-18. SRDSxXAUInCR1

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	ALGD	0	CG_ALIG				0	0	0	0	0	0	DISP_ERR			
W																
Reset	0000_0000_0000_0000															
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CHAR_ERR				0	0	SYNC				0	0	PAT			
W																
Reset	0000_0000_0000_0000															
Offset	SRDSxXAUIACR‘:0x06A4 SRDSxXAUBICR‘:0x06A4															

Table 19-22. SRDSXAUInCR1 Field Descriptions

Bits	Name	Descriptions
0	ALGD	Lane alignment Status. Asserted to indicate that the lanes are correctly synchronized, deasserted when the lane synchronizaton is lost or not accuired
1-31	—	Reserved

19.6.2.6 XFI*n* Protocol Control Registers

This section describes control and status registers for the XFI Protocol Control blocks.

19.6.2.6.1 XFI*n* Protocol Control Register 1 (SRDSxXFInCR1)

As shown in Figure 19-19, SRDSxXFInCR1 (n=A,B,C,D) contains the control bits used for the XFI*n* Protocol Converter

Figure 19-19. SRDSxXFICR1

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	MDEV_PORT					—										
Reset	0000_0000_0000_0000															
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W																
Reset	0000_0000_0000_0000															
Offset	SRDSxXFIACR1: 0x6C4 SRDSxXFIBCR1: 0x6D4 SRDSxXFICCR1: 0x6E4 SRDSxXFIDCR1: 0x6F4															

19.6.2.7 Per-lane SerDes Control/Status Registers

This section describes the configuration, control and status registers for the SerDes lanes.

19.6.2.7.1 SRDSx Lane m General Control Register 0 (SRDSxLNmGCR0)

As shown in Figure 19-20, SRDSxLNmGCR0 (m=0-7) contains functional control bits for SerDes lane m.

Special consideration must be taken when writing this register while SRDSxPLLnRSTCTL[RST_DONE]=0. See RRST_B and TRST_B fields for details.

Figure 19-20. SRDSx Lane m General Control Register 0 (SRDSxLNmGCR0)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	RPLL_LES	—	RRAT_SEL		TPLL_LES	—	TRAT_SEL		—	RRST_B	TRST_B	RX_PD	TX_PD	IF20BIT_EN	IACC_DIS	FIRST_LANE
Reset	n0nn_n0nn_0nnn_nn0n															
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	—		TTRM_VM_SEL		PROTS					—						
Reset	00nn_nnnn_n000_0000															
Offset	SRDSxLN0GCR0: 0x800 SRDSxLN1GCR0: 0x840 SRDSxLN2GCR0: 0x880 SRDSxLN3GCR0: 0x8C0 SRDSxLN4GCR0: 0x900 SRDSxLN5GCR0: 0x940 SRDSxLN6GCR0: 0x980 SRDSxLN7GCR0: 0x9C0															

Table 19-24. SRDSxLNmGCR0 Field Descriptions

Bits	Name	Description
0	RPLL_LES	Directs the RX portion of lane m to use the corresponding PLL. 0: If m=0-1, use PLL1 If m=2-7, use PLL2 1: If m=0-1, use PLL1 If m=2-7, use PLL1 Default value set by reset configuration. Note: for m=0-1, PLL2 cannot be selected. PLL1 is used regardless of this setting. Note: must be set the same as TPLL_LES for normal function. See Section 19.8.4.1, "Lane Reset and Reconfiguration," for details on the sequence required to change this setting.
1	—	Reserved
2-3	RRAT_SEL	Receiver speed selection for lane m, relative to the corresponding SRDSxPLL*n*CR0[FRATE_SEL], *n* as selected by SRDSxLNmGCR0[RPLL_LES]: 00: Same as FRATE_SEL 01: FRATE_SEL/2 10: FRATE_SEL/4 11: FRATE_SEL*2 Default value set by reset configuration. Required setting per protocol: SGMII: 10 (1.25 Gbaud) SGMII: 00 (3.125 Gbaud) XAUI: 00 XFI: 11 S-RIO: 01 (2.5 Gbaud) S-RIO: 00 (3.125/5 Gbaud) Aurora: 01 (2.5 Gbaud) Aurora: 00 (3.125/5 Gbaud) PCIe: N/A (receiver speed selection automatically selected by protocol) SATA: N/A (receiver speed selection automatically selected by protocol) This register value is overridden by the protocol logic when running in PCIe or SATA mode. Note: must be set the same as TRAT_SEL for normal function.
4	TPLL_LES	Used to direct the TX portion of lane to use the corresponding PLL. 0: If m=0-1, use PLL1 If m=2-7, use PLL2 1: If m=0-1, use PLL1 If m=2-7, use PLL1 Default value set by reset configuration. Note: for m=0-1, PLL2 cannot be selected. PLL1 is used regardless of this setting. Note: must be set the same as RPLL_LES for normal function. See Section 19.8.4.1, "Lane Reset and Reconfiguration," for details on the sequence required to change this setting.
5	—	Reserved

Table 19-24. SRDSxLNmGCR0 Field Descriptions

6-7	TRAT_SEL	Transmitter speed selection for lane m, relative to the corresponding SRDSxPLL*n*CR0[FRATE_SEL], *n* as selected by SRDSxLNmGCR0[TPLL_LES]: 00: Same as FRATE_SEL 01: FRATE_SEL/2 10: FRATE_SEL/4 11: FRATE_SEL*2 Default value set by reset configuration. Required setting per protocol: SGMII: 10 (1.25 Gbaud) SGMII: 00 (3.125 Gbaud) XAUI: 00 XFI: 11 S-RIO: 01 (2.5 Gbaud) S-RIO: 00 (3.125/5 Gbaud) Aurora: 01 (2.5 Gbaud) Aurora: 00 (3.125/5 Gbaud) PCIe: N/A (receiver speed selection automatically selected by protocol) SATA: N/A (receiver speed selection automatically selected by protocol) This register value is overridden by the protocol logic when running in PCIe or SATA mode. Note: must be set the same as RRAT_SEL for normal function.
8	—	Reserved
9	RRST_B	Resets receiver for lane m 0:Reset 1: Application Mode Recommended setting per protocol: 1, unless SRDSxPLLnRSTCTL[RST_DONE]=0, then 0. This register value is automatically set to 1 after SerDes reset (POR or SRDSxPLLnRSTCTL[RSTREQ]). This register is overridden when running in SATA or PCIe mode, unless srds_test_en. Used in lane reset and reconfiguring procedures. See Section 19.8.4.1, “Lane Reset and Reconfiguration,” for details.
10	TRST_B	Resets transmitter for lane m 0:Reset 1: Application Mode Recommended setting per protocol: 1 This register value is automatically set to 1 after SerDes reset (POR or SRDSxPLLnRSTCTL[RSTREQ]). This register is overridden when running in SATA or PCIe mode, unless srds_test_en. Used in lane reset and reconfiguring procedures. See Section 19.8.4.1, “Lane Reset and Reconfiguration,” for details.
11	RX_PD	Lane powerdown for receiver on lane m 0: Lane receiver active 1: Lane receiver powered down Default value set by reset configuration. This register value is overridden by the protocol controls during SerDes reset (POR or SRDSxPLLnRSTCTL[RSTREQ]). See Section 19.8.4.2, “Lane Enable After Powerdown for details on re-enabling powered down lanes.

Table 19-24. SRDSxLNmGCR0 Field Descriptions

12	TX_PD	Lane powerdown for transmitter on lane m 0: Lane transmitter active 1: Lane transmitter powered down Default value set by reset configuration. This register value is overridden by the protocol controls during SerDes reset (POR or SRDSxPLLnRSTCTL[RSTREQ]). 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 Section 19.8.4.2, "Lane Enable After Powerdown for details on e-enabling powered down lanes.
13	IF20BIT_EN	20-bit interface enable 0: 10-bit interface 1: 20-bit interface Default value set by reset configuration. Required value per protocol: PCIe : 1 S-RIO: 0 XFI: 1 XAUI: 0 SGMII: 0 Aurora: 0 SATA: 0
14	—	Reserved
15	FIRST_LANE	Indicates this lane is the first (lane 0) of a group of lanes 0: Lane m is not lane 0 of the link 1: Lane m is lane 0 of the link Default/ercommended value set by reset configuration
16-19	—	Reserved
18-19	TTRM_VM_SEL	Selects RX termination configuration on SerDes RX inputs: 00: If the SerDes is popwered down (SRDSxPLLnRSTCTL[SDEN] = 0) or the lane transmitter is powered down (SRDSxLNmGCR0[TX_PD] = 1), common mode impedance is Hi-Z, else common mode impedance is calibrated termination to xcorevss. 01: Common mode impedance is always calibrated termination to xcorevss. 10: Common mode impedance is Hi-Z, RX termination is uncalibrated 120 Ohm differential. 11: Common mode is 0.7*xcorevdd through 3 kOhm, RX termination is uncalibrated 120 Ohm differential. Recommended settings per protocol: PCIe: 00 S-RIO: 01 Aurora: 01 XAUI: 01 SATA: 01 XFI: 01

Table 19-24. SRDSxLNmGCR0 Field Descriptions

20-24	PROTS	Lane Protocol Select 0_0000: PCI EXP 0_0001 = SGMII, , SGMII 2.5x 0_0010 = SATA 0_0011 = S-RIO 0_0100 = XAUI 0_0101 = Aurora 0_0110 = HiGig, HiGig2 (3.125 Gbaud) 0_0111 = HiGig2 (3.75 Gbaud) 0_1010 =XFI All others reserved
25-31	—	Reserved

19.6.2.7.2 SRDSx Lane m General Control Register 1 (SRDSxLNmGCR1)

As shown in Figure 19-21, SRDSxLNmGCR1 (m=0-7) contains functional control bits for SerDes lane m

Figure 19-21. SRDSx Lane m General Control Register 1 (SRDSxLNmGCR1)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	RDAT_INV	TDAT_INV	—							REIDL_TH			REIDL_EX_SEL		REIDL_ET_SEL	
Reset	00nn_0n00_0nnn_0000															
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	REIDL_EX_MSB	REIDL_ET_MSB	—						TRSTDIR	—	ISLEW_RCTL		—		OSLEW_RCTL	
Reset	0000_0000_n000_0000															
Offset	SRDSxLN0GCR1:0x0804 SRDSxLN1GCR1:0x0844 SRDSxLN2GCR1:0x0884 SRDSxLN3GCR1:0x08C4 SRDSxLN4GCR1:0x0904 SRDSxLN5GCR1:0x0944 SRDSxLN6GCR1:0x0984 SRDSxLN7GCR1:0x09C4															

Table 19-25. SRDSxLNmGCR1 Field Descriptions

Bits	Name	Description
0	RDAT_INV	Invert Rx data. Has the same effect as swapping SD_RX[m]_P and SD_RX[m]_N. 0 Rx data is not inverted 1 Rx data is inverted before it is decoded
1	TDAT_INV	Invert Tx data. Has the same effect as swapping SD_TX[m]_P and SD_TX[m]_N. 0 Tx data is not inverted 1 Tx data is inverted before it is transmitted
2-4	—	Reserved
5	OPAD_CTL	TX Output pad control signal for common mode 0 Transmitter Enabled 1 Force Transmitter Output to Common Mode
6-8	—	Reserved
9-11	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 SRDSxLNmSSCR0[REIDL_TH_0). Note: PCIe 8 Gbaud setting taken from SRDSxLNmSSCR1[REIDL_TH_1).
12-13	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 SRDSxLNmSSCR0[REIDL_EX_SEL_0). Note: PCIe 8 Gbaud setting taken from SRDSxLNmSSCR1[REIDL_EX_SEL_1).

Table 19-25. SRDSxLNmGCR1 Field Descriptions

14-15	REIDL_ET_SEL	Enter idle filter select = {REIDL_ET_MSB,REIDL_ET_SEL}: Recommended setting per protocol: PCIe: 111 (2.5 Gbaud) SGMII: 001 (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 SRDSxLNmSSCR0[REIDL_ET_SEL_0). Note: PCIe 8 Gbaud setting taken from SRDSxLNmSSCR1[REIDL_ET_SEL_1).
16	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 SRDSxLNmSSCR0[REIDL_EX_MSB_0). Note: PCIe 8 Gbaud setting taken from SRDSxLNmSSCR1[REIDL_EX_MSB_1).
17	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 SRDSxLNmSSCR0[REIDL_ET_MSB_0). Note: PCIe 8 Gbaud setting taken from SRDSxLNmSSCR1[REIDL_ET_MSB_1).
18	REQ_CTL_SNP	Initiate snapshot of RX Equalization Control Gaink2, Gaink3, and Offset Registers Recommended Setting per protocol: 0
19	REQ_CDR_SNP	Initiate snapshot of RX Clock/Data Recovery (CDR) Registers Recommended Setting per protocol: 0

Table 19-25. SRDSxLNmGCR1 Field Descriptions

20-23	—	Reserved
24	TRSTDIR	Multi-lane protocol Tx clock synchronization control Default value set by RCW configuration. Recommended setting per protocol: PCIe: 1 XAUI: 1 S-RIO: 0 Aurora: 0 Others: Don't care (single-lane protocols)
26-27	ISLEW_RCTL	Slew control for Quadrature Generator Default value set by RCS configuration. Recommended setting per protocol: PCIe: 01 (2.5 Gbaud) S-RIO: 01 (2.5/5 Gbaud) S-RIO: 10 (3.125 Gbaud) SGMII: 01 (1.25 Gbaud) SGMII: 10 (3.125 Gbaud) XAUI: 10 HiGig/2: 01 (3.125/3.75 Gbaud) XFI: 01 SATA: 10 (1.5 Gbaud) Aurora: 01 (2.5/5 Gbaud Aurora: 10 (3.125 Gbaud Note: SATA 3 Gbaud and PCIe 5 Gbaud setting taken from SRDSxLNmSSCR0[ISLEW_RCTL_0). Note: PCIe 8 Gbaud setting taken from SRDSxLNmSSCR1[ISLEW_RCTL_1).
28-29	—	Reserved
30-31	OSLEW_RCTL	Phase Interpolator Output clock edge rate control Default value set by RCW configuration. Recommended setting per protocol: PCIe: 01 (2.5 Gbaud) S-RIO: 01 (2.5/5 Gbaud) S-RIO: 10 (3.125 Gbaud) SGMII: 01 (1.25 Gbaud) SGMII: 10 (3.125 Gbaud) XAUI: 10 HiGig: 10 HiGig2: 10 (3.125 Gbaud) HiGig2: 01 (3.75 Gbaud) XFI: 01 SATA: 10 (1.5 Gbaud) Aurora: 01 (2.5/5 Gbaud Aurora: 10 (3.125 Gbaud Note: SATA 3 Gbaud and PCIe 5 Gbaud setting taken from SRDSxLNmSSCR0[ISLEW_RCTL_0). Note: PCIe 8 Gbaud setting taken from SRDSxLNmSSCR1[ISLEW_RCTL_1).

19.6.2.7.3 SRDSx Lane m Speed Switch Control Register 0 (SRDSxLNmSSCR0)

As shown in Figure 19-22, SRDSxLNmSSCR0 (m=0-7) contains control bits for modifying the PCI Express 5 Gbaud and SATA 3 Gbaud behavior of SerDes lane m.

Figure 19-22. SRDSx Lane m Speed Switch Control Register 0 (SRDSxLNmSSCR0)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	REIDL_TH_0			REIDL_EX_SEL_0		REIDL_ET_SEL_0		REIDL_EX_MSB_0	REIDL_ET_MSB_0	—				RXEQ_BST_0	BASE_WAND_0	
Reset	nnnn_nnnn_nnnn_nnnn															
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	OSETOVD6_0	TEQ_TYPE_0		SGN_PREQ_0	SGN_POST1Q_0	RATIO_PST1Q_0					AMP_RED_0					
Reset	nnnn_nnnn_nnnn_nnnn															
Offset	SRDSxLN0SSCR0: 0x80C SRDSxLN1SSCR0: 0x84C SRDSxLN2SSCR0: 0x88C SRDSxLN3SSCR0: 0x8CC															

Table 19-26. SRDSxLNmSSCR0 Field Descriptions

Bits	Name	Description
0-2	REIDL_TH_0	Receiver electrical idle detection threshold control Default settings set per RCW configuration. Recommended settings per protocol: 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 SRDSxLNmGCR1[REIDL_TH]. For PCIe 8 Gbaud , see SRDSxLNmSSCR1[REIDL_TH_1].
3-4	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 SRDSxLNmGCR1[REIDL_EX_SEL]. For PCIe 8 Gbaud, see SRDSxLNmSSCR1[REIDL_EX_SEL_1].
5-6	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 SRDSxLNmGCR1[REIDL_ET_SEL]. For PCIe 8 Gbaud, see SRDSxLNmSSCR1[REIDL_ET_SEL_1].
7	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 SRDSxLNmGCR1[REIDL_EX_MSB]. For PCIe 8 Gbaud, see SRDSxLNmSSCR1[REIDL_EX_MSB_1].
8	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 SRDSxLNmGCR1[REIDL_ET_MSB]. For PCIe 8 Gbaud, see SRDSxLNmSSCR1[REIDL_ET_MSB_1].
9-12	—	Reserved

Table 19-26. SRDSxLNmSSCR0 Field Descriptions

13	RXEQ_BST_0	Rx Equalization Boost Default value set by RCW configuration Recommended values per protocol: PCIe: 0 (5 Gbaud) SATA: 0 (3 Gbaud) Others: 0 For PCIe 2.5 Gbaud and SATA 1.5 Gbaud, see SRDSxLNmRECR0[RXEQ_BST]. For PCIe 8 Gbaud, see SRDSxLNmSSCR1[RXEQ_BST_1].
14-15	BASE_WAND_0	Baseline Wander Control Select 00: off (8b10b data) 01: default BinBLW threshold 10: alternate BinBLW sign 11: Use lnx_(m)_rx_eq_offset[4:0] as GainBLW override Default settings set per RCW configuration. Recommended settings per protocol: 00 For PCIe 2.5 Gbaud and SATA 1.5 Gbaud, see SRDSxLNmRECR0[BASE_WAND]. For PCIe 8 Gbaud , see SRDSxLNmSSCR1[BASE_WAND_1].
16	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 SRDSxLNmRECR0[OSETOVD]. For PCIe 8 Gbaud, see SRDSxLNmSSCR1[OSETOVD6_0].

Table 19-26. SRDSxLNmSSCR0 Field Descriptions

17-18	TEQ_TYPE_0	Lane transmit equalization. 00: No TX Equalization 01: 2 Levels of TX Equalization (+1 post cursor) 10: 3 Levels of TX Equalization (+1 pre-cursor and +1 post-cursor) 11: Reserved 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 SRDSxLNmTECR0[TEQ_TYPE]. For PCIe 8 Gbaud, see SRDSxLNmSSCR1[TEQ_TYPE_0].
19	SGN_PREQ_0	Precursor sign 0: Negative Sign (close eye) 1: Positive Sign (open eye) 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 SRDSxLNmTECR0[SGN_PREQ]. For PCIe 8 Gbaud, see SRDSxLNmSSCR1[SGN_PREQ_0].
20	SGN_POST1Q_0	Post1q sign 0: Negative Sign (close eye) 1: Positive Sign (open eye) 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 SRDSxLNmTECR0[SGN_POST1Q]. For PCIe 8 Gbaud, see SRDSxLNmSSCR1[SGN_POST1Q_0].

Table 19-26. SRDSxLNmSSCR0 Field Descriptions

21:25	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 Note: Driven by protocol logic when programmed to PCIe mode. For PCIe 2.5 Gbaud and SATA 1.5 Gbaud and protocols other than PCIe or SATA, see SRDSxLNmTECR0[AMP_RED]. For PCIe 8 Gbaud, see SRDSxLNmSSCR1[AMP_RED_0].
26-31	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 Note: Driven by protocol logic when programmed to PCIe mode. For PCIe 2.5 Gbaud and SATA 1.5 Gbaud and protocols other than PCIe or SATA, see SRDSxLNmTECR0[AMP_RED]. For PCIe 8 Gbaud, see SRDSxLNmSSCR1[AMP_RED_0].

19.6.2.7.4 SRDSx Lane m Receive Equalization Control Register 0 (SRDSxLNmRECR0)

As shown in Figure 19-23, SRDSxLNmRECR0 (m=0-7) contains functional control bits for SerDes lane m

Figure 19-23. SRDSx Lane m Receive Equalization Control Register 0 (lSRDSxLNmRECR0)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	—				GK2OVD				—				GK3OVD			
Reset	000n_nnnn_0000_nnnn															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	GK2OVD_EN	GK3OVD_EN	OSETOVD_EN	—	BASE_WAND		—			OSETOVD						
Reset	nn00_0000_0011_1111															
Offset	SRDSxLN0RECR0:0x0810 SRDSxLN1RECR0:0x0850 SRDSxLN2RECR0:0x0890 SRDSxLN3RECR0:0x08D0 SRDSxLN4RECR0:0x0910 SRDSxLN5RECR0:0x0950 SRDSxLN6RECR0:0x0990 SRDSxLN7RECR0:0x09D0															

Table 19-27. SRDSxLNmRECR0 Field Descriptions

Bits	Name	Description
0-3	—	Reserved
3	RXEQ_BST	Rx Equalization Boost Default value set by RCW configuration Recommended values per protocol: XFI: 1 S-RIO: 0 (2.5/3.125 Gbaud) S-RIO: 1 (5 Gbaud) Others: 0 For PCIe 5 Gbaud and SATA 3 Gbaud, see SRDSxLNmSSCR1[RXEQ_BST_0]. For PCIe 8 Gbaud , see SRDSxLNmSSCR1[RXEQ_BST_1].
4-7	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) S-RIO: 0000 SGMII: 1111 (1.25 Gbaud) SGMII: 0000 (3.125 Gbaud) XAUI: 0000 HiGig: 0000 HiGig2: 0000 XFI: 0000 SATA: 0000 (1.5 Gbaud) Aurora: 0000 For PCIe 5 Gbaud and SATA 3 Gbaud, see SRDSxLNmSSCR1[GK2OVD_0]. For PCIe 8 Gbaud, see SRDSxLNmSSCR1[GK2OVD_1].
8-11	—	Reserved

Table 19-27. SRDSxLNmRECR0 Field Descriptions

12-15	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) S-RIO: 0000 SGMII: 1111 (1.25 Gbaud) SGMII: 0000 (3.125 Gbaud) XAUI: 0000 HiGig: 0000 HiGig2: 0000 XFI: 0000 SATA: 0000 (1.5 Gbaud) Aurora: 0000 For PCIe 5 Gbaud and SATA 3 Gbaud, see SRDSxLNmSSCR1[GK3OVD_0]. For PCIe 8 Gbaud, see SRDSxLNmSSCR1[GK3OVD_1].
16	GK2OVD_EN	Controls source of rx equalization “gaink2” setting. 0: Use rxeq adaption derived gaink2 1: Fix gaink2 according to GK2OVD Recommended settings per protocol: SGMII: 1 (1.25 Gbaud) SGMII: 0 (3.125 Gbaud) Others: 0 For PCIe 5 Gbaud and SATA 3 Gbaud, see SRDSxLNmSSCR1[GK2OVD_EN_0]. For PCIe 8 Gbaud, see SRDSxLNmSSCR1[GK2OVD_EN_1].
17	GK3OVD_EN	Controls source of rx equalization “gaink3” setting. 0: Use rxeq adaption derived gaink3 1: Fix gaink3 according to GK3OVD Recommended settings per protocol: SGMII: 1 (1.25 Gbaud) SGMII: 0 (3.125 Gbaud) Others: 0 For PCIe 5 Gbaud and SATA 3 Gbaud, see SRDSxLNmSSCR1[GK3OVD_EN_0]. For PCIe 8 Gbaud , see SRDSxLNmSSCR1[GK3OVD_EN_1].

Table 19-27. SRDSxLNmRECR0 Field Descriptions

18	OSETOVD_EN	Controls source of rx equalization “offset” setting. 0: On release of srds_reset_b, initial rx eq offset is lnx_(m)_rx_eq_offset_ovrd[6:0] and rx eq loop will begin adjustment at this value 1: rx eq offset is fixed at lnx_(m)_rx_eq_offset_ovrd[6:0] Default value set by RCW config Recommended setting per protocol: 0
19	—	Reserved
20-21	BASE_WAND	Baseline Wander Control Select 00: off (8b10b data) 01: default BinBLW threshold 10: alternate BinBLW sign 11: Use lnx_(m)_rx_eq_offset[4:0] as GainBLW override Default value set by RCW config Recommended setting per protocol: XFI: 01 Others: 00 For PCIe 5 Gbaud and SATA 3 Gbaud, see SRDSxLNmSSCR1[BASE_WAND_0]. For PCIe 8 Gbaud, see SRDSxLNmSSCR1[BASE_WAND_1].
22-24	—	Reserved
25-31	OSETOVD	Binary Decode of Lane Adaptive Equalization offset intitialization or override value. [6] = 1: Double Imposed Offset [5:0] = 6’b00 0000: + Maximum Imposed Offset [5:0] = 6’01 1111: No imposed offset [5:0] = 6’11 1111: - Maximum Imposed Offset Note: IF lnx_(m)_rx_eq_base_wand[1:0] = 11: Use lnx_(m)_rx_eq_offset[4:0] as GainBLW Override Default value set by RCW config Recommended setting per protocol: XFI: 101_1111 Others: 001_1111 For PCIe 5 Gbaud and SATA 3 Gbaud settings of OSETOVD[6], see SRDSxLNmSSCR1[OSETOVD_0]. For PCIe 8 Gbaudsettings of OSETOVD[6], see SRDSxLNmSSCR1[OSETOVD_1].

19.6.2.7.5 SRDSx Lane m Receiver Equalization Control Register 1 (SRDSxLNmRECR1)

As shown in Figure 19-24, SRDSxLNmRECR1 (m=0-7) contains control and status bits for receiver equalization on lane m.

Figure 19-24. Receive Equalization Control Register 1 (SRDSxLNmRECR1)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	—															
Reset	000n_nnnn_0000_nnnn															
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R														EQ_BSNP_DN	EQ_CSNP_DN	CDR_SNP_DN
W																
Reset	0nnn_nnnn_00nn_nnnn															
Offset	SRDSxLN0RECR1:0x814 SRDSxLN1RECR1:0x854 SRDSxLN2RECR1:0x894 SRDSxLN3RECR1:0x8D4 SRDSxLN4RECR1:0x914 SRDSxLN5RECR1:0x954 SRDSxLN6RECR1:0x994 SRDSxLN7RECR1:0x9D4;															

Table 19-28. SRDSxLNmRECR1 Field Descriptions

Bits	Name	Description
0-28	—	Reserved
29	EQ_BSNP_DN	Snapshot of RX EQ Bin Complete
30	EQ_CSNP_DN	Snapshot of RX EQ Ctrl Complete
31	CDR_SNP_DN	Snapshot of CDR Loop Complete

19.6.2.7.6 SRDSx Lane m Transmit Equalization Control Register 0 (SRDSxLNmTECR0)

As shown in Figure 19-25, SRDSxLNmTECR0 (m=0-7) contains Tx equalization control bits for SerDes lane m.

Figure 19-25. SRDSx Lane m Transmit Equalization Control Register 0 (SRDSxLNmTECR0)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	—		TEQ_TYPE		—	SGN_PREQ	RATIO_PREQ				SGN_POST1Q	RATIO_PST1Q				
Reset	00nn_0000_00nn_nnnn															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	—		ADPT_EQ						—		AMP_RED					
Reset	00nn_nnnn_00nn_nnnn															
Offset	SRDSxLN0TECR0:0x0818 SRDSxLN1TECR0:0x0858 SRDSxLN2TECR0:0x0898 SRDSxLN3TECR0:0x08D8 SRDSxLN4TECR0:0x0918 SRDSxLN5TECR0:0x0958 SRDSxLN6TECR0:0x0998 SRDSxLN7TECR0:0x09D8															

Table 19-29. SRDSxLNmTECR0 Field Descriptions

Bits	Name	Description
0-1	—	Reserved
2-3	TEQ_TYPE	Selects amount/type of Transmit Equalization 00: No TX Equalization 01: 2 Levels of TX Equalization (+1 post cursor) 10: 3 Levels of TX Equalization (+1 pre-cursor and +1 post-cursor) 11: Reserved Default value set by RCW configuration. Recommended settings per protocol: SGMII: 00 (1.25 Gbaud) SGMII: 01 (3.125 Gbaud) XFI: 01 10G-KR: 10 Others: 01 For PCIe 5 Gbaud and SATA 3 Gbaud, see SRDSxLNmSSCR1[TEQ_TYPE_0]. For PCIe 8 Gbaud, see SRDSxLNmSSCR1[TEQ_TYPE_1].
4	—	Reserved
5	SGN_PREQ	Precursor sign 0: Negative Sign(close eye) 1: Positive Sign (open eye) Default value set by RCW configuration. Recommended settings per protocol: XFI: 0 10G-KR: 1 Others: 0 For PCIe 5 Gbaud and SATA 3 Gbaud, see SRDSxLNmSSCR1[SGN_PREQ_0]. For PCIe 8 Gbaud, see SRDSxLNmSSCR1[SGN_PREQ_1].

Table 19-29. SRDSxLNmTECR0 Field Descriptions

6-9	RATIO_PREQ	Ratio of full swing transition bit to pre-cursor Default value set by RCW configuration. Recommended settings per protocol: XFI: 0000 10G-KR: 0011 Others: 0000

Table 19-29. SRDSxLNmTECR0 Field Descriptions

10	SGN_POST1Q	Post q Sign 0= Negative Sign(close eye) 1= Positive Sign (open eye) Default value set by RCW configuration. Recommended settings per protocol: SGMII: 0 (1.25 Gbaud) SGMII: 1 (3.125 Gbaud) Others: 1 For PCIe 5 Gbaud and SATA 3 Gbaud, see SRDSxLNmSSCR1[SGN_POST1Q_0]. For PCIe 8 Gbaud, see SRDSxLNmSSCR1[SGN_POST1Q_1].
11-15	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) XAUI: 0_1000 HiGig: 0_0110 HiGig2: 0_0110 XFI: 0_0011 10G-KR: 0_1100 SATA: 0_0010 (1.5 Gbaud) Aurora: 0_01000 (2.5/3.125/5 Gbaud) Others: 0_0110 For PCIe 5 Gbaud and SATA 3 Gbaud settings, see SRDSxLNmSSCR1[RATIO_PST1Q_0]. For PCIe 8 Gbaud and SATA 6 Gbaud settings, see SRDSxLNmSSCR1[RATIO_PST1Q_1]. Note: by default, this value is overridden by H/W control when in PCIe mode.
18-23	ADPT_EQ	Transmitter Adjustments for 8G/10G Default value set by RCW configuration. Recommended settings per protocol: 11_0000 Note: by default, this value is overridden by H/W control when in PCIe mode.
24-25	—	Reserved

Table 19-29. SRDSxLNmTECR0 Field Descriptions

26-31	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: 00_0000 XFI: 00_0111 10GKR: 00_0000 SATA: 01_0000 (1.5 Gbaud) Others: 00_0000 For PCIe 5 Gbaud and SATA 3 Gbaud, see SRDSxLNmSSCR1[AMP_RED_0]. For PCIe 8 Gbaud , see SRDSxLNmSSCR1[AMP_RED_1]. Note: by default, this value is overridden by H/W control when in PCIe mode.

19.6.2.7.7 SRDSx Lane m Speed Switch Control Register 1 (SRDSxLNmSSCR1)

As shown in Figure 19-26, SRDSxLNmSSCR1 (m=0-7) contains control bits for modifying the PCI Express 8 Gbaud behavior of SerDes lane m

Figure 19-26. SRDSx Lane m Speed Switch Control Register 1 (SRDSxLNmSSCR1)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	REIDL_TH_1			REIDL_EX_SEL_1		REIDL_ET_SEL_1		REIDL_EX_MSB_1	REIDL_ET_MSB_1	—				RXEQ_BST_1	BASE_WAND_1	
Reset	nnnn_nnnn_nnnn_nnnn															
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	OSETOVD6_1	TEQ_TYPE_1		SGN_PREQ_1	SGN_POST1Q_1	RATIO_PST1Q_1					AMP_RED_1					
Reset	nnnn_nnnn_nnnn_nnnn															
Offset	SRDSxLN0SSCR1: 0x81C SRDSxLN1SSCR1: 0x85C SRDSxLN2SSCR1: 0x89C SRDSxLN3SSCR1: 0x8DC															

Table 19-30. SRDSxLNmSSCR1 Field Descriptions

Bits	Name	Description
0-2	REIDL_TH_1	Receiver electrical idle detection threshold control Default set by RCW configuration. Recommended settings per protocol: PCIe: 100 (8 Gbaud) SATA: 000 (6 Gbaud) Others: 000 For PCIe 2.5 Gbaud and SATA 1.5 Gbaud and protocols other than PCIe or SATA, see SRDSxLNmGCR1[REIDL_TH]. For PCIe 5 Gbaud and SATA 3 Gbaud, see SRDSxLNmSSCR0[REIDL_TH_0].
3-4	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: PCIe: 011 (8 Gbaud) SATA: 000 (6 Gbaud) Others: 000 For PCIe 2.5 Gbaud and SATA 1.5 Gbaud and protocols other than PCIe or SATA, see SRDSxLNmGCR1[REIDL_EX_SEL]. For PCIe 5 Gbaud and SATA 3 Gbaud, see SRDSxLNmSSCR0[REIDL_EX_SEL_0].
5-6	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: PCIe: 000 (8 Gbaud) SATA: 000 (6 Gbaud) Others: 000 For PCIe 2.5 Gbaud and SATA 1.5 Gbaud and protocols other than PCIe or SATA, see SRDSxLNmGCR1[REIDL_ET_SEL]. For PCIe 5 Gbaud and SATA 3 Gbaud, see SRDSxLNmSSCR0[REIDL_ET_SEL_0].
7	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 SRDSxLNmGCR1[REIDL_EX_MSB]. For PCIe 5 Gbaud and SATA 3 Gbaud, see SRDSxLNmSSCR0[REIDL_EX_MSB_0].
8	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 SRDSxLNmGCR1[REIDL_ET_MSB]. For PCIe 5 Gbaud and SATA 3 Gbaud, see SRDSxLNmSSCR0[REIDL_ET_MSB_0].

Table 19-30. SRDSxLNmSSCR1 Field Descriptions

9-12	—	Reserved
13	RXEQ_BST_1	Rx Equalization Boost Default value set by RCW configuration Recommended values per protocol: PCIe: 1 (8 Gbaud) SATA: 1 (6 Gbaud) Others: 0 For PCIe 2.5 Gbaud and SATA 1.5 Gbaud settings, see SRDSxLNmRECR0[RXEQ_BST]. For PCIe 5 Gbaud and SATA 3 Gbaud settings, see SRDSxLNmSSCR0[RXEQ_BST_0].
14-15	BASE_WAND_1	Baseline Wander Control Select 00: off (8b10b data) 01: default BinBLW threshold 10: alternate BinBLW sign 11: Use lnx_(m)_rx_eq_offset[4:0] as GainBLW override Default value set by RCW config Recommended setting per protocol: PCIe: 01 (8 Gbaud) SATA: 00 (6 Gbaud) Others: 00 For PCIe 2.5 Gbaud and SATA 1.5 Gbaud settings, see SRDSxLNmRECR0[BASE_WAND]. For PCIe 5 Gbaud and SATA 3 Gbaud settings, see SRDSxLNmSSCR0[BASE_WAND_0].
16	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: PCIe: 1 (8 Gbaud) SATA: 0 (6 Gbaud) Others: 0 For PCIe 2.5 Gbaud and SATA 1.5 Gbaud and protocols other than PCIe or SATA, see SRDSxLNmRECR0[OSETOVD]. For PCIe 5 Gbaud and SATA 3 Gbaud, see SRDSxLNmSSCR0[OSETOVD6_0].

Table 19-30. SRDSxLNmSSCR1 Field Descriptions

17-18	TEQ_TYPE_1	Lane transmit equalization. 00: No TX Equalization 01: 2 Levels of TX Equalization (+1 post cursor) 10: 3 Levels of TX Equalization (+1 pre-cursor and +1 post-cursor) 11: Reserved Default value set by RCW configuration Recommended setting per protocol: PCIe: 10 (8 Gbaud) SATA: 01 (6 Gbaud) Others: 00 For PCIe 2.5 Gbaud and SATA 1.5 Gbaud and protocols other than PCIe or SATA, see SRDSxLNmTECR0[TEQ_TYPE]. For PCIe 5 Gbaud and SATA 3 Gbaud, see SRDSxLNmSSCR0[TEQ_TYPE_0].
19	SGN_PREQ_1	Precursor sign 0: Negative Sign (close eye) 1: Positive Sign (open eye) Default value set by RCW configuration Recommended setting per protocol: PCIe: 1 (8 Gbaud) SATA: 0 (6 Gbaud) Others: 0 For PCIe 2.5 Gbaud and SATA 1.5 Gbaud and protocols other than PCIe or SATA, see SRDSxLNmTECR0[SGN_PREQ]. For PCIe 5 Gbaud and SATA 3 Gbaud, see SRDSxLNmSSCR0[SGN_PREQ_0].
20	SGN_POST1Q_1	Post1q sign 0: Negative Sign (close eye) 1: Positive Sign (open eye) Default value set by RCW configuration Recommended setting per protocol: PCIe: 1 (8 Gbaud) SATA: 1 (6 Gbaud) Others: 0 For PCIe 2.5 Gbaud and SATA 1.5 Gbaud and protocols other than PCIe or SATA, see SRDSxLNmTECR0[SGN_POST1Q]. For PCIe 5 Gbaud and SATA 3 Gbaud, see SRDSxLNmSSCR0[SGN_POST1Q_0].

Table 19-30. SRDSxLNmSSCR1 Field Descriptions

21:25	RATIO_PST1Q_1	Ratio of full swing transition bit to first post-cursor. Default value set by RCW configuration Recommended setting per protocol: PCIe: 0_1100 (8 Gbaud) SATA: 0_0010 (6 Gbaud) Others: 0_0000 Note: Driven by protocol logic when programmed to PCIe mode. For SATA 1.5 Gbaud and protocols other than PCIe or SATA, see SRDSxLNmTECR0[AMP_RED]. For SATA 3 Gbaud, see SRDSxLNmSSCR0[AMP_RED_0].
26-31	AMP_RED_1	Overall TX Amplitude Reduction Default value set by RCW configuration Recommended setting per protocol: PCIe: 00_0000 (8 Gbaud) SATA: 00_0000 (6 Gbaud) Others: 00_0000 Note: Driven by protocol logic when programmed to PCIe mode. For PCIe 2.5 Gbaud and SATA 1.5 Gbaud and protocols other than PCIe or SATA, see SRDSxLNmTECR0[AMP_RED]. For PCIe 5 Gbaud and SATA 3 Gbaud, see SRDSxLNmSSCR0[AMP_RED_0].

19.6.2.7.8 Transition Tracking Loop Control Register 0 (SRDSxLNmTTLCR0)

As shown in Figure 19-27, SRDSxLNmTTLCR0 (m=0-7)contains control bits for the Transition Tracking Loop (TTL) on SerDes lane m.

Figure 19-27. SRDSx Lane m Transition Tracking Loop Control Register (SRDSxLNmTTLCR0)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	—		FLT_SEL						—							
Reset	00nn_nnnn_0000_0000															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	—															
Reset	1n00_0000_0000_0000															
Offset	SRDSxLN0TTLCR0: 0x820 SRDSxLN1TTLCR0: 0x860 SRDSxLN2TTLCR0: 0x8A0 SRDSxLN3TTLCR0: 0x8E0 SRDSxLN4TTLCR0: 0x920 SRDSxLN5TTLCR0: 0x960 SRDSxLN6TTLCR0: 0x9A0 SRDSxLN7TTLCR0: 0x9E0															

Table 19-31. SRDSxLNmTTLCR0 Field Descriptions

Bits	Name	Description
0-1	—	Reserved
2-7	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) Others: 00_0000
8-31	—	Reserved

19.6.2.7.9 SRDS x Lane m Test Control/Status Register 3 (SRDSxLNmTCSR3)

SRDSxLNmTCSR3 (m=0-7) contains test control and status bits, as shown in Figure 19-28.

Figure 19-28. SerDes x, Lane m Test/Control Status Register 3 (SRDSxLNmTCSR3)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	—		LPBK_EN		CDR_LCK	—										
W																
Reset	0000_nn10_0000_00nn															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	—															
Reset	nnnn_nnnn_0000_0000															
Offset	SRDSxLN0TCSR3: 0x83C SRDSxLN1TCSR3: 0x87C SRDSxLN2TCSR3: 0x8BC SRDSxLN3TCSR3: 0x8FC SRDSxLN4TCSR3: 0x93C SRDSxLN5TCSR3: 0x97C SRDSxLN6TCSR3: 0x9BC SRDSxLN7TCSR3: 0x9FC															

Table 19-32. SRDSxLNmTCSR3 Field Descriptions

Bits	Name	Description
0-1	—	Reserved
2-3	LPBK_EN	Loopback data from TX to RX 00: Application Mode 01: Loopback Mode All others reserved Note: Loopback using PCI-Express requires Root Complex configuration. PCI Express End point loopback is not supported.
4	CDR_LCK	When asserted, CDR loop has acquired a valid Rx clock
5-31	—	Reserved

19.7 MDIO Registers

The SerDes module also contains MDIO slave ports containing registers for each XFI, SGMII and PCS module. Each PCS MDIO is driven by the corresponding MAC's MDIO. Table 19-33 shows the mapping of PCS MDIO Slave to MAC MDIO Master.

Note: these registers only apply to SerDes 1.

Table 19-33. SerDes MDIO Port Mapping

SerDes *n*	MAC MDIO Master	PCS MDIO Slave	PCS MDIO Device Port Address	Supported DEVAD or PHYAD values	PCS MDIO Register Offsets Per DEVAD/PHYAD
1, 2	#3	SGMIIa	SRDSSGMIIACR1[MDEV_PORT]	N/A (SGMII) 0x07, 0x1D, 0x1E (1000Base-KX)	0x00-0x1F (SGMII) 0x0000-0xFFFF (1000Base-KX)
1	#4	SGMIIb	SRDSSGMIIBCR1[MDEV_PORT]	N/A (SGMII) 0x07, 0x1D, 0x1E (1000Base-KX)	0x00-0x1F (SGMII) 0x0000-0xFFFF (1000Base-KX)

Table 19-33. SerDes MDIO Port Mapping

SerDes *n*	MAC MDIO Master	PCS MDIO Slave	PCS MDIO Device Port Address	Supported DEVAD or PHYAD values	PCS MDIO Register Offsets Per DEVAD/PHYAD
1	#5	SGMIIc	SRDSSGMIICCR1[MDEV_PORT]	N/A (SGMII) 0x07, 0x1D, 0x1E (1000Base-KX)	0x00-0x1F (SGMII) 0x0000-0xFFFF (1000Base-KX)
1	#6	SGMIId	SRDSSGMIIDCR1[MDEV_PORT]	N/A (SGMII) 0x07, 0x1D, 0x1E (1000Base-KX)	0x00-0x1F (SGMII) 0x0000-0xFFFF (1000Base-KX)

19.7.1 Detailed MDIO Register Map

Table 19-34 lists the Clause 45 MDIO Registers in the XFI PCS and SGMII PCS for 1000Base-KX mode. Note that the XFI PCSs and MDIO are also used for 10GBase-KR.

Table 19-35 lists the Clause 22 MDIO Registers in the SGMII PCSs.

Each MDIO register is 16 bits. Addresses not listed are reserved. Read accesses to reserved addresses return 0x0000. Note that register offsets are 16-bit offsets, not byte offsets.

Table 19-34. Clause 45 SerDes MDIO Register Map

Port Address	Device Address	Register Address	Register	Access	Reset	Section/Page
XFI MDIO (Clause 45)						

Table 19-34. Clause 45 SerDes MDIO Register Map (continued)

Port Address	Device Address	Register Address	Register	Access	Reset	Section/Page
MDEV_PORT	0x01	XFI PMA/PMD Registers				
		0x0000	XFI PMD Control 1	R/W	0x0000	19.7.2.1.1/19-67
		0x0001-0x0095	Reserved	—	—	
		0x0096	XFI 10GBASE-KR PMD Control Register	R/W	0x0000	/19-68
		0x0097	XFI 10GBASE-KR PMD Status Register	R/W	0x0000	19.7.2.1.2/19-68
		0x0098	XFI 10GBASE-KR LP Coefficient Update Register	R/W	0x0000	19.7.2.1.3/19-69
		0x0099	XFI 10GBASE-KR LP Status Report Register	R	0x0000	19.7.2.1.4/19-70
		0x009A	XFI 10GBASE-KR LD Coefficient Update Register	R	0x0000	19.7.2.1.5/19-71
		0x009B	XFI 10GBASE-KR LD Status Report Register	R/W	0x0000	19.7.2.1.6/19-72
		0x009C-0x00A9	Reserved	—	—	
		0x00AA	XFI 10GBASE-R FEC Ability	R	0x0000	19.7.2.1.7/19-73
		0x00AB-0x8000	Reserved	—	—	—
		0x8001	XFI Number or PRBS Sequence Errors Lower	R	0x0000	19.7.2.1.8/19-74
		0x8002	XFI Number of PRBS Sequence Errors Upper	R	0x0000	19.7.2.1.9/19-75
		0x8003-0xFFFF	Reserved	—	—	—

Table 19-34. Clause 45 SerDes MDIO Register Map (continued)

Port Address	Device Address	Register Address	Register	Access	Reset	Section/Page
MDEV_PORT	0x03	XFI PCS Registers				
		0x0000	XFI PCS Control 1	RW	0x2000	19.7.2.2.1/19-75
		0x0001	XFI PCS Status 1	R	0x0002	19.7.2.2.2/19-76
		0x0002	XFI PCS Device Identifier Upper	R	0x0083	19.7.2.2.3/19-77
		0x0003	XFI PCS Device Identifier Lower	R	0xE400	19.7.2.2.4/19-78
		0x0004	XFI PCS Speed Ability	R	0x0001	19.7.2.2.5/19-79
		0x0005	XFI PCS Devices In Package 0	R	0x008A	19.7.2.2.6/19-79
		0x0006	XFI PCS Devices in Package 1	R	0x0000	19.7.2.2.7/19-80
		0x0007	XFI 10G PCS Control 2	R	0x000B	19.7.2.2.8/19-81
		0x0008	XFI 10G PCS Status 2	R	0x8001	19.7.2.2.9/19-82
		0x0009-0x000D	Reserved	—	—	—
		0x000E	XFI PCS Package Identifier Upper	R	0x0083	19.7.2.2.10/19-83
		0x000F	XFI PCS Package Identifier Lower	R	0xE400	19.7.2.2.11/19-83
		0x0010-0x001F	Reserved	—	—	—
		0x0020	XFI 10GBASE-R PCS Status 1	R	0x0000	19.7.2.2.12/19-84
		0x0021	XFI 10GBASE-R PCS Status 2	R	0x0000	19.7.2.2.13/19-85
		0x0022	XFI 10GBASE-R PCS Test Pattern Seed A 0	R/W	0x0000	19.7.2.2.14/19-86
		0x0023	XFI 10GBASE-R PCS Test Pattern Seed A 1	R/W	0x0000	19.7.2.2.15/19-86
		0x0024	XFI 10GBASE-R PCS Test Pattern Seed A 2	R/W	0x0000	19.7.2.2.16/19-87
		0x0025	XFI 10GBASE-R PCS Test Pattern Seed A 3	R/W	0x0000	19.7.2.2.17/19-87
		0x0026	XFI 10GBASE-R PCS Test Pattern Seed B 0	R/W	0x0000	19.7.2.2.18/19-88
		0x0027	XFI 10GBASE-R PCS Test Pattern Seed B 1	R/W	0x0000	19.7.2.2.19/19-88
		0x0028	XFI 10GBASE-R PCS Test Pattern Seed B 2	R/W	0x0000	19.7.2.2.20/19-89
		0x0029	XFI 10GBASE-R PCS Test Pattern Seed B 3	R/W	0x0000	19.7.2.2.21/19-89
		0x002A	XFI 10GBASE-R PCS Test Pattern Control	R/W	0x0000	19.7.2.2.22/19-90
		0x002B	XFI 10GBASE-R PCS Test Pattern Error Counter	R	0x0000	19.7.2.2.23/19-91
		0x002C-0x7FFF	Reserved	—	—	—
		0x8000	Vendor Specific PCS Status	R	0x0000	19.7.2.2.24/19-91
		0x8001-0xFFFF	Reserved	—	—	—

Table 19-34. Clause 45 SerDes MDIO Register Map (continued)

Port Address	Device Address	Register Address	Register	Access	Reset	Section/Page
MDEV_ PORT	0x07	XFI Auto-Negotiation Registers				
		0x0000	XFI AN Control	R/W	0x0000	19.7.2.3.1/19-92
		0x0001	XFI AN Status	R	0x0008	19.7.2.3.2/19-93
		0x0002	XFI AN Device Identifier Upper	R	0x0083	19.7.2.3.3/19-94
		0x0003	XFI AN Device Identifier Lower	R	0xE400	19.7.2.3.4/19-95
		0x0004	Reserved	—	—	—
		0x0005	XFI AN Devices In Package 0	R	0x008A	19.7.2.3.5/19-95
		0x0006	XFI AN Devices In Package 1	R	0x0000	19.7.2.3.6/19-96
		0x0007-0x000D	Reserved	—	—	—
		0x000E	XFI AN Package Identifier Upper	R	0x0083	19.7.2.3.7/19-97
		0x000F	XFI AN Package Identifier Lower	R	0xE400	19.7.2.3.8/19-98
		0x0010	XFI AN Advertisement 0	R/W	0x0001	19.7.2.3.9/19-98
		0x0011	XFI AN Advertisement 1	R/W	0x001F	19.7.2.3.10/19-99
		0x0012	XFI AN Advertisement 2	R/W	0x0000	19.7.2.3.11/19-100
		0x0013	XFI AN LP Base Page Ability 0	R	0x0000	19.7.2.3.11/19-100
		0x0014	XFI AN LP Base Page Ability 1	R	0x0000	19.7.2.3.12/19-100
		0x0015	XFI AN LP Base Page Ability 2	R	0x0000	19.7.2.3.14/19-102
		0x0016	XFI AN Extended Next Page Transmit 0	R/W	0x2001	19.7.2.3.15/19-103
		0x0017	XFI AN Extended Next Page Transmit 1	R/W	0x0000	19.7.2.3.16/19-104
		0x0018	XFI AN Extended Next Page Transmit 2	R/W	0x0000	19.7.2.3.17/19-104
		0x0019	XFI AN LP Extended Next Page Ability 0	R	0x0000	19.7.2.3.18/19-105
		0x001A	XFI AN LP Extended Next Page Ability 1	R	0x0000	19.7.2.3.19/19-106
		0x001B	XFI AN LP Extended Next Page Ability 2	R	0x0000	19.7.2.3.20/19-106
		0x001C-0x002F	Reserved	—	—	—
		0x0030	XFI Backplane Ethernet Status	R	0x0000	19.7.2.3.21/19-107
		0x0031-0x7FFF	Reserved	—	—	—
		0x8000	XFI Millisecond Counter	R/W	0x9896	19.7.2.3.22/19-108
		0x8001-0xFFFF	Reserved	—	—	—

Table 19-34. Clause 45 SerDes MDIO Register Map (continued)

Port Address	Device Address	Register Address	Register	Access	Reset	Section/Page
MDEV_PORT	0x1E	XFI Vendor-Specific 1 Registers				
		0x0000	XFI Revision	R	0x1010	19.7.2.4.1/19-108
		0x0001	XFI Scratch	R/W	0x0000	19.7.2.4.2/19-109
		0x0002	XFI PCS Interrupt Event	R	0x0000	19.7.2.4.3/19-110
		0x0003	XFI PCS Interrupt Mask	R/W	0x0000	19.7.2.4.4/19-111
		0x0004	XFI Auto-Negotiation Interrupt Event	RC	0x0000	19.7.2.4.5/19-113
		0x0005	XFI Auto-Negotiation Interrupt Mask	R/W	0x0000	19.7.2.4.6/19-114
		0x0006	XFI Link Training Interrupt Event	RC	0x0000	19.7.2.4.7/19-115
		0x0007	XFI Link Training Interrupt Mask	R/W	0x0000	19.7.2.4.8/19-116
		0x0008-0xFFFF	Reserved	—	—	—
1000Base-KX MDIO						
MDEV_PORT	0x03	PCS Registers (Including vendor-specific alias of Clause 22 SGMII registers)				
		0x0000	KX PCS Control	R/W	0x1140	19.7.2.6.1/19-130
		0x0001	KX PCS Status	R/W	0x0009	19.7.2.6.2/19-131
		0x0002	KX PCS Device Identifier Upper	R	0x0083	19.7.2.6.3/19-131
		0x0003	KX PCS Device Identifier Lower	R	0xE400	19.7.2.6.4/19-132
		0x0004	Reserved	—	—	—
		0x0005	KX PCS Devices In Package 0	R	0x0000	19.7.2.6.5/19-133
		0x0006	KX PCS Devices In Package 1	R	0x0088	19.7.2.6.6/19-134
		0x0007-0x000D	Reserved	—	—	—
		0x000E	KX PCS Package Identifier Upper	R	0x0083	19.7.2.6.7/19-134
		0x000F	KX PCS Package Identifier Lower	R	0xE400	19.7.2.6.8/19-135
		0x0010-0x7FFF	Reserved	—	—	—
		0x8000	SGMII Control	R/W	0x1140	19.7.2.5.1/19-117
		0x8001	SGMII Status	R	0x0009	19.7.2.5.2/19-118
		0x8002	SGMII PHY Identifier Upper	R	0x0083	19.7.2.5.3/19-119
		0x8003	SGMII PHY Identifier Lower	R	0xE400	19.7.2.5.4/19-120
		0x8004	SGMII Device Ability	R/W	0x01A0	19.7.2.5.6/19-121
		0x8005	SGMII Partner Ability	R	0x0000	19.7.2.5.7/19-122
		0x8006	SGMII AN Expansion	RC	0x0004	19.7.2.5.9/19-124
		0x8007	SGMII Next Page Transmit	R/W	0x0000	19.7.2.5.10/19-125
		0x8008	SGMII LP Next Page Receive	R	0x0000	19.7.2.5.11/19-125

Table 19-34. Clause 45 SerDes MDIO Register Map (continued)

Port Address	Device Address	Register Address	Register	Access	Reset	Section/Page
		0x8009-0x800E	Reserved	—	—	—
		0x800F	SGMII Extended Status	R	0x0000	19.7.2.5.12/19-126
		0x8010	SGMII Scratch	R/W	0x0000	19.7.2.5.13/19-127
		0x8011	SGMII Design Revision	R	0x0001	19.7.2.5.14/19-127
		0x8012	SGMII Link Timer Lower	R/W	0x12D0	19.7.2.5.15/19-128
		0x8013	SGMII Link Timer Upper	R/W	0x0013	19.7.2.5.16/19-128
		0x8014	SGMII IF Mode	R/W	0x0000	19.7.2.5.17/19-129
		0x8016-0xFFFF	Reserved	—	—	—

Table 19-34. Clause 45 SerDes MDIO Register Map (continued)

Port Address	Device Address	Register Address	Register	Access	Reset	Section/Page
MDEV_ PORT	0x07	KX Auto-Negotiation Registers				
		0x0000	KX AN Control	R/W	0x0000	19.7.2.7.1/19-136
		0x0001	KX AN Status	Special	0x0040	19.7.2.7.2/19-137
		0x0002	KX AN Device Identifier Upper	RO	0x0083	19.7.2.7.3/19-138
		0x0003	KX AN Device Identifier Lower	RO	0xE400	19.7.2.7.4/19-139
		0x0004	Reserved	—	—	—
		0x0005	KX AN Devices in Package 0	RO	0x0088	19.7.2.7.5/19-139
		0x0006	KX AN Devices in Package 1	RO	0x0000	19.7.2.7.6/19-140
		0x0007-0x000D	Reserved	—	—	—
		0x000E	KX AN Package Identifier Upper	RO	0x0083	19.7.2.7.7/19-141
		0x000F	KX AN Package Identifier Lower	RO	0xE400	19.7.2.7.8/19-142
		0x0010	KX AN Advertisement 0	RW	0x0C01	19.7.2.7.9/19-142
		0x0011	KX AN Advertisement 1	RW	0x003F	19.7.2.7.10/19-143
		0x0012	KX AN Advertisement 2	RW	0x0000	19.7.2.7.11/19-144
		0x0013	KX AN LP Base Page Ability 0	RO	0x0000	19.7.2.7.12/19-145
		0x0014	KX AN LP Base Page Ability 1	RO	0x0000	19.7.2.7.13/19-146
		0x0015	KX AN LP Base Page Ability 2	RO	0x0000	19.7.2.7.14/19-147
		0x0016	KX AN XNP Transmit 0	RO	0x2001	19.7.2.7.15/19-147
		0x0017	KX AN XNP Transmit 1	RO	0x0000	19.7.2.7.16/19-148
		0x0018	KX AN XNP Transmit 2	RO	0x0000	19.7.2.7.17/19-149
		0x0019	KX AN LP XNP Ability 0	RO	0x0000	19.7.2.7.18/19-149
		0x001A	KX AN LP XNP Ability 1	RO	0x0000	19.7.2.7.19/19-150
		0x001B	KX AN LP XNP Ability 2	RO	0x0000	19.7.2.7.20/19-151
		0x001C-0x002F	Reserved	—	—	—
		0x0030	KX Backplane Ethernet Status	RO	0x0001	19.7.2.7.21/19-151
		0x0031-0x07FF	Reserved	—	—	—
		0x8000	KX Millisecond Count	RW	0x9896	19.7.2.7.22/19-152
		0x8001-0xFFFF	Reserved	—	—	—

Table 19-34. Clause 45 SerDes MDIO Register Map (continued)

Port Address	Device Address	Register Address	Register	Access	Reset	Section/Page
MDEV_PORT	0x1D	Vendor Specific 1 Registers				
		0x0000	KX Revision	RO	0x0114	19.7.2.8.1/19-153
		0x0001	KX Scratch	R/W	0x0000	19.7.2.8.2/19-153
		0x0002	KX PCS Interrupt Event	Special	0x0000	19.7.2.8.3/19-154
		0x0003	KX PCS Interrupt Mask	R/W	0x0000	19.7.2.8.4/19-154
		0x0004	KX AN Interrupt Event	Special	0x0000	19.7.2.8.5/19-155
		0x0005	KX AN Interrupt Mask	R/W	0x0000	19.7.2.8.6/19-156
		0x0006-0xFFFF	Reserved	R/W	0x0000	—

Table 19-35. Clause 22 SerDes MDIO Register Map

PHY Address	Register Address	Register	Access	Reset	Section/Page
SGMII MDIO (Clause 22)					
MDEV_PORT	0x00	SGMII Control	R/W	0x1140	19.7.2.5.1/19-117
	0x01	SGMII Status	R	0x0009	19.7.2.5.1/19-117
	0x02	SGMII PHY Identifier Upper	R	0x0083	19.7.2.5.2/19-118
	0x03	SGMII PHY Identifier Lower	R	0xE400	19.7.2.5.3/19-119
	0x04	SGMII Device Ability	R/W	0x01A0	19.7.2.5.4/19-120
	0x05	SGMII Partner Ability	R	0x0000	19.7.2.5.6/19-121
	0x06	SGMII AN Expansion	RC	0x0004	19.7.2.5.7/19-122
	0x07	SGMII Next Page Transmit	R/W	0x0000	19.7.2.5.9/19-124
	0x08	SGMII LP Next Page Receive	R	0x0000	19.7.2.5.10/19-125
	0x09-0x0E	Reserved	—	—	19.7.2.5.11/19-125
	0x0F	SGMII Extended Status	R	0x0000	—
	0x10	SGMII Scratch	R/W	0x0000	19.7.2.5.12/19-126
	0x11	SGMII Design Revision	R	0x0001	19.7.2.5.13/19-127
	0x12	SGMII Link Timer Lower	R/W	0x12D0	19.7.2.5.14/19-127
	0x13	SGMII Link Timer Upper	R/W	0x0013	19.7.2.5.15/19-128
	0x14	SGMII IF Mode	R/W	0x0000	19.7.2.5.16/19-128
	0x15-0x1F	Reserved	—	—	—

19.7.2 MDIO Register Descriptions

In the following register descriptions, special access is one of:

- SC- Self-clearing
- W1C - Write-one-to-clear
- RC - Read-with-clear

If the ‘special access’ field is blank, the register field is standard read-only, write-only, or read-write, as shown in the R and W rows.

19.7.2.1 XFI PMA/PMD Registers

The PMA/PMD MDIO space includes the registers required to control link training and advertise FEC ability.

19.7.2.1.1 XFI PMD Control 1

XFI PMD Control 1 register contains global controls for the XFI PMD module, as shown in Figure 19-29

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PMD_RESET	—														
Special Access	SC															
Reset	0000_0000_0000_0000															
Offset	0x0000															

Figure 19-29. XFI PMD Control 1 Register

Table 19-36. XFI PMD Control 1 Register Field Descriptions

Bits	Name	Description
15-1	—	Reserved
0	PMD_RESET	PMD Reset Self-Clearing Reset Bit. When set to 1, resets the 10GBase-KR Link Training modules. The reset command clears all bits in the 10GBase-KR PMD control register

XFI 10GBASE-KR PMD Control RegisterXFI 10GBASE-KR PMD Control Register contains global controls for the 10GBASE-KR, as shown in Figure 19-30

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	—														TRAIN_EN	RESTART_TRAIN
W																
Special Access																SC
Reset	0000_0000_0000_0000															
Offset	0x0096															

Figure 19-30. XFI 10GBASE-KR PMD Control Register

Table 19-37. XFI 10GBASE-KR PMD Control Register Field Descriptions

Bits	Name	Description
15-1	—	Reserved
1	TRAIN_EN	Training Enable 0: 10GBase-KR startup protocol disabled 1: 10GBase-KR startup protocol enabled Note: To safely disable link training, the Restart bit should be written with 1 with enable=0 to force exit of the state machine from whatever state is would be in. (see IEEE 802.3 Fig 72-5 Training State Diagram).
0	RESTART_TRAIN	Restart Training Self Clearing Bit. When set to 1, resets the 10GBase-KR startup protocol and begins or ends its operation (depending on TRAIN_EN).

19.7.2.1.2 XFI 10GBASE-KR PMD Status Register

XFI 10GBASE-KR PMD Control Register contains status for the 10GBASE-KR, as shown in Figure 19-31

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	—												TRAIN_FAIL	SUP_STAT	FRAME_LOCK	RX_STAT
W																
Special Access																
Reset	0000_0000_0000_0000															
Offset	0x0097															

Figure 19-31. XFI 10GBASE-KR PMD Status Register

Table 19-38. XFI 10GBASE-KR PMD Status Register Field Descriptions

Bits	Name	Description
15-4	—	Reserved
3	TRAIN_FAIL	Training Failure: 0: Training failure has not been detected 1: Training failure has been detected Note: when training failure occurs, the link is kept in an inoperable state (PCS is not allowed to lock). The application then must write the 10GBase-KR PMD Control register with the restart bit being set and the enable bit being cleared to exit this condition and allow the link to come up without link training.
2	SUP_STAT	Start-up Protocol Status 0: Startup protocol is completed (or not started) 1: Startup protocol is in progress
1	FRAME_LOCK	Frame Lock 0: Training frame delineation not detected 1: Training frame delineation detected
0	RX_STAT	Receiver Status 0: Receiver training is proceeding (or not started) 1: Receiver is trained and is ready to receive data Note: this bit is writeable, which differs from the corresponding notion in IEEE 802.3 Clause 45.2.1.77 for MMD register 1.151) as the application has to decide when the local device is trained. Writing this bit sets the mr_trained variable allowing the training statemachine to proceed (see IEEE 802.3 Clause 72.6.10.3.1)

19.7.2.1.3 XFI 10GBASE-KR LP Coefficient Update Register

XFI 10GBASE-KR LP Coefficient Update Register contains controls for the 10GBASE-KR Link Partner Coefficient Update, as shown in Figure 19-32. The fields in this register are read-only if XFI 10GBASE-KR PMD Control Register[TRAIN_EN]=1, but they are writeable if TRAIN_EN=0.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	—		PRESET	INIT	—						COP1_UPD		COZ_UPD		COM1_UPD	
Special Access			*	*												
Reset	0000_0000_0000_0000															
Offset	0x0098															

Figure 19-32. XFI 10GBASE-KR LP Coefficient Update Register

Table 19-39. XFI 10GBASE-KR LP Coefficient Update Register Field Descriptions

Bits	Name	Description
15-14	—	Reserved
13	PRESET	Preset the transmitter coefficients 0: Normal operation 1: Preset the transmitter coefficients
12	INIT	Initialize the transmitter coefficients 0: Normal operation 1: Initialize the transmitter coefficients
11-6	—	Reserved
5-4	COP1_UPD	Coefficient (+1) update: 11: reserved 10: decrement 01: increment 00: hold
3-2	COZ_UPD	Coefficient (0) update: 11: reserved 10: decrement 01: increment 00: hold
1-0	COM1_UPD	Coefficient (-1) update: 11: reserved 10: decrement 01: increment 00: hold

19.7.2.1.4 XFI 10GBASE-KR LP Coefficient Status Report Register

XFI 10GBASE-KR LP Coefficient Status Report Register contains the 10GBASE-KR Link Partner Coefficient reported status, as shown in Figure 19-33

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RX_RDY	—									COP1_STAT		COZ_STAT		COM1_STAT	
W																
Special Access																
Reset	0000_0000_0000_0000															
Offset	0x0099															

Figure 19-33. XFI 10GBASE-KR LP Coefficient Status Report Register

Table 19-40. XFI 10GBASE-KR LP Coefficient Status Report Register Field Descriptions

Bits	Name	Description
15	RX_RDY	Receiver Ready 0: The LP receiver is requesting that training continue 1: The LP receiver has determined that training is complete and is prepared to receive data.
14-6	—	Reserved
5-4	COP1_STAT	Coefficient (+1) status: 11: maximum 01: minimum 10: updated 00: not updated
3-2	COZ_STAT	Coefficient (0) status: 11: maximum 01: minimum 10: updated 00: not updated
1-0	COM1_STAT	Coefficient (-1) status: 11: maximum 01: minimum 10: updated 00: not updated

19.7.2.1.5 XFI 10GBASE-KR LD Coefficient Update Register

XFI 10GBASE-KR LD Coefficient Update Register contains controls for the 10GBASE-KR Link Device Coefficient Update, as shown in Figure 19-34

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	—		PRESET	INIT	—						COP1_UPD		COZ_UPD		COM1_UPD	
Special Access																
Reset	0000_0000_0000_0000															
Offset	0x009A															

Figure 19-34. XFI 10GBASE-KR LD Coefficient Update Register

Table 19-41. XFI 10GBASE-KR LD Coefficient Update Register Field Descriptions

Bits	Name	Description
15-14	—	Reserved
13	PRESET	Preset the transmitter coefficients 0: Normal operation 1: Preset the transmitter coefficients
12	INIT	Initialize the transmitter coefficients 0: Normal operation 1: Initialize the transmitter coefficients
11-6	—	Reserved
5-4	COP1_UPD	Coefficient (+1) update: 11: reserved 10: decrement 01: increment 00: hold
3-2	COZ_UPD	Coefficient (0) update: 11: reserved 10: decrement 01: increment 00: hold
1-0	COM1_UPD	Coefficient (-1) update: 11: reserved 10: decrement 01: increment 00: hold

19.7.2.1.6 XFI 10GBASE-KR LD Coefficient Status Report Register

XFI 10GBASE-KR LD Coefficient Status Report Register contains the 10GBASE-KR Link Device Coefficient reported status, as shown in Figure 19-35

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RX_RDY	—									COP1_STAT		COZ_STAT		COM1_STAT	
Special Access																
Reset	0000_0000_0000_0000															
Offset	0x009B															

Figure 19-35. XFI 10GBASE-KR LD Coefficient Status Report Register

Table 19-42. XFI 10GBASE-KR LD Coefficient Status Report Register Field Descriptions

Bits	Name	Description
15	RX_RDY	Receiver Ready 0: The LD receiver is requesting that training continue 1: The LD receiver has determined that training is complete and is prepared to receive data.
14-6	—	Reserved
5-4	COP1_STAT	Coefficient (+1) status: 11: maximum 01: minimum 10: updated 00: not updated
3-2	COZ_STAT	Coefficient (0) status: 11: maximum 01: minimum 10: updated 00: not updated
1-0	COM1_STAT	Coefficient (-1) status: 11: maximum 01: minimum 10: updated 00: not updated

19.7.2.1.7 XFI 10GBASE-R FEC Ability Register

XFI 10GBASE-R FEC Ability Register contains the status bit showing the PCS FEC capability, as shown in Figure 19-36. Note that this device does not support FEC, so the FEC ability is a constant 0.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	—														FEC_ERR_ABIL	FEC_ABIL
W																
Special Access																
Reset	0000_0000_0000_0000															
Offset	0x00AA															

Figure 19-36. XFI 10GBase-R FEC Ability Register

Table 19-43. XFI 10GBase-R FEC Ability Register Field Descriptions

Bits	Name	Description
15-2	—	Reserved
1	FEC_ERR_ABIL	10GBASE-R FEC Error Indication Ability Set to 0 since this PCS does not implement the FEC functions
0	FEC_ABIL	10GBase-R FEC Ability Set to 0 to indicate this PCS does not implement the FEC functions

19.7.2.1.8 XFI Number or PRBS Sequence Errors Lower Register

The XFI Number or PRBS Sequence Errors Lower Register contains the lower half of the 32-bit counter for tracking number of link training frame PRBS Sequence Errors, as shown in Figure 19-37

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	NPRBS_ERR_CNT_L															
W																
Special Access	RC	RC	RC	RC	RC	RC	RC	RC	RC	RC	RC	RC	RC	RC	RC	RC
Reset	0000_0000_0000_0000															
Offset	0x8001															

Figure 19-37. XFI Number or PRBS Sequence Errors Lower Register

Table 19-44. XFI Number or PRBS Sequence Errors Lower Register Field Descriptions

Bits	Name	Description
15-0	NPRBS_ERR_CNT_L	Number of Link Training Frame PRBS Sequence Error, lower 16 Bits. Reset to 0 when a read is performed on the counter upper register, the counter does not roll over to 0 when the value 0xFFFFFFFF is reached.

19.7.2.1.9 XFI Number or PRBS Sequence Errors Upper Register

The XFI Number or PRBS Sequence Errors Upper Register contains the upper half of the 32-bit counter for tracking number of link training frame PRBS Sequence Errors, as shown in Figure 19-38

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	NPRBS_ERR_CNT_U															
W																
Special Access	RC	RC	RC	RC	RC	RC	RC	RC	RC	RC	RC	RC	RC	RC	RC	RC
Reset	0000_0000_0000_0000															
Offset	0x8002															

Figure 19-38. XFI Number or PRBS Sequence Errors Upper Register

Table 19-45. XFI Number or PRBS Sequence Errors Upper Register Field Descriptions

Bits	Name	Description
15-0	NPRBS_ERR_CNT_L	Number of Link Training Frame PRBS Sequence Error, upper 16 Bits. Reset to 0 when a read is performed on the counter upper register, the counter does not roll over to 0 when the value 0xFFFFFFFF is reached.

19.7.2.2 XFI PCS Registers

The XFI PCS MDIO space includes the registers required to control the XFI PCS functions.

19.7.2.2.1 XFI PCS Control 1 Register

The PCS Control 1 Register contains controls for the XFI PCS, as shown in Figure 19-39

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RESET	LPBK	SPEED_SEL0	—	LPWR	—				SPEED_SEL1	SPEED_SEL				—	
W																
Special Access	SC															
Reset	0010_0000_0100_0000															
Offset	0x0000															

Figure 19-39. XFI PCS Control 1 Register

Table 19-46. XFI PCS Control 1 Register Field Descriptions

Bits	Name	Description
15	RESET	Reset 0: PCS is not in reset 1: PCS transmit and receive functions are being reset Self Clearing bit.
14	LPBK	Loopback Enable 0: Normal function 1: XGMII Tx data is returned to XGMII Rx
13	SPEED_SEL1	Speed Selection 13 6 1 1: Bits 5:2 select speed Read only
12	—	Reserved
11	LPWR	Low Power Operation 0: Normal function 1: Low power operation: PCS is in reset state and most functions are disabled
10-7	—	Reserved
6	SPEED_SEL0	See SPEED_SEL1 Read only
5-2	SPEED_SEL	Speed Selection 0000: 10 Gbps Read only
1-0	—	Reserved

19.7.2.2.2 XFI PCS Status 1 Register

The XFI PCS Status 1 Register contains status bits for the XFI PCS, as shown in Figure 19-40

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	—				TX_LPI	RX_LPI	TX_LPI_ACTIVE	RX_LPI_ACTIVE	FAULT	LPI_CLK_STP_CAP	—			PCS_RX_LNK_STAT	LPWR_ABIL	—
W																
Special Access					LH	LH								LL		
Reset	0000_0000_0000_0010															
Offset	0x0001															

Figure 19-40. XFI PCS Status 1 Register

Table 19-47. XFI PCS Status 1 Register Field Descriptions

Bits	Name	Description
15-12	—	Reserved
11	TX_LPI	Transmit LPI 0: normal operation. 1::transmit is or was in LPI (EEE) Once set, stays set until register read Note: for this device Tx LPI state does not
10	RX_LPI	Receive LPI 0: normal operation. 1: receive is or was in LPI state (EEE) Once set, stays set until register read
9	TX_LPI_ACTIVE	Transmit LPI Active 0: normal operation 1: transmit is currently in LPI state (EEE)
8	RX_LPI_ACTIVE	Receive LPI Active 0: normal operation 1: receive is currently in LPI state (EEE).
7	FAULT	Fault 0: no fault condition detected 1: fault condition detected
6	LPI_CLK_STP_CAP	Clock Stop Capable: 0: MAC is not capable of stopping the clock during LPI
5-3	—	Reserved
2	PCS_RX_LNK_STAT	PCS Receive Link Status 0: indicates that the PCS receive link is or was down 1: indicates that the PCS receive link is up. Once cleared, stays cleared until register read
1	LPWR_ABIL	Low Power Ability 1: The PCS implements a low power mode, meaning bit setting 11 of PCS Control register 1 register is supported and allows to place the PCS is a reset state.
0	—	Reserved

19.7.2.2.3 XFI PCS Device Identifier Upper Register

The XFI PCS Device Identifier Upper Register contains the upper half of the PCS Device Identifier, as shown in Figure 19-41

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PCS_DEV_ID_U															
W																
Special Access																
Reset	0000_0000_1000_0011															
Offset	0x0002															

Figure 19-41. XFI PCS Device Identifier Upper Register

Table 19-48. XFI PCS Device Identifier Upper Register Field Descriptions

Bits	Name	Description
15-0	PCS_DEV_ID_U	PCS Device ID Upper: 15:0 - OIU[3:18]

19.7.2.2.4 XFI PCS Device Identifier Lower Register

The XFI PCS Device Identifier Lower Register contains the lower half half of the PCS Device Identifier, as shown in Figure 19-42

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PCS_DEV_ID_L															
W																
Special Access																
Reset	1110_0100_0000_0000															
Offset	0x0003															

Figure 19-42. XFI PCS Device Identifier Lower Register

Table 19-49. XFI PCS Device Identifier Lower Register Field Descriptions

Bits	Name	Description
15-0	PCS_DEV_ID_L	PCS Device ID Lower: 15:10 - OIU[19:24] 9:4 - Manufacturer's Model Number 3:0 - Revision Number

19.7.2.2.5 XFI PCS Speed Ability Register

The XFI PCS Speed Ability Register contains ability bits for the 10GBASE-R PCS, as shown in Figure 19-43

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	—														10P_TS_2B_TL_CAP	10G_CAP
W																
Special Access																
Reset	0000_0000_0000_0001															
Offset	0x0004															

Figure 19-43. XFI PCS Speed Ability Register

Table 19-50. XFI PCS Speed Ability Register Field Descriptions

Bits	Name	Description
15-2	—	Reserved
1	10P_TS_2B_TL_CAP	10PASS-TS/2BASE-TL Capable 0: The PCS is not capable of operating as the 10P/2B PCS
0	10G_CAP	10G Capable 1: PCS is capable of operating at 10Gbps.

19.7.2.2.6 XFI PCS Devices in Package 0 Register

The XFI PCS Devices in Package 0 Register contains half of the devices present status, as shown in Figure 19-44

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	—								AN_PRES	TC_PRES	DTE_XS_PRES	PHY_XS_PRES	PCS_PRES	WIS_PRES	PMD_PMA_RES	C22_REG_PRES
W																
Special Access																
Reset	0000_0000_1000_1010															
Offset	0x0005															

Figure 19-44. XFI PCS Devices in Package 0 Register

Table 19-51. XFI PCS Devices in Package 0 Register Field Descriptions

Bits	Name	Description
15-8	—	Reserved
7	AN_PRES	Auto-negotiation Present 1: the PCS implements the Auto-Negotiation function
6	TC_PRES	TC Present 0: the PCS does not implement TC functions
5	DTE_XS_PRES	DTE XS Present 0: the PCS does not implement DTE XS functions
4	PHY_XS_PRES	PHY XS Present 0: the PCS does not implement PHY XS functions.
3	PCS_PRES	PCS Present 1: the PCS implements PCS functions
2	WIS_PRES	WIS Present 0: the PCS does not implement a WIS function.
1	PMD_PMA_RES	PMD/PMA Present 1: the PCS implements PMD/PMA functions (Link Training)
0	C22_REG_PRES	Clause 22 Registers Present 0: the PCS does not implement the Clause 22 Registers.

19.7.2.2.7 XFI PCS Devices in Package 1 Register

The XFI PCS Devices in Package 1 Register contains half of the devices present status, as shown in Figure 19-45

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	VEND_SPEC_DEV2_PRES	VEND_SPEC_DEV1_PRES	CLAUSE22_EXT_PRES	—												
W																
Special Access																
Reset	0000_0000_0000_0000															
Offset	0x0006															

Figure 19-45. XFI PCS Devices in Package 1 Register

Table 19-52. XFI PCS Devices in Package 1 Register Field Descriptions

Bits	Name	Description
15	VEND_SPEC_DEV2_PRES	Vendor Specific Device 2 Present 0: the PCS does not implement any vendor specific device.
14	VEND_SPEC_DEV1_PRES	Vendor Specific Device 1 Present 0: the PCS does not implement any vendor specific device.
12	CLAUSE22_EXT_PRES	Clause 22 Extension Present 0: the PCS does not implement Clause 22 extensions.
12-0	—	Reserved

19.7.2.2.8 XFI 10G PCS Control 2 Register

The XFI 10G PCS Control 2 Register contains control bits for the 10G PCS, as shown in Figure 19-46

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	—														PCS_TYP_SEL	
W																
Special Access																
Reset	0000_0000_0000_1011															
Offset	0x0007															

Figure 19-46. XFI 10G PCS Control 2 Register

Table 19-53. XFI PCS Control 2 Register Field Descriptions

Bits	Name	Description
15-3	—	Reserved
2-0	PCS_TYP_SEL	PCS Type Selection Read-only field, as this PCS only supports 10GBase-R functions. Note that the value of this field does not match the 802.3 encoding for 10GBase-R.

19.7.2.2.9 XFI 10G PCS Status 2 Register

The XFI 10G PCS Status 2 Register contains status bits for the 10G PCS, as shown in Figure 19-47

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DEV_PRES		—		TX_FAULT	RX_FAULT	—						10GBASET_CAP	10GBASEW_CAP	10GBASEX_CAP	10GBASER_CAP
W																
Special Access																
Reset	1000_0000_0000_0001															
Offset	0x0008															

Figure 19-47. XFI 10G PCS Status 2 Register

Table 19-54. XFI PCS Status 2 Register Field Descriptions

Bits	Name	Description
15-14	DEV_PRES	Device Present 10: device responding at this address
13-12	—	
11	TX_FAULT	Transmit Fault 0: no fault condition on transmit path 1: fault condition on transmit path
10	RX_FAULT	Receive Fault 0: no fault condition on receive path 1: fault condition on receive path
9-4	—	Reserved
3	10GBASET_CAP	10GBase-T Capable 0: PCS is not able to support 10GBase-T PCS type
2	10GBASEW_CAP	10GBase-W Capable 0: PCS is not able to support 10GBase-W PCS type
1	10GBASEX_CAP	10GBase-X Capable 0: PCS is not able to support 10GBase-X PCS type
0	10GBASER_CAP	10GBase-R Capable 1: PCS is able to support 10GBase-R PCS type

19.7.2.2.10 XFI PCS Package Identifier Upper Register

The XFI PCS Package Identifier Upper Register contains the upper half of a 32-bit unique identifier for a particular type of package that the PCS is instantiated within, as shown in Figure 19-48

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PCS_PKG_ID_U															
W																
Special Access																
Reset	0000_0000_1000_0011															
Offset	0x000E															

Figure 19-48. XFI PCS Package Identifier Upper Register

Table 19-55. XFI PCS Package Identifier Upper Register Field Descriptions

Bits	Name	Description
15-0	PCS_PKG_ID_U	PCS Package ID Upper: 15:0 - OIU[3:18]

19.7.2.2.11 XFI PCS Package Identifier Lower Register

The XFI PCS Package Identifier Lower Register contains the lower half of a 32-bit unique identifier for a particular type of package that the PCS is instantiated within, as shown in Figure 19-49

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PCS_PKG_ID_L															
W																
Special Access																
Reset	1110_0100_0000_0000															
Offset	0x000F															

Figure 19-49. XFI PCS Package Identifier Lower Register

Table 19-56. XFI PCS Package Identifier Lower Register Field Descriptions

Bits	Name	Description
15-0	PCS_PKG_ID_L	PCS Package ID Lower: 15:10 - OIU[19:24] 9:4 - Manufacturer's Model Number 3:0 - Revision Number

19.7.2.2.12 XFI 10GBASE-R PCS Status 1 Register

The XFI 10GBASE-R PCS Status 1 Register contains 10GBase-R PCS status, as shown in Figure 19-50

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	—			RX_LNK_STAT	—								PRBS9_ABIL	PRBS31_ABIL	PCS_HI_BER	PCS_BLK_LK
W																
Special Access																
Reset	0000_0000_0000_0000															
Offset	0x0020															

Figure 19-50. XFI 10GBASE-R PCS Status 1 Register

Table 19-57. XFI 10GBASE-R PCS Status 1 Register Field Descriptions

Bits	Name	Description
15-13	—	Reserved
12	RX_LNK_STAT	10GBase-R Receive Link Status 0: 10GBase-R PCS receive link is down 1: 10GBase-R PCS receive link is up
11-4	—	Reserved
3	PRBS9_ABIL	PRBS9 Pattern Testing Ability 0: the PCS does not support PRBS9 pattern testing
2	PRBS31_ABIL	PRBS31 Pattern Testing Ability 0: the PCS does not support PRBS31 pattern testing
1	PCS_HI_BER	10GBase-R PCS high BER 0: PCS not reporting a high BER 1: PCS reporting a high BER
0	PCS_BLK_LK	10GBase-R PCS block lock 0: PCS not locked to receive blocks 1: PCS locked to receive blocks

19.7.2.2.13 XFI 10GBASE-R PCS Status 2 Register

XFI 10GBASE-R PCS Status 2 Register contains PCS status bits, as shown in Figure 19-51

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	L_BLK_LK	LH_BER	BER						ERR_BLKS							
W																
Special Access	LL	LH														
Reset	0000_0000_0000_0000															
Offset	0x0021															

Figure 19-51. XFI 10GBASE-R PCS Status 2 Register

Table 19-58. XFI 10GBASE-R PCS Status 2 Register Field Descriptions

Bits	Name	Description
15	L_BLK_LK	Latched Block Lock 0: PCS does not have block lock 1: PCS has block lock When cleared, stays cleared until register read
14	LH_BER	Latched high BER 0: PCS has not reported a high BER 1: PCS has reported a high BER When set, stays set until register read
13-8	BER	BER Counter Does not roll over from 0x3F to 0x00 Cleared on register read or PCS reset
7-0	ERR_BLKS	Errored Blocks Counter Does not roll over from 0xFF to 0x00 Cleared on register read or PCS reset

19.7.2.2.14 XFI 10GBASE-R PCS Test Pattern Seed A Register 0

The XFI 10GBASE-R PCS Test Pattern Seed A Register 0 contains bits 15:0 of Test Pattern Seed A, as shown in Figure 19-52

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	TEST_PAT_SEED_A															
Special Access																
Reset	0000_0000_0000_0000															
Offset	0x0022															

Figure 19-52. XFI 10GBASE-R PCS Test Pattern Seed A Register 0

Table 19-59. XFI 10GBASE-R PCS Test Pattern Seed A Register 0 Field Descriptions

Bits	Name	Description
15-0	TEST_PAT_SEED_A	Test Pattern Seed A [15:0]

19.7.2.2.15 XFI 10GBASE-R PCS Test Pattern Seed A Register 1

The XFI 10GBASE-R PCS Test Pattern Seed A Register 1 contains bits 31:16 of Test Pattern Seed A, as shown in Figure 19-53

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	TEST_PAT_SEED_A															
Special Access																
Reset	0000_0000_0000_0000															
Offset	0x0023															

Figure 19-53. XFI 10GBASE-R PCS Test Pattern Seed A Register 1

Table 19-60. XFI 10GBASE-R PCS Test Pattern Seed A Register 1 Field Descriptions

Bits	Name	Description
15-0	TEST_PAT_SEED_A	Test Pattern Seed A [31:16]

19.7.2.2.16 XFI 10GBASE-R PCS Test Pattern Seed A Register 2

The XFI 10GBASE-R PCS Test Pattern Seed A Register 2 contains bits 47:32 of Test Pattern Seed A, as shown in Figure 19-54

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	TEST_PAT_SEED_A															
Special Access																
Reset	0000_0000_0000_0000															
Offset	0x0024															

Figure 19-54. XFI 10GBASE-R PCS Test Pattern Seed A Register 2

Table 19-61. XFI 10GBASE-R PCS Test Pattern Seed A Register 2 Field Descriptions

Bits	Name	Description
15-0	TEST_PAT_SEED_A	Test Pattern Seed A [47:32]

19.7.2.2.17 XFI 10GBASE-R PCS Test Pattern Seed A Register 3

The XFI 10GBASE-R PCS Test Pattern Seed A Register 3 contains bits 57:48 of Test Pattern Seed A, as shown in Figure 19-52

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	—;						TEST_PAT_SEED_A									
Special Access																
Reset	0000_0000_0000_0000															
Offset	0x0025															

Figure 19-55. XFI 10GBASE-R PCS Test Pattern Seed A Register 3

Table 19-62. XFI 10GBASE-R PCS Test Pattern Seed A Register 3 Field Descriptions

Bits	Name	Description
15-10	—	Reserved
9-0	TEST_PAT_SEED_A	Test Pattern Seed A [57:48]

19.7.2.2.18 XFI 10GBASE-R PCS Test Pattern Seed B Register 0

The XFI 10GBASE-R PCS Test Pattern Seed B Register 0 contains bits 15:0 of Test Pattern Seed B, as shown in Figure 19-52

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	TEST_PAT_SEED_B															
Special Access																
Reset	0000_0000_0000_0000															
Offset	0x0026															

Figure 19-56. XFI 10GBASE-R PCS Test Pattern Seed B Register 0

Table 19-63. XFI 10GBASE-R PCS Test Pattern Seed B Register 0 Field Descriptions

Bits	Name	Description
15-0	TEST_PAT_SEED_B	Test Pattern Seed B [15:0]

19.7.2.2.19 XFI 10GBASE-R PCS Test Pattern Seed B Register 1

The XFI 10GBASE-R PCS Test Pattern Seed B Register 1 contains bits 31:16 of Test Pattern Seed B, as shown in Figure 19-53

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	TEST_PAT_SEED_B															
Special Access																
Reset	0000_0000_0000_0000															
Offset	0x0027															

Figure 19-57. XFI 10GBASE-R PCS Test Pattern Seed B Register 1

Table 19-64. XFI 10GBASE-R PCS Test Pattern Seed B Register 1 Field Descriptions

Bits	Name	Description
15-0	TEST_PAT_SEED_B	Test Pattern Seed B [31:16]

19.7.2.2.20 XFI 10GBASE-R PCS Test Pattern Seed B Register 2

The XFI 10GBASE-R PCS Test Pattern Seed B Register 2 contains bits 47:32 of Test Pattern Seed B, as shown in Figure 19-54

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	TEST_PAT_SEED_A															
Special Access																
Reset	0000_0000_0000_0000															
Offset	0x0028															

Figure 19-58. XFI 10GBASE-R PCS Test Pattern Seed B Register 2

Table 19-65. XFI 10GBASE-R PCS Test Pattern Seed B Register 2 Field Descriptions

Bits	Name	Description
15-0	TEST_PAT_SEED_B	Test Pattern Seed B [47:32]

19.7.2.2.21 XFI 10GBASE-R PCS Test Pattern Seed B Register 3

The XFI 10GBASE-R PCS Test Pattern Seed B Register 3 contains bits 57:48 of Test Pattern Seed B, as shown in Figure 19-52

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	—						TEST_PAT_SEED_B									
Special Access																
Reset	0000_0000_0000_0000															
Offset	0x0029															

Figure 19-59. XFI 10GBASE-R PCS Test Pattern Seed B Register 3

Table 19-66. XFI 10GBASE-R PCS Test Pattern Seed B Register 3 Field Descriptions

Bits	Name	Description
15-10	—	Reserved
9-0	TEST_PAT_SEED_B	Test Pattern Seed B [57:48]

19.7.2.2.22 XFI 10GBASE-R PCS Test Pattern Control Register

The XFI 10GBASE-R PCS Test Pattern Control Register contains control bits for test pattern function, as shown in Figure 19-60

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	—									PRBS9_TX_TEST_PAT_EN	PRBS31_RX_TEST_PAT_EN	PRBS31_TX_TEST_PAT_EN	TX_TEST_PAT_EN	RX_TEST_PAT_EN	TEST_PAT_SEL	DATA_PAT_SEL
W																
Special Access																
Reset	0000_0000_0000_0000															
Offset	0x002A															

Figure 19-60. XFI 10GBASE-R PCS Test Pattern Control Register

Table 19-67. XFI 10GBASE-R PCS Test Pattern Control Register Field Descriptions

Bits	Name	Description
15-7	—	Reserved
6	PRBS9_TX_TEST_PAT_EN	PRBS9 transmit test pattern enable 0: Disable PRBS9 test-pattern mode on the transmit path Read-only always set to 0
5	PRBS31_RX_TEST_PAT_EN	PRBS31 receive test pattern enable 0: Disable PRBS31 test-pattern mode on the receive path Read-only always set to 0
4	PRBS31_TX_TEST_PAT_EN	PRBS31 transmit test pattern enable 0: Disable PRBS31 test-pattern mode on the transmit path Read-only always set to 0
3	TX_TEST_PAT_EN	Transmit test pattern enable 0: Disable transmit test pattern 1: Enable transmit test pattern
2	RX_TEST_PAT_EN	Receive test pattern enable 0: Disable receive test pattern 1: Enable receive test pattern
1	TEST_PAT_SEL	Test pattern select 0: Pseudo-random test pattern 1: Square wave test pattern
0	DATA_PAT_SEL	Transmit test pattern enable 0: LF data pattern 1: Zeroes pattern

19.7.2.2.23 XFI 10GBASE-R PCS Test Pattern Error Counter Register

The XFI 10GBASE-R PCS Test Pattern Error Counter Register contains the error counter for the test pattern, as shown in Figure 19-61

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TEST_PAT_ERR_CNT															
W																
Special Access																
Reset	0000_0000_0000_0000															
Offset	0x002B															

Figure 19-61. XFI 10GBASE-R PCS Test Pattern Error Counter Register

Table 19-68. XFI 10GBASE-R PCS Test Pattern Error Counter Register Field Descriptions

Bits	Name	Description
15-0	TEST_PAT_ERR_CNT	Test Pattern Error Counter

19.7.2.2.24 XFI Vendor Specific PCS Status Register

The XFI Vendor Specific PCS Status Register contains the status for the XFI PCS, as shown in Figure 19-62

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	—														TX_FIFO_FAULT	RX_FIFO_FAULT
W																
Special Access															LH	LH
Reset	0000_0000_0000_0000															
Offset	0x8000															

Figure 19-62. XFI Vendor Specific PCS Status Register

Table 19-69. XFI 10GBASE-R PCS Test Pattern Error Counter Register Field Descriptions

Bits	Name	Description
15-2	—	Reserved
1	TX_FIFO_FAULT	Transmit FIFO Fault 0: No error on the transmit decoupling FIFO 1: Error (overflow or underflow) on the transmit decoupling FIFO Once set, stays set until register read
0	RX_FIFO_FAULT	Receive FIFO Fault 0: No error on the receive decoupling FIFO 1: Error (overflow or underflow) on the receive decoupling FIFO Once set, stays set until register read

19.7.2.3 XFI Auto-Negotiation Registers

The XFI Auto-Negotiation MDIO space includes the registers required to control the functions of Clause 73 Auto-Negotiation for Backplane Ethernet.

19.7.2.3.1 XFI AN Control Register

The AN Control Register contains general controls for auto-negotiation, as shown in Figure 19-63

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	AN_RST	—	EXT_NP_CTRL	AN_EN	—		RESTART_AN	—								
Special Access	SC						SC									
Reset	0000_0000_0000_0000															
Offset	0x0000															

Figure 19-63. XFI AN Control Register

Table 19-70. XFI AN Control Register Field Descriptions

Bits	Name	Description
15	AN_RST	AN reset 0: AN normal operation 1: AN reset Self-clearing bit
14	—	Reserved

Table 19-70. XFI AN Control Register Field Descriptions

13	EXT_NP_CTRL	Extended Next Page Control 0: Extended next pages are disabled 1: Extended next pages are enabled
12	AN_EN	Auto-negotiation enable 0: Disable auto-negotiation process 1: Enable auto-negotiation process
11-10	—	Reserved
9	RESTART_AN	Restart auto-negotiation 0: Auto-negotiation in process, disabled, or not supported 1: Restart auto-negotiation process
8-0	—	Reserved

19.7.2.3.2 XFI AN Status Register

The XFI AN Status Register contains status bits for the XFI auto-negotiate function, as shown in Figure 19-64

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	—						PAR_DET_FAULT	—	EXT_NP_STAT	PAGE_RCV	AN_COMP	REM_FAULT	AN_ABIL	LNK_STAT	—	LP_AN_ABIL
W																
Special Access							LH			LH		LH		LL	LH	
Reset	0000_0000_0000_1000															
Offset	0x0001															

Figure 19-64. XFI AN Status Register

Table 19-71. XFI AN Status Register Field Descriptions

Bits	Name	Description
15-10	—	Reserved
9	PAR_DET_FAULT	Parallel detection fault 0: A fault has not been detected via the parallel detection function 1: A fault has been detected via the parallel detection function Once set, stays set until register read
8	—	Reserved
7	EXT_NP_STAT	Extended next page status 0: Extended next page is not allowed 1: Extended next page format is used
6	PAGE_RCV	Page received 0: A page has not been received 1: A page has been received Once set, stays set until register read

Table 19-71. XFI AN Status Register Field Descriptions

5	AN_COMP	Auto-Negotiation complete 0: Auto-negotiation process not completed 1: Auto-negotiation process completed
4	REM_FAULT	Remote Fault 0: No remote fault condition detected 1: Remote fault condition detected Once set, stays set until register read.
3	AN_ABIL	Auto-Negotiation Ability 1: PHY is able to perform Auto-Negotiation
2	LNK_STAT	Link Status 0: Link is down 1: Link is up Once set, stays set until register read.
1	—	Reserved
0	LP_AN_ABIL	Link partner Auto-Negotiation ability 0: LP is not able to perform Auto-Negotiation 1: LP is able to perform Auto-Negotiation

19.7.2.3.3 XFI AN Device Identifier Upper Register

The AN Device Identifier Upper Register contains the upper half of the 32-bit device identifier, as shown in Figure 19-65

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	AN_DEV_ID_U															
W																
Special Access																
Reset	0000_0000_1000_0011															
Offset	0x0002															

Figure 19-65. XFI AN Device Identifier Upper Register

Table 19-72. XFI AN Device Identifier Upper Register Field Descriptions

Bits	Name	Description
15-0	AN_DEV_ID_U	AN Device ID Upper: 15:0 - OIU[3:18]

19.7.2.3.4 XFI AN Device Identifier Lower Register

The AN Device Identifier Lower Register contains the lower half of the 32-bit device identifier, as shown in Figure 19-66

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	AN_DEV_ID_L															
W																
Special Access																
Reset	1110_0100_0000_0000															
Offset	0x0003															

Figure 19-66. XFI AN Device Identifier Lower Register

Table 19-73. XFI AN Device Identifier Lower Register Field Descriptions

Bits	Name	Description
15-0	AN_DEV_ID_L	AN Device ID Lower: 15:10 - OIU[19:24] 9:4 - Manufacturer's Model Number 3:0 - Revision Number

19.7.2.3.5 XFI AN Devices in Package 0 Register

The XFI AN Devices in Package 0 Register contains half of the AN devices in package status, as shown in Figure 19-67

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R									AN_PRES	TC_PRES	DTE_XS_PRES	PHY_XS_PRES	PCS_PRES	WIS_PRES	PMD_PMA_PRES	CLASE_22_REG_PRES
W																
Special Access																
Reset	0000_0000_1000_1010															
Offset	0x0005															

Figure 19-67. XFI AN Devices in Package 0 Register

Table 19-74. XFI AN Devices in Package 0 Register Field Descriptions

Bits	Name	Description
15-8	—	Reserved
7	AN_PRES	Auto-negotiation Present 1: the PCS implements the Auto-Negotiation function
6	TC_PRES	TC Present 0: the PCS does not implement TC functions
5	DTE_XS_PRES	DTE XS Present 0: the PCS does not implement DTE XS functions
4	PHY_XS_PRES	PHY XS Present 0: the PCS does not implement PHY XS functions.
3	PCS_PRES	PCS Present 1: the PCS implements PCS functions
2	WIS_PRES	WIS Present 0: the PCS does not implement a WIS function.
1	PMD_PMA_RES	PMD/PMA Present 1: the PCS implements PMD/PMA functions (Link Training)
0	C22_REG_PRES	Clause 22 Registers Present 0: the PCS does not implement the Clause 22 Registers.

19.7.2.3.6 XFI AN Devices in Package 1 Register

The XFI AN Devices in Package 1 Register contains half of the AN devices in package status, as shown in Figure 19-68

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	VEND_SPEC_DEV2_PRES	VEND_SPEC_DEV1_PRES	CLAUSE22_EXT_PRES													
W																
Special Access																
Reset	0000_0000_0000_0000															
Offset	0x0006															

Figure 19-68. XFI AN Devices in Package 1 Register

Table 19-75. XFI AN Devices in Package 1 Register Field Descriptions

Bits	Name	Description
15	VEND_SPEC_DEV2_PRES	Vendor Specific Device 2 Present 0: the PCS does not implement any vendor specific device.
14	VEND_SPEC_DEV1_PRES	Vendor Specific Device 1 Present 0: the PCS does not implement any vendor specific device.
12	CLAUSE22_EXT_PRES	Clause 22 Extension Present 0: the PCS does not implement Clause 22 extensions.
12-0	—	Reserved

19.7.2.3.7 XFI AN Package Identifier Upper Register

The AN Package Identifier Upper Register contains the upper half of the 32-bit package identifier, as shown in Figure 19-69

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	AN_PKG_ID_U															
W																
Special Access																
Reset	0000_0000_1000_0011															
Offset	0x000E															

Figure 19-69. XFI AN Package Identifier Upper Register

Table 19-76. XFI AN Package Identifier Upper Register Field Descriptions

Bits	Name	Description
15-0	AN_PKG_ID_U	AN Package ID Upper: 15:0 - OIU[3:18]

19.7.2.3.8 XFI AN Package Identifier Lower Register

The AN Package Identifier Lower Register contains the lower half of the 32-bit package identifier, as shown in Figure 19-70

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	AN_PKG_ID_L															
W																
Special Access																
Reset	1110_0100_0000_0000															
Offset	0x000F															

Figure 19-70. XFI AN Package Identifier Lower Register

Table 19-77. XFI AN Package Identifier Lower Register Field Descriptions

Bits	Name	Description
15-0	AN_PKG_ID_L	AN Package ID Lower: 15:10 - OIU[19:24] 9:4 - Manufacturer's Model Number 3:0 - Revision Number

19.7.2.3.9 XFI AN Advertisement Register 0

The XFI AN Advertisement Register 0 contains bits 15:0 of the 48-bit page bits used during base page exchange, as shown in Figure 19-71 They should be programmed before auto-negotiation is started. The 48-bit value is distributed over registers 0, 1, 2, where register 2 holds the most significant 16 bits of value.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	NP	ACK	REM_FAULT	XNP_CAP	PAUSE_CAP		ECHOED_NONCE					SELECT_FLD				
W																
Special Access																
Reset	0000_0000_0000_0001															
Offset	0x0010															

Figure 19-71. XFI AN Advertisement Register 0

Table 19-78. XFI AN Advertisement Register 0 Field Descriptions

Bits	Name	Description
15	NP	Next Page: 0: Device does not have a Next Page to send 1: Device has a Next Page to send
14	ACK	Should always be set to 0
13	REM_FAULT	Remote Fault 0: Normal operation 1: Device is indicating a remote fault condition
12	XNP_CAP	Extended Next Page Capability 0: Device does not support extended next pages
11-10	PAUSE_CAP	Pause Capability (ASM_DIR:PAUSE) 00: No PAUSE 01: Symmetric PAUSE 10: Asymmetric PAUSE toward link partner 11: Both Symmetric PAUSE and Asymmetric PAUSE toward local device
9-5	ECHOED_NONCE	Echoed Nonce Nonce value received from link partner
4-0	SELECT_FLD	Selector Field 0001: IEEE Standard 802.3

19.7.2.3.10 XFI AN Advertisement Register 1

The XFI AN Advertisement Register 1 contains bits 31:16 of the 48-bit page bits used during base page exchange, as shown in Figure 19-72 They should be programmed before auto-negotiation is started. The 48-bit value is distributed over registers 0, 1, 2, where register 2 holds the most significant 16 bits of value.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TECH_A10_A0											TRANS_NONCE				
W																
Special Access																
Reset	0000_0000_1001_1111															
Offset	0x0011															

Figure 19-72. XFI AN Advertisement Register 1

Table 19-79. XFI AN Advertisement Register 1 Field Descriptions

Bits	Name	Description
15-5	TECH_A10_A0	Technology Field (A10:A0) A10:A3: reserved must be set to 0 A2: 10GBase-KR A1:A0: reserved must be set to 0
4-0	TRANS_NONCE	Transmitted Nonce Must be set to unique value per device prior to enabling auto-negotiation Defaults to 0h1F

19.7.2.3.11 XFI AN Advertisement Register 2

The XFI AN Advertisement Register 0 contains bits 47:32 of the 48-bit page bits used during base page exchange, as shown in Figure 19-73 They should be programmed before auto-negotiation is started. The 48-bit value is distributed over registers 0, 1, 2, where register 2 holds the most significant 16 bits of value.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FEC_CAP		TECH_A24_A11													
W																
Special Access																
Reset	0000_0000_0000_0000															
Offset	0x0012															

Figure 19-73. XFI AN Advertisement Register 2

Table 19-80. XFI AN Advertisement Register 2 Field Descriptions

Bits	Name	Description
15-14	FEC_CAP	FEC Capability 00: Device is not capable of or requesting FEC
13-0	TECH_A24_A11	Technology Field (A24:A11) A24:A11:reserved must be set to 0

19.7.2.3.12 XFI AN LP Base Page Ability Register 0

The XFI AN LP Base Page Ability Register 0 contains bits 15:0 of the link partner base storage, as shown in Figure 19-74 The 48-bit value is stored over registers 0, 1 and 2, where register 2 holds the most

significant 16 bits of the value. The contents of the registers are identical to the XFI AN avertisement registers (see above).

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	NP	ACK	REM_FAULT	XNP_CAP	PAUSE_CAP		ECHOED_NONCE						SELECT_FLD			
W																
Special Access																
Reset	0000_0000_0000_0000															
Offset	0x0013															

Figure 19-74. XFI AN LP Base Ability Register 0

Table 19-81. XFI AN LP Base Page Ability Register 0 Field Descriptions

Bits	Name	Description
15	NP	Next Page: 0: Link Partner does not have a Next Page to send 1: Link Partner has a Next Page to send
14	ACK	Acknowledge 0: Base Ability data not valid 1: Base Ability data valid
13	REM_FAULT	Remote Fault 0: Normal operation 1: Link Partner is indicating a remote fault condition
12	XNP_CAP	Extended Next Page Capability 0: Link Partner does not support extended next pages 1: Link Partner supports extended next pages
11-10	PAUSE_CAP	Pause Capability (ASM_DIR:PAUSE) 00: No PAUSE 01: Symmetric PAUSE 10: Asymmetric PAUSE toward local device 11: Both Symmetric PAUSE and Asymmetric PAUSE toward link partner
9-5	ECHOED_NONCE	Echoed Nonce Nonce value received by link partner
4-0	SELECT_FLD	Selector Field 0001: IEEE Standard 802.3

19.7.2.3.13 XFI AN LP BASE Page Ability Register 1

The XFI AN LP Base Page Ability Register 1 contains bits 31:16 of the link partner base storage, as shown in Figure 19-74 The 48-bit value is stored over registers 0, 1 and 2, where register 2 holds the most

significant 16 bits of the value. The contents of the registers are identical to the XFI AN avertisement registers (see above).

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TECH_A10_A0											TRANS_NONCE				
W																
Special Access																
Reset	0000_0000_0000_0000															
Offset	0x0014															

Figure 19-75. XFI AN LP Base Page Ability Register 1

Table 19-82. XFI AN LP Base Page Ability Register 1 Field Descriptions

Bits	Name	Description
15-5	TECH_A10_A0	Technology Field (A10:A0) A10:A3: reserved must be set to 0 A2: 10GBase-KR A1:A0: reserved must be set to 0
4-0	TRANS_NONCE	Transmitted Nonce Must be set to unique value per device prior to enabling auto-negotiation

19.7.2.3.14 XFI AN LP Base Page Ability Register 2

The XFI AN LP Base Page Ability Register 2 contains bits 47:31 of the link partner base storage, as shown in Figure 19-74 The 48-bit value is stored over registers 0, 1 and 2, where register 2 holds the most significant 16 bits of the value. The contents of the registers are identical to the XFI AN avertisement registers (see above).

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FEC_CAP		TECH_A24_A11													
W																
Special Access																
Reset	0000_0000_0000_0000															
Offset	0x0015															

Figure 19-76. XFI AN LP Base Page Ability Register 2

Table 19-83. XFI AN LP Base Page Ability Register 2 Field Descriptions

Bits	Name	Description
15-14	FEC_CAP	FEC Capability (F1:F0) 00: Link partner is not capable of or requesting FEC 01: Link partner is capable of but not requesting FEC 11: Link partner is capable of and requesting FEC
13-0	TECH_A24_A11	Technology Field (A24:A11) A24:A11:reserved must be set to 0

19.7.2.3.15 XFI AN XNP Transmit Register 0

The XFI AN XNP Transmit Register 0 contains bits 15:0 of the extended next page transmit data, as shown in Figure 19-77

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	NP	—	MSG_ PAGE	ACK2	TOG GLE	MSG_UNF_CODE_FLD										
Special Access																
Reset	0010_0000_0000_0001															
Offset	0x0016															

Figure 19-77. XFI AN XNP Transmit Register 0

Table 19-84. XFI AN XNP Transmit Register 0 Field Descriptions

Bits	Name	Description
15	NP	Next Page 0: Device does not have any more Next Pages to send 1: Device has more Next Pages to Send
14	—	Reserved
13	MSG_PAGE	Message Page 0: Next page is an unformatted page 1: Next page is a message page
12	ACK2	Acknowledge 2 0: The receiver is not able to act on the information defined in the message 1: The receiver is able to act on the information defined in the message
11	TOGGLE	Toggle
10-0	MSG_UNF_CO DE_FLD	Message/Unformatted Code Field When MSG_PAGE=1: Message code field, else unformatted code field. For the null message code, the value is 0x1

19.7.2.3.16 XFI AN XNP Transmit Register 1

The XFI AN XNP Transmit Register 1 contains bits 31:16 of the extended next page transmit data, as shown in Figure 19-78

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	UNF_CODE_FLD1															
W																
Special Access																
Reset	0000_0000_0000_0000															
Offset	0x0017															

Figure 19-78. XFI AN XNP Transmit Register 1

Table 19-85. XFI AN XNP Transmit Register 1 Field Descriptions

Bits	Name	Description
15-0	UNF_CODE_FLD1	Unformatted Code Field 1

19.7.2.3.17 XFI AN XNP Transmit Register 2

The XFI AN XNP Transmit Register 2 contains bits 47:32 of the extended next page transmit data, as shown in Figure 19-79

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	UNF_CODE_FLD2															
W																
Special Access																
Reset	0000_0000_0000_0000															
Offset	0x0018															

Figure 19-79. XFI AN XNP Transmit Register 2

Table 19-86. XFI AN XNP Transmit Register 2 Field Descriptions

Bits	Name	Description
15-0	UNF_CODE_FLD2	Unformatted Code Field 2

19.7.2.3.18 XFI AN LP XNP Ability Register 0

The XFI AN LP XNP Ability Register 0 contains bits 15:0 of the extended next page value from the remote device, as shown in Figure 19-80

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	NP	—	MSG_PAGE	ACK2	TOGGLE	MSG_UNF_CODE_FLD										
W																
Special Access																
Reset	0000_0000_0000_0000															
Offset	0x0019															

Figure 19-80. XFI AN LP XNP Ability Register 0

Table 19-87. XFI AN LP XNP Ability Register 0 Field Descriptions

Bits	Name	Description
15	NP	Next Page 0: Device does not have any more Next Pages to send 1: Device has more Next Pages to Send
14	—	Reserved
13	MSG_PAGE	Message Page 0: Next page is an unformatted page 1: Next page is a message page
12	ACK2	Acknowledge 2 0: The receiver is not able to act on the information defined in the message 1: The receiver is able to act on the information defined in the message
11	TOGGLE	Toggle
10-0	MSG_UNF_CODE_FLD	Message/Unformatted Code Field When MSG_PAGE=1: Message code field, else unformatted code field. For the null message code, the value is 0x1

19.7.2.3.19 XFI AN LP XNP Ability Register 1

The XFI AN LP XNP Ability Register 1 contains bits 31:16 of the extended next page value from the remote device, as shown in Figure 19-81

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	UNF_CODE_FLD1															
W																
Special Access																
Reset	0000_0000_0000_0000															
Offset	0x001A															

Figure 19-81. XFI AN LP XNP Ability Register 1

Table 19-88. XFI AN LP XNP Ability Register 1 Field Descriptions

Bits	Name	Description
15-0	UNF_CODE_FLD1	Unformatted Code Field 1

19.7.2.3.20 XFI AN LP XNP Ability Register 2

The XFI AN LP XNP Ability Register 2 contains bits 47:32 of the extended next page value from the remote device, as shown in Figure 19-82

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	UNF_CODE_FLD2															
W																
Special Access																
Reset	0000_0000_0000_0000															
Offset	0x001B															

Figure 19-82. XFI AN LP XNP Ability Register 2

Table 19-89. XFI AN LP XNP Ability Register 2 Field Descriptions

Bits	Name	Description
15-0	UNF_CODE_FLD2	Unformatted Code Field 2

19.7.2.3.21 XFI Backplane Ethernet Status Register

The XFI Backplane Ethernet Status Register provides the result after auto-negotiation, as shown in Figure 19-83

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	—							100GBASE_CR10	—	40GBASE-CR4	40GBASE-KR4	BASE_R_FEC	10GBASE-KR	10GBASE-KX4	1000BASE-KX	BP_AB_ABIL
W																
Special Access																
Reset	0000_0001_0110_1111															
Offset	0x0030															

Figure 19-83. XFI Backplane Ethernet Status Register

Table 19-90. XFI AN Backplane Ethernet Status Register Field Descriptions

Bits	Name	Description
15-9	—	Reserved
8	100GBASE_CR10	Always 0 after negotiation complete
7	—	Reserved
6	40GBASE_CR4	Always 0 after negotiation complete
5	40GBASE_KR4	Always 0 after negotiation complete
4	BASE_R_FEC	Always 0 after negotiation complete
3	10GBASE_KR	10GBase-KR 0: Not auto-negotiated to 10GBase-KR 1: Auto-negotiated to 10GBase-KR
2	10GBASE_KX4	Always 0 after negotiation complete
1	1000BASE_KX	10GBase-KR
0	BP_AN_ABIL	Backplane Base-R capable PHY type is implemented Always 1

19.7.2.3.22 XFI Millisecond Counter Register

The XFI Millisecond Counter Register controls the upper 16 bits of the 18-bit counter for counting 1 ms, as shown in Figure 19-84

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	COUNT_MS															
W																
Special Access																
Reset	1001_1000_1001_0110															
Offset	0x8000															

Figure 19-84. XFI Millisecond Counter Register

Table 19-91. XFI Millisecond Counter Register Field Descriptions

Bits	Name	Description
15-0	COUNT_MS	Count Milliconds Upper 16-bits value of the 18-bit counter is provided for counting 1ms. The milliseconds counter operates on 66bit samples (i.e. 6.4ns) incrementing by 4 with every sample (i.e. 0x9896*4*6.4ns=1ms)

19.7.2.4 XFI Vendor-Specific 1 Registers

19.7.2.4.1 XFI Revision Register

The XFI Revision Register contains revision information for the XFI PCS, as shown in Figure 19-85

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CFG_REV								IP_VER				IP_REV			
W																
Special Access																
Reset	0001_0000_0001_0000															
Offset	0x0000															

Figure 19-85. XFI Revision Register

Table 19-92. XFI Revision Register Field Descriptions

Bits	Name	Description
15-8	CFG_VER	Integration Version
7-4	IP_VER	IP Version
3-0	IP_REV	IP Revision

19.7.2.4.2 XFI Scratch Register

The XFI Scratch Register contains a read/writeable scratch register that can be used to test MDIO register access, as shown in Figure 19-86

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SCRATCH															
W																
Special Access																
Reset	0000_0000_0000_0000															
Offset	0x0001															

Figure 19-86. XFI Scratch Register

Table 19-93. XFI Scratch Register Field Descriptions

Bits	Name	Description
15-	SCRATCH	Scratch register

19.7.2.4.3 XFI PCS Interrupt Event Register

The XFI PCS Interrupt Event Register contains interrupt event bits for PCS function, as shown in Figure 19-87

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	—						PCS_TX_LP_IDLE	PCS_RX_LP_IDLE	PCS_TX_LOC_FAULT	PCS_TX_REM_FAULT	PCS_RX_LOC_FAULT	PCS_RX_REM_FAULT	PCS_HI_BER	PCS_BLK_LK_LOSS	TX_BUF_FAULT	RX_BUF_FAULT
W																
Special Access							RC	RC	RC	RC	RC	RC	RC	RC	RC	RC
Reset	0000_0000_0000_0000															
Offset	0x0002															

Figure 19-87. XFI PCS Interrupt Event Register

Table 19-94. XFI PCS Interrupt Event Register Field Descriptions

Bits	Name	Description
15-10	—	Reserved
9	PCS_TX_LP_IDLE	PCS Tx Low Power Idle 0: PCS transmit path has not detected a Low Power Idle condition 1: PCS trnasmit path has detected a Low Power Idle condition Cleared on register read
8	PCS_RX_LP_IDLE	PCS Rx Low Power Idle 0: PCS receive path has not detected a Low Power Idle condition 1: PCS receive path has detected a Low Power Idle condition Cleared on register read
7	PCS_TX_LOC_FAULT	PCS Tx Local Fault 0: PCS transmit path has not detected a Local Fault condition 1: PCS transmit path has detected a Local Fault condition Cleared on register read
6	PCS_TX_REM_FAULT	PCS Tx Remote Fault 0: PCS transmit path has not detected a Remote Fault condition 1: PCS transmit path has detected a Remote Fault condition Cleared on register read
5	PCS_RX_LOC_FAULT	PCS Rx Local Fault 0: PCS receive path has not detected a Local Fault condition 1: PCS receive path has detected a Local Fault condition Cleared on register read

Table 19-94. XFI PCS Interrupt Event Register Field Descriptions

4	PCS_RX_REM_FAULT	PCS Rx Remote Fault 0: PCS receive path has not detected a Remote Fault condition 1: PCS receive path has detected a Remote Fault condition Cleared on register read
3	PCS_HI_BER	PCS High BER 0: PCS receive path has not detected a High Bit Error Rate condition 1: PCS receive path has detected a High Bit Error Rate condition Cleared on register read
2	PCS_BLK_LK_LOSS	PCS Block Lock Loss 0: PCS block lock has not lost synchronization 1: PCS block lock has lost synchronization Cleared on register read
1	TX_BUF_FAULT	Tx Buffer Fault 0: No buffer error detected on Transmit Rate Matching FIFO 1: Buffer error detected on Transmit Rate Matching FIFO Cleared on register read
0	RX_BUF_FAULT	Rx Buffer Fault 0: No buffer error detected on Receive Rate Matching FIFO 1: Buffer error detected on Receive Rate Matching FIFO Cleared on register read

19.7.2.4.4 XFI PCS Interrupt Mask Register

The XFI PCS Interrupt Mask Register contains the interrupt mask bits for the above PCS interrupt event bits, as shown in Figure 19-88

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	—						PCS_TX_LP_IDLE_EN	PCS_RX_LP_IDLE_EN	PCS_TX_LOC_FAULT_EN	PCS_TX_REM_FAULT_EN	PCS_RX_LOC_FAULT_EN	PCS_RX_REM_FAULT_EN	PCS_HI_BER_EN	PCS_BLK_LK_LOSS_EN	TX_BUF_FAULT_EN	RX_BUF_FAULT_EN
Special Access																
Reset	0000_0000_0000_0000															
Offset	0x0003															

Figure 19-88. XFI PCS Interrupt Mask Register

Table 19-95. XFI PCS Interrupt Mask Register Field Descriptions

Bits	Name	Description
15-10	—	Reserved
9	PCS_TX_LP_IDLE_EN	PCS Tx Low Power Idle Interrupt Mask 0: PCS Tx Low Power Idle events do not cause an interrupt 1: PCS Tx Low Power Idle events cause an interrupt
8	PCS_RX_LP_IDLE_EN	PCS Rx Low Power Idle 0: PCS Rx Low Power Idle events do not cause an interrupt 1: PCS Rx Low Power Idle events cause an interrupt
7	PCS_TX_LOC_FAULT_EN	PCS Tx Local Fault 0: PCS Tx Local Fault events do not cause an interrupt 1: PCS Tx Local Fault events cause an interrupt
6	PCS_TX_REM_FAULT_EN	PCS Tx Remote Fault 0: PCS Tx Remote Fault events do not cause an interrupt 1: PCS Tx Remote Fault events cause an interrupt
5	PCS_RX_LOC_FAULT_EN	PCS Rx Local Fault 0: PCS Rx Local Fault events do not cause an interrupt 1: PCS Rx Local Fault events cause an interrupt
4	PCS_RX_REM_FAULT_EN	PCS Rx Remote Fault 0: PCS Rx Remote Fault events do not cause an interrupt 1: PCS Rx Remote Fault events cause an interrupt
3	PCS_HI_BER_EN	PCS High BER 0: PCS High BER events do not cause an interrupt 1: PCS High BER events cause an interrupt
2	PCS_BLK_LK_LOSS_EN	PCS Block Lock Loss 0: PCS Block Lock Loss events do not cause an interrupt 1: PCS Block Lock Loss events cause an interrupt
1	TX_BUF_FAULT_EN	Tx Buffer Fault 0: PCS Tx Buffer Fault events do not cause an interrupt 1: PCS Tx Buffer Fault events cause an interrupt
0	RX_BUF_FAULT_EN	Rx Buffer Fault 0: PCS Rx Buffer Fault events do not cause an interrupt 1: PCS Rx Buffer Fault events cause an interrupt

19.7.2.4.5 XFI Auto-Negotiation Interrupt Event Register

The XFI Auto-Negotiation Interrupt Event Register contains interrupt events bits related to auto-negotiation, as shown in Figure 19-89 Note that the XFI Auto-Negotiation Interrrupt Event register bits are cleared when read.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	—											NEXT_PAGE	BASE_PAGE	AN_PAR_DET_FAULT	AN_PG_RCV	AN_DONE
W																
Special Access												RC	RC	RC	RC	RC
Reset	0000_0000_0000_0000															
Offset	0x0004															

Figure 19-89. XFI Auto-Negotiation Interrupt Event Register

Table 19-96. XFI AN Interrupt Event Register Field Descriptions

Bits	Name	Description
15-5	—	Reserved
4	NEXT_PAGE	Next Page Received 0: No next page received 1: Next page received and Partner Next Page Ability register is set Asserted together with AN_PAGE_RX Cleared on register read
3	BASE_PAGE	Base Page Received 0: No base page received 1: Base page received and Partner Ability register is set Asserted together with AN_PAGE_RX Cleared on register read
2	AN_PARDET_FLT	Parallel detection fault 0: No fault detected 1: Ambiguous link status detected while waiting on DME page receive Cleared on register read
1	AN_PAGE_RX	Auto-negotiate page receive 0: An ability word was not received from the remote device 1: An ability word was received from the remote device during backplane auto-negotiation Indicates the validity of the remote ability word (base page or next page) Cleared on register read
0	AN_DONE	Backplane auto-negotiation complete 0: Backplane auto-negotiation not complete 1: Backplane auto-negotiation complete Cleared on register read

19.7.2.4.6 XFI Auto-Negotiation Interrupt Mask Register

The XFI Auto-Negotiation Interrupt Mask Register contains interrupt mask bits for the above auto-negotiate interrupt events, as shown in Figure 19-90

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W												NEXT_PAGE_EN	BASE_PAGE_EN	AN_PAR_DET_FAULT_EN	AN_PG_RCV_EN	AN_DONE_EN
Special Access																
Reset	0000_0000_0000_0000															
Offset	0x0005															

Figure 19-90. XFI Auto-Negotiation Interrupt Mask Register

Table 19-97. XFI AN Interrupt Mask Register Field Descriptions

Bits	Name	Description
15-5	—	Reserved
4	NEXT_PAGE	Next Page Received Event Enable 0: AN Next Page Received will not cause an interrupt event 1: AN Next Page Received will cause an interrupt event
3	BASE_PAGE	Base Page Received Event Enable 0: AN Base Page Received will not cause an interrupt event 1: AN Base Page Received will cause an interrupt event
2	AN_PARDET_FLT_EN	AN Parallel Detection Fault Event Enable 0: AN Parallel Dection Fault will not cause an interrupt event 1: AN Parallel Dection Fault will cause an interrupt event
1	AN_PAGE_RX_EN	AN Page Recieve Event Enable 0: AN Page Receive will not cause an interrupt event 1: AN Page Receive will cause an interrupt event
0	AN_DONE_EN	AN Done Event Enable 0: AN done will not cause an interrupt event 1: AN done will cause an interrupt event

19.7.2.4.7 XFI Link Training Interrupt Event Register

The XFI Link Training Interrupt Event Register contains interrupt event bits related to link training, as shown in Figure 19-91

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R													TRAIN_DONE	TRAIN_FAIL	TRAIN_FRM_RCV	TRAIN_LOCK
W																
Special Access													RC	RC	RC	RC
Reset	0000_0000_0000_0000															
Offset	0x0006															

Figure 19-91. XFI Link Training Interrupt Event Register

Table 19-98. XFI Link Training Interrupt Event Register Field Descriptions

Bits	Name	Description
15-4	—	Reserved
3	TRAIN_DONE	Link Training Completed 0: Link training not complete 1: Link training complete Cleared on register read
2	TRAIN_FAIL	Link Taining Fault Indication 0: Link training successful or not complete 1: Link training failed Cleared on register read
1	TRAIN_FRM_RCV	Link Training Frame Receive Indication 0: New coefficient update and status fields not available 1: New coefficient update and status fields available Cleared on register read Note: during link training, training frames are received continuously, leading to assertion of this event every ~420 ns.
0	TRAIN_LOCK	Link Training Lock Indication 0: Link training state machine is not locked 1: Link training state machine is locked Cleared on register read

19.7.2.4.8 XFI Link Training Interrupt Mask Register

The XFI Link Training Interrupt Mask Register contains the mask bits for the above link training interrupt events, as shown in Figure 19-92

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	—												TRAIN_DONE_EN	TRAIN_FAIL_EN	TRAIN_FRM_RCV_EN	TRAIN_LOCK_EN
Special Access																
Reset	0000_0000_0000_0000															
Offset	0x0007															

Figure 19-92. XFI Link Training Interrupt Mask Register

Table 19-99. XFI Link Training Interrupt Mask Register Field Descriptions

Bits	Name	Description
15-5	—	Reserved
3	TRAIN_DONE_EN	Link Training Completed Interrupt Enable 0: Link Training Complete will not cause an interrupt event 1: Link Training Complete will cause an interrupt event
2	TRAIN_FAIL_EN	Link Taining Fault Indication Interrupt Enable 0: Link Taining Fault Indication will not cause an interrupt event 1: Link Taining Fault Indication will cause an interrupt event
1	TRAIN_FRM_RCV_EN	Link Training Frame Receive Indication Interrupt Enable 0: Link Training Frame Receive Indication will not cause an interrupt event 1: Link Training Frame Receive Indication will cause an interrupt event
0	TRAIN_LOCK_EN	Link Training Lock Indication Interrupt Enable 0: Link Training Lock Indication will not cause an interrupt event 1: Link Training Lock Indication will cause an interrupt event

19.7.2.5 SGMII Registers

The SGMII/ MDIO space implements registers for IEEE 802.3 Clause 22 and other function in the SGMII/ PCSs.

Each PCS contains 4 SGMII PCSs, so there are four copies of the SGMII management registers in the PCS, one per port.

In 1000Base-KX operation, these same registers are available via Clause 45 access in the PCS MMD (0x03), at the same relative offset starting at 0x8000.

19.7.2.5.1 SGMII Control Register

The SGMII Control Register contains control bits used to enable/disable functions in the PCS, and to initiate commands such as reset, as shown in Figure 19-93.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RST	LPBK	SPEED_SEL0	AN_EN	PD	ISOLATE	RESTART_AN	FD	COL_TEST	SPEED_SEL1	—					
W																
Special Access	SC						SC									
Reset	0001_0001_0100_0000															
Offset	0x00 (Clause 22) 0x8000 (Clause 45)															

Figure 19-93. SGMII Control Register

Table 19-100. SGMII Control Register Field Descriptions

Bits	Name	Description
15	RST	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 Command. 0: Normal operation 1: Reserved
13	SPEED_SEL0	Speed selection (LSB) MSB,LSB 10: 1000 Mb/s All others reserved Note: SGMII speed is controlled with the IF Mode register
12	AN_EN	Auto-negotiation enable 0: Disable auto-negotiation 1: Enable auto-negotiation
11	PD	Power down 0: Normal operation 1: Power down PCS. The PCS is held in internal soft reset until the bit is cleared again.
10	ISOLATE	Isolate 0: Normal operation 1: Reserved

Table 19-100. SGMII Control Register Field Descriptions

9	RESTART_AN	Restart auto-negotiation 0: Normal operation 1: Restart auto-negotiation This bit is self-clearing
8	FD	Duplex mode. Read-only 0: Reserved 1: Full duplex
7	COL_TEST	Collision test. Read-only 0: Disable COL signal test 1: Reserved
6	SPEED_SEL1	Speed selection (MSB) See SPEED_SEL0
5-0	—	Reserved

19.7.2.5.2 SGMII Status Register

The SGMII Status Register contains status bits on the operation of the PCS, as shown in Figure 19-94.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	100BASET4	100BASEX_FD	100BASEX_HD	10MBPS_FD	10MBPS_HD	100BASET2_FD	100BASET2_HD	EXT_STAT	—		AN_COMP	REM_FAULT	AN_ABIL	LINK_STAT	JAB_DET	EXT_CAP
W																
Special Access														LL		
Reset	0000_0000_0000_1001															
Offset	0x01 (Clause 22) 0x8001 (Clause 45)															

Figure 19-94. SGMII Status Register

Table 19-101. SGMII Status Register Field Descriptions

Bits	Name	Description
15	100BASET4	Read Only bit set to 0 to indicate that the PCS does not support 100Base-T4 operation
14	100BASEX_FD	Read Only bit set to 0 to indicate that the PCS does not support 100Base-X operation
13	100BASEX_HD	Read Only bit set to 0 to indicate that the PCS does not support 100Base-X operation
12	10MBPS_FD	Read Only bit set to 0 to indicate that the PCS does not support 10Mbps operation
11	10MBPS_HD	Read Only bit set to 0 to indicate that the PCS does not support 10Mbps operation
10	100BASET2_FD	Read Only bit set to 0 to indicate that the PCS does not support 100Base-T2 operation.
9	100BASET2_HD	Read Only bit set to 0 to indicate that the PCS does not support 100Base-T2 operation.
8	EXT_STAT	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	Auto-negotiation complete. Read Only Bit 0: The Auto Negotiation process is not completed or Auto Negotiation is disabled 1: The Auto Negotiation process is completed and that the Auto Negotiation control registers are valid.

Table 19-101. SGMII Status Register Field Descriptions

4	REM_FAULT	Read Only Bit always set to 0. The PCS does not implement a PHY specific remote fault detection optional function.
3	AN_ABIL	Auto Negotiation Ability. Read Only Bit set to „1. to indicate that the PCS supports Auto-Negotiation.
2	LINK_STAT	Link Status 0: The link is not valid. If the link synchronization is lost a 0 is latched which is cleared only after a register read access 1: This link is valid.
1	JAB_DET	Read Only bit always set to 0, the Core does not support the optional Jabber detection function
0	EXT_CAP	Read Only bit set to „1. to indicate that the Core supports extended registers

19.7.2.5.3 SGMII PHY Identifier Upper Register

The SGMII PHY Identifier Upper Register contains the upper half of the 32-bit PHY Identifier, as shown in Figure 19-95.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PHY_IDENTIFIER_U															
Special Access																
Reset	0000_0000_1000_0011															
Offset	0x02 (Clause 22) 0x8002 (Clause 45)															

Figure 19-95. SGMII PHY Identifier Upper Register

Table 19-102. SGMII PHY Identifier Upper Register Field Descriptions

Bits	Name	Description
15-0	PHY_IDENTIFIER_U	PHY Identifier Upper: OUI[3:18]

19.7.2.5.4 SGMII PHY Identifier Lower Register

The SGMII PHY Identifier Lower Register contains the lower half of the 32-bit PHY Identifier, as shown in Figure 19-96.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PHY_IDENTIFIER_L															
Special Access																
Reset	1110_0100_0000_0000															
Offset	0x03 (Clause 22) 0x8003 (Clause 45)															

Figure 19-96. SGMII PHY Identifier Lower Register

Table 19-103. SGMII PHY Identifier Lower Register Field Descriptions

Bits	Name	Description
15-0	PHY_IDENTIFIER_L	PHY Identifier Lower: 15:10 OUI[19:24] 9:4 Manufacturer's Model Number 3:0 Revision Number

19.7.2.5.5 SGMII Device Ability Register (SGMII Use)

The SGMII Device Ability Register contains the control bits used to advertise the device abilities to the Link Partner during auto-negotiation, as shown in Figure 19-97 for SGMII mode.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	—	ACK	—					EEE_CLK_STP_EN	—							SGMII
Special Access																
Reset	0000_0001_1010_0400															
Offset	0x04 (Clause 22) 0x8004 (Clause 45)															

Figure 19-97. SGMII Device Ability Register (SGMII)

Table 19-104. SGMII Device Ability Field Descriptions (SGMII)

Bits	Name	Description
15	—	Reserved
14	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
9	EEE_CLK_STP_EN	EEE Clock Stop Enable 0: EEE Clock Stop Disabled 1: EEE Clock Stop Enabled Should be set to 0 before SGMII Auto-Negotiation enable/restart, as this device does not support EEE clock stop.
8-1	—	Reserved. Should be set to 0
0	SGMII	SGMII mode. Should be set to 1.

19.7.2.5.6 SGMII Device Ability Register (1000Base-X)

The SGMII Device Ability Register contains the control bits used to advertise the device abilities to the Link Partner during auto-negotiation, as shown in Figure 19-98 for 1000Base-X mode.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	NP	ACK	RF2	RF1	—			PS2	PS1	HD	FD	—				
W																
Special Access																
Reset	0000_0001_1010_0000															
Offset	0x04 (Clause 22) 0x8004 (Clause 45)															

Figure 19-98. SGMII Device Ability Register (1000Base-X)

Table 19-105. SGMII Device Ability Field Descriptions (1000Base-X)

Bits	Name	Description
15	NP	Next page support. Set to 1 to indicate next page can be transferred following base page exchange
14	ACK	Read only bit set to 1 when device has received three consecutive matching ability values from the link partner.
13	RF2	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	See RF2
11-9	—	Reserved

Table 19-105. SGMII Device Ability Field Descriptions (1000Base-X)

8	PS2	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	See PS2
6	HD	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 Enable. Set to 1 when the device advertises that it supports Full Duplex Mode of operation
4-0	—	Reserved

19.7.2.5.7 SGMII Partner Ability Register (SGMII)

The SGMII Partner Ability Register contains the capability status of the SGMII link parner, as shown in Figure 19-99.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	COP_LNK_STAT	ACK	—	COP_DUPL	COP_SPD		EEE_CAP	EEE_CLK_STOP_CAP	—							
W																
Special Access																
Reset	0000_0000_0000_0000															
Offset	0x05 (Clause 22) 0x8005 (Clause 45)															

Figure 19-99. SGMII Partner Ability Register

Table 19-106. SGMII Partner Ability Field Descriptions (SGMII)

Bits	Name	Description
15	COP_LNK_STAT	Read only bit, used by the SGMII PHY to advertise the Link Partner Copper status: 1: Copper interface link is up 0: Copper interface link is down
14	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

Table 19-106. SGMII Partner Ability Field Descriptions (SGMII)

12	COP_DUPL	Read only bit, used by the SGMII PHY to advertise the Link Copper duplex capability: 0: Copper Interface resolved to Half-Duplex 1: Copper Interface resolved to Full-Duplex
11-10	COP_SPD	Read only bits, used by the SGMII PHY to advertise the Link Partner Copper interface speed 00: Copper Interface Speed is 10Mbps 01: Copper Interface Speed is 100Mbps 10: Copper Interface Speed is Gigabit 11: Reserved
9	EEE_CAP	EEE Capability 0: EEE not supported 1: EEE supported
8	EEE_CLK_STOP_CAP	EEE Clock Stop Capability 0: EEE Clock Stop not supported 1: EEE Clock Stop supported
7-0	—	Reserved

19.7.2.5.8 SGMII Partner Ability Register (1000Base-X)

The SGMII Partner Ability Register contains the bits advertised by the Link Partner during auto-negotiation, as shown in Figure 19-100 for 1000Base-X mode.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	NP	ACK	RF2	RF1	—			PS2	PS1	HD	FD	—				
W																
Special Access																
Reset	0000_0000_0000_0000															
Offset	0x05 (Clause 22) 0x8005 (Clause 45)															

Figure 19-100. SGMII Partner Ability Register (1000Base-X)

Table 19-107. SGMII Partner Ability Field Descriptions (1000Base-X)

Bits	Name	Description
15	NP	Next page support.
14	ACK	Set to 1 when link partner has received three consecutive matching ability values from the device.
13	RF2	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	See RF2
11-9	—	Reserved

Table 19-107. SGMII Partner Ability Field Descriptions (1000Base-X)

8	PS2	Pause capability of link partner (ASM_DIR:PAUSE) 00: No Pause 11: Symmetric Pause 10: Asymmetric Pause toward link partner 11: Both Symmetric Pause and Asymmetric Pause toward local device
7	PS1	See PS2
6	HD	Half Duplex support
5	FD	Full Duplex support
4-0	—	Reserved

19.7.2.5.9 SGMII AN Expansion Register

The SGMII AN Expansion Register contains status bits indicating the PCS next page auto-negotiation capability and status, as shown in Figure 19-101.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	—													NP_ABLE	PG_RCV	—
W																
Special Access															RC	
Reset	0000_0000_0000_0100															
Offset	0x06 (Clause 22) 0x8006 (Clause 45)															

Figure 19-101. SGMII AN Expansion Register

Table 19-108. SGMII AN Expansion Register Field Descriptions

Bits	Name	Description
15-3	—	Reserved
2	NP_ABLE	Read Only bit set to 1 to indicate the PCS does support the Next Page function
1	PG_RCV	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

19.7.2.5.10 SGMII NP TX Register

The SGMII NP TX Register contains next page data to transfer to the remote devide, as shown in Figure 19-102. Writing to this register initiates a next page exchange (sets mr_np_loaded variable).

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	NP	ACK	MP	ACK2	TOG GLE	DATA										
Special Access																
Reset	0000_0000_0000_0000															
Offset	0x07 (Clause 22) 0x8007 (Clause 45)															

Figure 19-102. SGMII NP TX Register

Table 19-109. SGMII NP TX Field Descriptions

Bits	Name	Description
15	NP	Next page indication. Indicates if additional next pages will follow (1) or this is the last page (0).
14	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 (1) or unformatted page (0) format.
12	ACK2	Application level acknowledge to indicate acceptance of a message. Use is defined by application layer.
11	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	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

19.7.2.5.11 SGMII NP RX Register

The SGMII NP RX Register contains the next page data received from the remote device during the latest next page exchange, as shown in Figure 19-103 The value is overridden with every new next page. Next page transfers are controlled by the application.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	NP	ACK	MP	ACK2	TOG GLE	DATA										
W																
Special Access																
Reset	0000_0000_0000_0000															
Offset	0x08 (Clause 22) 0x8008 (Clause 45)															

Figure 19-103. SGMII NP RX Register

Table 19-110. SGMII NP RX Register Field Descriptions

Bits	Name	Description
15	NP	Next page indication. Indicates if additional next pages will follow (1) or this is the last page (0).
14	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 (1) or unformatted page (0) format.
12	ACK2	Application level acknowledge to indicate acceptance of a message. Use is defined by application layer.
11	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	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

19.7.2.5.12 SGMII Extended Status Register

The SGMII Extended Status Register is reserved, as shown in Figure 19-104, as this device does not implement the optional extended status registers.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	—															
W																
Special Access																
Reset	0000_0000_0000_0000															
Offset	0x0F (Clause 22) 0x800F (Clause 45)															

Figure 19-104. SGMII Extended Status Register

Table 19-111. SGMII Extended Status Register Field Descriptions

Bits	Name	Description
15-0	—	Reserved. Always 0

19.7.2.5.13 SGMII Scratch Register

The SGMII Scratch Register provides a memory location available to test register read and write operations.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SCRATCH															
W																
Special Access																
Reset	0000_0000_0000_0000															
Offset	0x10 (Clause 22) 0x8010 (Clause 45)															

Figure 19-105. SGMII Scratch Register

Table 19-112. SGMII Scratch Register Field Descriptions

Bits	Name	Description
15-0	SCRATCH	Scratch field.

19.7.2.5.14 SGMII Design Revision Register

The SGMII Revision Register contains revision information for the PCS.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	—								IP_MJ				IP_MN			
W																
Special Access																
Reset	0000_0000_0000_0001															
Offset	0x11 (Clause 22) 0x8011 (Clause 45)															

Figure 19-106. SGMII Revision Register

Table 19-113. SGMII Revision Register Field Descriptions

Bits	Name	Description
15-8	—	Reserved
7-4	IP_MJ	Major revision
3-0	IP_MN	Minor revision

19.7.2.5.15 SGMII Link Timer Register Lower

The SGMII Link Timer Register Lower contains bits 15:1 of the link timer value.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LINK_TMR_L															—
W																
Special Access																
Reset	0001_0020_1101_0000															
Offset	0x12 (Clause 22) 0x8012 (Clause 45)															

Figure 19-107. SGMII Link Timer Register Lower

Table 19-114. SGMII Link Timer Register Lower Field Descriptions

Bits	Name	Description
15-1	LINK_TMR_L	Link timer[15:1]
0	—	Reserved

19.7.2.5.16 SGMII Link Timer Register Upper

The SGMII Link Timer Register Upper contains bits 20:16 of the link timer value.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0	0	0	0	LINK_TMR_U				
W																
Special Access																
Reset	0000_0000_0001_0011															
Offset	0x13 (Clause 22) 0x8013 (Clause 45)															

Figure 19-108. SGMII Link Timer Register Upper

Table 19-115. SGMII Control Register Field Descriptions

Bits	Name	Description
15-5	—	Reserved
4-0	LINK_TMR_U	Link timer[20:16]

19.7.2.5.17 SGMII IF Mode Register

The SGMII IF Mode Register contains control bits to set the interface mode.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W												SGMII_DUPLEX	SGMII_SPEED		USE_SGMII_AN	SGMII_EN
Special Access																
Reset	0000_0000_0000_0000															
Offset	0x14 (Clause 22) 0x8014 (Clause 45)															

Figure 19-109. SGMII IF Mode Register

Table 19-116. SGMII IF Mode Register Field Descriptions

Bits	Name	Description
15-5	—	Reserved
4	SGMII_DUPLEX	SGMII Halfduplex Mode: 0: Full duplex enabled (default) 1: Half duplex enabled Bit ignored when SGMII_EN is set to 0 or USE_SGMII_AN is set to 1.
3-2	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: 00: 10Mbps 01: 100Mbps 10: Gigabit 11: Reserved Bits ignored when SGMII_EN is set to 0 or USE_SGMII_AN is set to 1.
1	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 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

19.7.2.6 1000Base-KX PCS Registers

The 1000Base-KX MDIO PCS MMD 3 implements registers for IEEE 802.3 Clause 45 and other function in the SGMII PCSs.

19.7.2.6.1 1000Base-KX PCS Control Register

The 1000Base-KX PCS Control Register contains control bits used to enable/disable functions in the PCS, and to initiate commands such as reset, as shown in Figure 19-110.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RST	LPBK	SPEED_SEL0	—	PD	—				SPEED_SEL1	0	0	0	0	0	0
W																
Special Access	SC			WC												
Reset	0000_0001_0100_0000															
Offset	0x0000															

Figure 19-110. KX PCS Control Register

Table 19-117. KX PCS Control Register Field Descriptions

Bits	Name	Description
15	RST	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 Command. 0: Normal operation 1: Reserved
13	SPEED_SEL0	Speed selection (LSB) MSB,LSB 10: 1000 Mb/s All others reserved Note: SGMII speed is controlled with the IF Mode register
12	—	Reserved
11	PD	Power down 0: Normal operation 1: 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

19.7.2.6.2 1000Base-KX PCS Status Register

The 1000Base-KX PCS Status Register contains status bits on the operation of the PCS, as shown in Figure 19-111.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	—													LINK_STAT	—	
W																
R	0	0	0	0	0	0	0	0	0	0	0	0	1	LINK_STAT	0	1
W																
Special Access														LL		
Reset	0000_0000_0000_1001															
Offset	0x0001															

Figure 19-111. KX PCS Status Register

Table 19-118. KX PCS Status Register Field Descriptions

Bits	Name	Description
15-3	—	Reserved
2	LINK_STAT	Link Status 0: The link is not valid. If the link synchronization is lost a 0 is latched which is cleared only after a register read access 1: This link is valid.
1-0	—	Reserved

19.7.2.6.3 1000Base-KX PCS Device Identifier Upper Register

The 1000Base-KX PCS Device Identifier Upper Register contains the upper half of the 32-bit Device Identifier, as shown in Figure 19-112

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PCS_DEV_ID_U															
W																
Special Access																
Reset	0000_0000_1000_0011															
Offset	0x0002															

Figure 19-112. KX PCS Device Identifier Upper Register

Table 19-119. KX PCS Device Identifier Upper Register Field Descriptions

Bits	Name	Description
15-0	PCS_DEV_ID_U	PCS Device Identifier Upper: OUI[3:18]

19.7.2.6.4 1000Base-KX PCS Device Identifier Lower Register

The 1000Base-KX PCS Identifier Lower Register contains the lower half of the 32-bit Device Identifier, as shown in Figure 19-113.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PCS_DEV_ID_L															
W																
Special Access																
Reset	1110_0100_0000_0000															
Offset	0x0003															

Figure 19-113. KX PCS Device Identifier Lower Register

Table 19-120. KX PCS Device Identifier Lower Register Field Descriptions

Bits	Name	Description
15-0	PCS_DEV_ID_L	PCS Device Identifier Lower: 15:10 OUI[19:24] 9:4 Manufacturer's Model Number 3:0 Revision Number

19.7.2.6.5 1000Base-KX PCS Devices in Package 0 Register

The 1000Base-KX PCS Devices in Package 0 Register contains half of the PCS devices in package status, as shown in Figure 19-114

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	—								AN_PRES	TC_PRES	DTE_XS_PRES	PHY_XS_PRES	PCS_PRES	WIS_PRES	PMD_PMA_PRES	CLAUSE_22_REG_PRES
W																
Special Access																
Reset	0000_0000_1000_1000															
Offset	0x0005															

Figure 19-114. KX PCS Devices in Package 0 Register

Table 19-121. KX PCS Devices in Package 0 Register Field Descriptions

Bits	Name	Description
15-8	—	Reserved
7	AN_PRES	1 = Auto-Negotiation present in package 0 = Auto-Negotiation not present in package
6	TC_PRES	1 = TC present in package 0 = TC not present in package
5	DTE_XS_PRES	1 = DTE XS present in package 0 = DTE XS not present in package
4	PHY_XS_PRES	1 = PHY XS present in package 0 = PHY XS not present in package
3	PCS_PRES	1 = PCS present in package 0 = PCS not present in package
2	WIS_PRES	1 = WIS present in package 0 = WIS not present in package
1	PMD_PMA_PRES	1 = PMA/PMD present in package 0 = PMA/PMD not present in package
0	CLAUSE_22_REG_PRES	1 = Clause 22 registers present in package 0 = Clause 22 registers not present in package

19.7.2.6.6 1000Base-KX PCS Devices in Package 1 Register

The 1000Base-KX PCS Devices in Package 1 Register contains half of the PCS devices in package status, as shown in Figure 19-115

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	VEND_SPEC_DEV2_PRES	VEND_SPEC_DEV1_PRES	CLAUSE22_EXT_PRES	—												
W																
Special Access																
Reset	0000_0000_0000_0000															
Offset	0x0006															

Figure 19-115. KX PCS Devices in Package 1 Register

Table 19-122. KX PCS Devices in Package 0 Register Field Descriptions

Bits	Name	Description
15	VEND_SPEC_DEV2_PRES	1 = Vendor specific device 2 present in package 0 = Vendor specific device 2 not present in package
14	VEND_SPEC_DEV1_PRES	1 = Vendor specific device 1 present in package 0 = Vendor specific device 1 not present in package
13	CLAUSE22_EXT_PRES	1 = Clause 22 extension present in package 0 = Clause 22 extension not present in package
12-0	—	Reserved

19.7.2.6.7 1000Base-KX PCS Package Identifier Upper Register

The 1000Base-KX PCS Package Device Identifier Upper Register contains the upper half of the 32-bit Package Identifier, as shown in Figure 19-116

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PKG_DEV_ID_U															
W																
Special Access																
Reset	0000_0000_1000_0011															
Offset	0x000E															

Figure 19-116. KX PCS Package Device Identifier Upper Register

Table 19-123. KX PCS Package Device Identifier Upper Register Field Descriptions

Bits	Name	Description
15-0	PKG_DEV_ID_U	Package Device Identifier Upper: OUI[3:18]

19.7.2.6.8 1000Base-KX PCS Package Device Identifier Lower Register

The KX PCS Package Identifier Lower Register contains the lower half of the 32-bit Package Identifier, as shown in Figure 19-117.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PKG_DEV_ID_L															
W																
Special Access																
Reset	1110_0100_0000_0000															
Offset	0x000F															

Figure 19-117. KX PCS Package Device Identifier Lower Register

Table 19-124. KX PCS Package Identifier Lower Register Field Descriptions

Bits	Name	Description
15-0	PKG_DEV_ID_L	Package Device Identifier Lower: 15:10 OUI[19:24] 9:4 Manufacturer's Model Number 3:0 Revision Number

19.7.2.7 1000Base-KX Auto-Negotiate Registers

The 1000Base-KX MDIO space implements registers for IEEE 802.3 Clause 45 and other function in the SGMII PCSs.

19.7.2.7.1 1000Base-KX AN Control Register

The 1000Base-KX AN Control Register contains control bits used to enable/disable functions in the PCS, and to initiate commands such as reset, as shown in Figure 19-118.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	AN_RST	—	EXT_NP_CTRL	BP_AN_EN	—		RESTART_BP_AN	—								
Special Access	SC						SC									
Reset	0000_0000_0000_0000															
Offset	0x0000															

Figure 19-118. 1000Base-KX AN Control Register

Table 19-125. 1000Base-KX AN Control Register Field Descriptions

Bits	Name	Description
15	AN_RST	AN reset 0: AN normal operation 1: AN reset This bit is self-clearing
14	—	Reserved
13	EXT_NP_CTRL	Extended Next Page Control 0: Extended next pages are disabled 1: Extended next pages are enabled 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.
12	BP_AN_EN	Backplane auto-negotiation enable 0: Disable auto-negotiation 1: Enable auto-negotiation
11-10	—	Reserved
9	RESTART_BP_AN	Restart backplane auto-negotiation 0 = Auto-Negotiation in process, disabled, or not supported 1 = Restart Auto-Negotiation process This bit is self-clearing
8-0	—	Reserved

19.7.2.7.2 1000Base-KX AN Status Register

The 1000Base-KX AN Status Register contains status bits for the 1000Base-KX auto-negotiate function, as shown in Figure 19-119

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	—						PAR_DET_FAULT	—	EXT_NP_STAT	PAGE_RCV	AN_COMP	REM_FAULT	AN_ABIL	LNK_STAT	—	LP_AN_ABIL
W																
Special Access							LH			LH		LH		LL		
Reset	0000_0000_0000_1000															
Offset	0x0001															

Figure 19-119. 1000Base-KX AN Status Register

Table 19-126. 1000Base-KX AN Status Register Field Descriptions

Bits	Name	Description
15-10	—	Reserved
9	PAR_DET_FAULT	Parallel Detection Fault 0: A fault has not been detected via the parallel detection function. 1: A fault has been detected via the parallel detection function. Once set, stays set until the register is read.
8	—	Reserved
7	EXT_NP_STAT	Extended Next Page Status 0: Extended next pages disabled 1: Extended next pages enabled This bit is a copy of AN Control[EXT_NP_CTRL]
6	PAGE_RCV	Page received 0: Page has not been received 1: Page has been received Once set, stays set until the register is read
5	AN_COMP	Auto-negotiation complete 0: Auto-negotiation process not completed 1: Auto-negotiation process completed

Table 19-126. 1000Base-KX AN Status Register Field Descriptions

4	REM_FAULT	Remote Fault 0: No remote fault condition detected 1: Remote fault condition detected Once set, stays set until the register is read
3	AN_ABIL	Auto-negotiation Ability Always set to 1 to indicate that PCS is able to perform auto-negotiation
2	LNK_STAT	Link Status 0: Link is down 1: Link is up Once cleared, stays cleared until the register is read
1	—	Reserved
0	LP_AN_ABIL	Link Partner Auto-Negotiation Ability 0: Link Partner is not able to perform auto-negotiation, or Link Partner ability not yet captured 1: Link Parner is able to perform auto-negotiation

19.7.2.7.3 1000Base-KX AN Device Identifier Upper Register

The 1000Base-KX AN Device Identifier Upper Register contains the upper half of the 32-bit device identifier, as shown in Figure 19-120

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	AN_DEV_ID_U															
W																
Special Access																
Reset	0000_0000_1000_0011															
Offset	0x0002															

Figure 19-120. 1000Base-KX AN Device Identifier Upper Register

Table 19-127. KX AN Device Identifier Upper Register Field Descriptions

Bits	Name	Description
15-0	AN_DEV_ID_U	PCS Device Identifier Upper: OUI[3:18]

19.7.2.7.4 1000Base-KX AN Device Identifier Lower Register

The 1000Base-KX AN Device Identifier Lower Register contains the lower half of the 32-bit Device Identifier, as shown in Figure 19-121.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	AN_DEV_ID_L															
W																
Special Access																
Reset	1110_0100_0000_0000															
Offset	0x0003															

Figure 19-121. KX AN Device Identifier Lower Register

Table 19-128. KX AN Device Identifier Lower Register Field Descriptions

Bits	Name	Description
15-0	AN_DEV_ID_L	PCS Device Identifier Lower: 15:10 OUI[19:24] 9:4 Manufacturer's Model Number 3:0 Revision Number

19.7.2.7.5 1000Base-KX AN Devices in Package 0 Register

The 1000Base-KX AN Devices in Package 0 Register contains half of the AN devices in package status, as shown in Figure 19-122

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	—								AN_PRES	TC_PRES	DTE_XS_PRES	PHY_XS_PRES	PCS_PRES	WIS_PRES	PMD_PMA_PRES	CLAUSE_22_REG_PRES
W																
Special Access																
Reset	0000_0000_1000_1000															
Offset	0x0005															

Figure 19-122. KX AN Devices in Package 0 Register

Table 19-129. KX AN Devices in Package 0 Register Field Descriptions

Bits	Name	Description
15-8	—	Reserved
7	AN_PRES	1 = Auto-Negotiation present in package 0 = Auto-Negotiation not present in package
6	TC_PRES	1 = TC present in package 0 = TC not present in package
5	DTE_XS_PRES	1 = DTE XS present in package 0 = DTE XS not present in package
4	PHY_XS_PRES	1 = PHY XS present in package 0 = PHY XS not present in package
3	PCS_PRES	1 = PCS present in package 0 = PCS not present in package
2	WIS_PRES	1 = WIS present in package 0 = WIS not present in package
1	PMD_PMA_PRES	1 = PMA/PMD present in package 0 = PMA/PMD not present in package
0	CLAUSE_22_REG_PRES	1 = Clause 22 registers present in package 0 = Clause 22 registers not present in package

19.7.2.7.6 1000Base-KX AN Devices in Package 1 Register

The 1000Base-KX AN Devices in Package 1 Register contains half of the AN devices in package status, as shown in Figure 19-123

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	VEND_SPEC_DEV2_PRES	VEND_SPEC_DEV1_PRES	CLAUSE22_EXT_PRES	—												
W																
Special Access																
Reset	0000_0000_0000_0000															
Offset	0x0006															

Figure 19-123. KX AN Devices in Package 1 Register

Table 19-130. KX AN Devices in Package 0 Register Field Descriptions

Bits	Name	Description
15	VEND_SPEC_DEV2_PRES	1 = Vendor specific device 2 present in package 0 = Vendor specific device 2 not present in package
14	VEND_SPEC_DEV1_PRES	1 = Vendor specific device 1 present in package 0 = Vendor specific device 1 not present in package
13	CLAUSE22_EXT_PRES	1 = Clause 22 extension present in package 0 = Clause 22 extension not present in package
12-0	—	Reserved

19.7.2.7.7 1000Base-KX AN Package Identifier Upper Register

The 1000Base-KX AN Package Device Identifier Upper Register contains the upper half of the 32-bit Package Identifier, as shown in Figure 19-124

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PKG_DEV_ID_U															
W																
Special Access																
Reset	0000_0000_1000_0011															
Offset	0x000E															

Figure 19-124. KX AN Package Device Identifier Upper Register

Table 19-131. KX AN Package Device Identifier Upper Register Field Descriptions

Bits	Name	Description
15-0	PKG_DEV_ID_U	Package Device Identifier Upper: OUI[3:18]

19.7.2.7.8 1000Base-KX AN Package Device Identifier Lower Register

The KX AN Package Identifier Lower Register contains the lower half of the 32-bit Package Identifier, as shown in Figure 19-125.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PKG_DEV_ID_L															
W																
Special Access																
Reset	1110_0100_0000_0000															
Offset	0x000F															

Figure 19-125. KX AN Package Device Identifier Lower Register

Table 19-132. KX AN Package Identifier Lower Register Field Descriptions

Bits	Name	Description
15-0	PKG_DEV_ID_L	Package Device Identifier Lower: 15:10 OUI[19:24] 9:4 Manufacturer's Model Number 3:0 Revision Number

19.7.2.7.9 1000Base-KX AN Advertisement Register 0

The 1000Base-KX AN Advertisement Register 0 contains bits 15:0 of the 48-bit page bits used during base page exchange, as shown in Figure 19-126 They should be programmed before auto-negotiation is started. The 48-bit value is distributed over registers 0, 1, 2.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	NP	ACK	REM_FAULT	XNP_CAP	PAUSE_CAP		ECHOED_NONCE						SELECT_FLD			
W																
Special Access																
Reset	0000_1100_0000_0001															
Offset	0x0010															

Figure 19-126. KX AN Advertisement Register 0

Table 19-133. KX AN Advertisement Register 0 Field Descriptions

Bits	Name	Description
15	NP	Next Page 0: The device has no next pages to send 1: The device has next pages to send
14	ACK	Acknowledge Should always be set to 0
13	REM_FAULT	Remote Fault 0: No remote fault advertised 1: Advertise remote fault
12	XNP_CAP	Extended Next Page Capability 0: Device does not support extended next pages
11-10	PAUSE_CAP	Pause Capability (ASM_DIR:PAUSE) 00: No PAUSE 01: Symmetric PAUSE 10: Asymmetric PAUSE toward link partner 11: Both Symmetric PAUSE and Asymmetric PAUSE toward local device Default is 11
9-5	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 (S4:0) 00001: IEEE Std 802.3 All other values reserved

19.7.2.7.10 1000Base-KX AN Advertisement Register 1

The 1000Base-KX AN Advertisement Register 1 contains bits 31:16 of the 48-bit page bits used during base page exchange, as shown in Figure 19-127 They should be programmed before auto-negotiation is started. The 48-bit value is distributed over registers 0, 1, 2.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TECH_A10_A0											TRANS_NONCE				
W																
Special Access																
Reset	0000_0000_0010_1111															
Offset	0x0011															

Figure 19-127. KX AN Advertisement Register 1

Table 19-134. KX AN Advertisement Register 1 Field Descriptions

Bits	Name	Description
15-5	TECH_A10_A0	Bits A10:A0 of the technology field. 5 (A0): 1000Base-KX 6 (A1): 10GBase-KX4 7 (A2): 10GBase-KR 8 (A3): 40GBase-KR4 9 (A4): 40GBase-CR4 10 (A5): 100GBase-CR 11 (A6): reserved 12 (A7): reserved 13 (A8): reserved 14 (A9): reserved 15 (A10): reserved Must set all bits except A0 to 0
4-0	TRANS_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.

19.7.2.7.11 1000Base-KX AN Advertisement Register 2

The 1000Base-KX AN Advertisement Register 0 contains bits 47:32 of the 48-bit page bits used during base page exchange, as shown in Figure 19-128. 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.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FEC_CAP		TECH_A24_A11													
W																
Special Access																
Reset	0000_0000_0000_0000															
Offset	0x0012															

Figure 19-128. KX AN Advertisement Register 2

Table 19-135. KX AN Advertisement Register 2 Field Descriptions

Bits	Name	Description
15-14	FEC_CAP	Capability bits (F1:0) for FEC support 00: PCS does not implement a FEC function
13-0	TECH_A24_A11	Technology field A24:11 All bits reserved. Should be set to 0

19.7.2.7.12 1000Base-KX AN LP Base Page Ability Register 0

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.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	NP	ACK	REM_FAULT	XNP_CAP	PAUSE_CAP		ECHOED_NONCE						SELECT_FLD			
W																
Special Access																
Reset	0000_0000_0000_0000															
Offset	0x0013															

Figure 19-129. KX AN LP Base Page Ability Register 0

Table 19-136. KX AN LP Base Bage Ability Register 0 Field Descriptions

Bits	Name	Description
15	NP	Next Page 0: The link partner has no next pages to send 1: The link partner has next pages to send
14	ACK	Acknowledge 0: No link partner acknowledge 1: Link partner acknowledge
13	REM_FAULT	Remote Fault 0: No remote fault advertised by link partner 1: Remote fault advertised by link partner
12	XNP_CAP	Extended Next Page Capability 0: Link partner does not support extended next pages 1: Link parnet supports extended next pages
11-10	PAUSE_CAP	Pause Capability (ASM_DIR:PAUSE) 00: No PAUSE 01: Symmetric PAUSE 10: Asymmetric PAUSE toward local device 11: Both Symmetric PAUSE and Asymmetric PAUSE toward link partner
9-5	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 (S4:0) 00001: IEEE Std 802.3 All other values reserved

19.7.2.7.13 1000Base-KX AN LP Base Page Ability Register 1

The 1000Base-KX AN LP Base Page Ability Register 1 contains bits 31:16 of the link partner base storage, as shown in Figure 19-130. The 48-bit value is distributed over registers 0, 1, 2.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TECH_A10_A0											TRANS_NONCE				
W																
Special Access																
Reset	0000_0000_0000_000															
Offset	0x0014															

Figure 19-130. KX AN LP Base Page Ability Register 1

Table 19-137. KX AN LP Base Page Ability Register 1 Field Descriptions

Bits	Name	Description
15-5	TECH_A10_A0	Bits A10:A0 of the technology field. 5 (A0): 1000Base-KX 6 (A1): 10GBase-KX4 7 (A2): 10GBase-KR 8 (A3): 40GBase-KR4 9 (A4): 40GBase-CR4 10 (A5): 100GBase-CR 11 (A6): reserved 12 (A7): reserved 13 (A8): reserved 14 (A9): reserved 15 (A10):reserved
4-0	TRANS_NONCE	Transmitted Nonce

19.7.2.7.14 1000Base-KX AN LP Base Page Ability Register 2

The 1000Base-KX AN LP Base Page Ability Register 2 contains bits 47:32 of the link partner base storage, as shown in Figure 19-131. The 48-bit value is distributed over registers 0, 1, 2.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FEC_CAP		TECH_A24_A11													
W																
Special Access																
Reset	0000_0000_0000_0000															
Offset	0x0015															

Figure 19-131. KX AN LP Base Page Ability Register 2

Table 19-138. KX AN LP Base Page Ability Register 2 Field Descriptions

Bits	Name	Description
15-14	FEC_CAP	Capability bits (F1:0) for FEC support F1: FEC ability F0: FEC requested
13-0	TECH_A24_A11	Technology field A24:11 Reserved for future technology

19.7.2.7.15 1000Base-KX AN XNP Transmit Register 0

The 1000Base-KX AN XNP Transmit Register 0 contains bits 15:0 of the XNP Transmit Register, as shown in Figure 19-132. The 48-bit value is distributed over registers 0, 1, 2.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	NP	0	MSG_PAGE	ACK2	TOGGLE	MSG_UNF_CODE_FLD										
W																
Special Access																
Reset	0010_0000_0000_0001															
Offset	0x0016															

Figure 19-132. KX AN XNP Transmit Register 0

Table 19-139. KX AN XNP Transmit Register 0 Field Descriptions

Bits	Name	Description
15	NP	Next Page 0: Device does not have any more Next Pages to send 1: Device has more Next Pages to Send
14	—	Reserved
13	MSG_PAGE	Message Page 0: Next page is an unformatted page 1: Next page is a message page
12	ACK2	Acknowledge 2 0: The receiver is not able to act on the information defined in the message 1: The receiver is able to act on the information defined in the message
11	TOGGLE	Toggle
10-0	MSG_UNF_CODE_FLD	Message/Unformatted Code Field When MSG_PAGE=1: Message code field, else unformatted code field. For the null message code, the value is 0x1

19.7.2.7.16 1000Base-KX AN XNP Transmit Register 1

The 1000Base-KX AN XNP Transmit Register 1 contains bits 31:16 of the XNP Transmit Register, as shown in Figure 19-133. The 48-bit value is distributed over registers 0, 1, 2.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	UNF_CODE_FLD1															
Special Access																
Reset	0000_0000_0000_0000															
Offset	0x0017															

Figure 19-133. KX AN XNP Transmit Register 1

Table 19-140. KX AN XNP Transmit Register 1 Field Descriptions

Bits	Name	Description
15-0	UNF_CODE_FLD1	Unformatted Code Field 1

19.7.2.7.17 1000Base-KX AN XNP Transmit Register 2

The 1000Base-KX AN XNP Transmit Register 2 contains bits 48:32 of the XNP Transmit Register, as shown in Figure 19-134. The 48-bit value is distributed over registers 0, 1, 2. Register 2 must be written last.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	UNF_CODE_FLD2															
W																
Special Access																
Reset	0000_0000_0000_0000															
Offset	0x0018															

Figure 19-134. KX AN XNP Transmit Register 2

Table 19-141. KX AN XNP Transmit Register 2 Field Descriptions

Bits	Name	Description
15-0	UNF_CODE_FLD2	Unformatted Code Field 2

19.7.2.7.18 1000Base-KX AN LP XNP Ability Register 0

The 1000Base-KX AN LP XNP Ability Register 0 contains bits 15:0 of the LP XNP Ability Register, as shown in Figure 19-135. The 48-bit value is distributed over registers 0, 1, 2.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	NP	0	MSG_PAGE	ACK2	TOGGLE	MSG_UNF_CODE_FLD										
W																
Special Access																
Reset	0000_0000_0000_0000															
Offset	0x0019															

Figure 19-135. KX AN LP XNP Ability Register 0

Table 19-142. KX AN LP XNP Ability Register 0 Field Descriptions

Bits	Name	Description
15	NP	Next Page 0: Device does not have any more Next Pages to send 1: Device has more Next Pages to Send
14	—	Reserved
13	MSG_PAGE	Message Page 0: Next page is an unformatted page 1: Next page is a message page
12	ACK2	Acknowledge 2 0: The receiver is not able to act on the information defined in the message 1: The receiver is able to act on the information defined in the message
11	TOGGLE	Toggle
10-0	MSG_UNF_CODE_FLD	Message/Unformatted Code Field When MSG_PAGE=1: Message code field, else unformatted code field. For the null message code, the value is 0x1

19.7.2.7.19 1000Base-KX AN LP XNP Ability Register 1

The 1000Base-KX AN LP XNP Ability Register 1 contains bits 31:16 of the XNP Transmit Register, as shown in Figure 19-136. The 48-bit value is distributed over registers 0, 1, 2.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	UNF_CODE_FLD1															
W																
Special Access																
Reset	0000_0000_0000_0000															
Offset	0x001A															

Figure 19-136. KX AN LP XNP Ability Register 1

Table 19-143. KX AN LP XNP Ability Register 1 Field Descriptions

Bits	Name	Description
15-0	UNF_CODE_FLD1	Unformatted Code Field 1

19.7.2.7.20 1000Base-KX AN LP XNP Ability Register 2

The 1000Base-KX AN LP XNP Ability Register 2 contains bits 48:32 of the XNP Transmit Register, as shown in Figure 19-137. The 48-bit value is distributed over registers 0, 1, 2.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	UNF_CODE_FLD2															
Special Access																
Reset	0000_0000_0000_0000															
Offset	0x001B															

Figure 19-137. KX AN LP XNP Ability Register 2

Table 19-144. KX AN LP XNP Ability Register 2 Field Descriptions

Bits	Name	Description
15-0	UNF_CODE_FLD2	Unformatted Code Field 2

19.7.2.7.21 1000Base-KX AN BackPlane Ethernet Status Register

The 1000Base-KX AN Backplane Ethernet Status Register provides the result after auto-negotiation DME page exchange completed, as shown in Figure 19-138.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	100GBASE_CR10	0	40GBASE_CR4	40GBASE_KR4	BASE_R_FEC	10GBASE_KR	10GBASE_KX4	1000BASE_KX	BP_AN_ABIL
W																
Special Access																
Reset	0000_0001_0110_1111															
Offset	0x0030															

Figure 19-138. KX Backplane Ethernet Status Register

Table 19-145. KX AN Backplane Ethernet Status Register Field Descriptions

Bits	Name	Description
15-9	—	Reserved
8	100GBASE_CR10	Always 0 after negotiation complete
7	—	Reserved
6	40GBASE_CR4	Always 0 after negotiation complete
5	40GBASE_KR4	Always 0 after negotiation complete
4	BASE_R_FEC	Always 0 after negotiation complete
3	10GBASE_KR	Always 0 after negotiation complete
2	10GBASE_KX4	Always 0 after negotiation complete
1	1000BASE_KX	1000Base-KX 0: Not auto-negotiated to 1000Base-KX 1: Auto-negotiated to 1000Base-KX
0	BP_AN_ABIL	1000Base-KX capable PHY type is implemented Always 1

19.7.2.7.22 1000Base-KX Millisecond Counter Register

The 1000Base-KX AN Millisecond Counter Register contains the upper 16 bits of the 18-bit counter value for counting 1 ms, as shown in Figure 19-139.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	COUNT_MS															
W																
Special Access																
Reset	1001_1000_1001_0110															
Offset	0x8000															

Figure 19-139. KX Millesecond Counter Register

Table 19-146. KX Millisecond Counter Register Field Descriptions

Bits	Name	Description
15-0	COUNT_MS	Count Milliconds 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)

19.7.2.8 1000Base-KX Vendor-Specific 1 Registers

The 1000Base-KX MDIO space implements vendor-specific registers for IEEE 802.3 Clause 45 MMD 0x1D.

19.7.2.8.1 1000Base-KX Revision Register

The 1000Base-KX Version Register contains version information for the KX PHY, as shown in Figure 19-140.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CFG_VER								IP_VER				IP_REV			
W																
Special Access																
Reset	0000_0001_0001_0100															
Offset	0x0000															

Figure 19-140. KX Revision Register

Table 19-147. KX Revision Register Field Descriptions

Bits	Name	Description
15-8	CFG_VER	Configuration/Integration Version
7-4	IP_VER	IP Version
3-0	IP_REV	IP Revision

19.7.2.8.2 1000Base-KX Scratch Register

The 1000Base-KX Scratch Register may be used to test register reads and writes, as shown in Figure 19-141.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SCRATCH															
W																
Special Access																
Reset	0000_0000_0000_0000															
Offset	0x0001															

Figure 19-141. KX Scratch Register

Table 19-148. KX Scratch Register Field Descriptions

Bits	Name	Description
15-	SCRATCH	Scratch register

19.7.2.8.3 1000Base-KX PCS Interrupt Event Register

The 1000Base-KX PCS Interrupt Event Register contains detect bits for KX PCS interrupt events, as shown in Figure 19-142.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	—															PCS_LNK_LOS
W																
Special Access																RC
Reset	0000_0000_0000_0000															
Offset	0x0002															

Figure 19-142. KX PCS Interrupt Event Register

Table 19-149. KX PCS Interrupt Event Register Field Descriptions

Bits	Name	Description
15-1	—	Reserved
0	PCS_LNK_LOS	PCS Link Loss 0: PCS has not detected a loss of link synchronization 1: PCS has detected a loss of link synchronization Cleared on register read

19.7.2.8.4 1000Base-KX PCS Interrupt Mask Register

The 1000Base-KX PCS Interrupt Event Register contains mask bits for KX PCS interrupt events, as shown in Figure 19-143. See the Frame Manager reference manual for details on handling of PCS interrupts.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	—															PCS_LNK_LOS_EN
W																
Special Access																
Reset	0000_0000_0000_0000															
Offset	0x0003															

Figure 19-143. KX PCS Interrupt Mask Register

Table 19-150. KX PCS Interrupt Mask Register Field Descriptions

Bits	Name	Description
15-1	—	Reserved
0	PCS_LNK_LOS_EN	PCS Link Loss Event Enable 0: A PCS loss of link synchronization will not cause an interrupt event 0: A PCS loss of link synchronization will cause an interrupt event

19.7.2.8.5 1000Base-KX AN Interrupt Event Register

The 1000Base-KX PCS Interrupt Event Register contains detect bits for KX auto-negotiation interrupt events, as shown in Figure 19-144.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	—													AN_PARDET_FLT	AN_PAGE_RX	AN_DONE
W																
Special Access														RC	RC	RC
Reset	0000_0000_0000_0000															
Offset	0x0004															

Figure 19-144. KX AN Interrupt Event Register

Table 19-151. KX AN Interrupt Event Register Field Descriptions

Bits	Name	Description
15-3	—	Reserved
2	AN_PARDET_FLT	Parallel detection fault 0: No fault detected 1: Ambiguous link status detected while waiting on DME page receive Cleared on register read
1	AN_PAGE_RX	Auto-negotiate page receive 0: An ability word was not received from the remote device 1: An ability word was received from the remote device during backplane auto-negotiation Indicates the validity of the remote ability word (base page or next page) Cleared on register read
0	AN_DONE	Backplane auto-negotiation complete 0: Backplane auto-negotiation not complete 1: Backplane auto-negotiation complete Cleared on register read Note: not used for 1000Base-X/SGMII auto-negotiation

19.7.2.8.6 1000Base-KX AN Interrupt Mask Register

The 1000Base-KX PCS Interrupt Event Register contains mask bits for KX AN interrupt events, as shown in Figure 19-145. See the Frame Manager reference manual for details on handling of AN interrupts.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	—													AN_PARDET_FLT_EN	AN_PAGE_RX_EN	AN_DONE_EN
Special Access																
Reset	0000_0000_0000_0000															
Offset	0x0005															

Figure 19-145. KX AN Interrupt Mask Register

Table 19-152. KX AN Interrupt Mask Register Field Descriptions

Bits	Name	Description
15-3	—	Reserved
2	AN_PARDET_FLT_EN	AN Parallel Detection Fault Event Enable 0: AN Parallel Dection Fault will not cause an interrupt event 1: AN Parallel Dection Fault will cause an interrupt event
1	AN_PAGE_RX_EN	AN Page Recieve Event Enable 0: AN Page Receive will not cause an interrupt event 1: AN Page Receive will cause an interrupt event
0	AN_DONE_EN	AN Done Event Enable 0: AN done will not cause an interrupt event 1: AN done will cause an interrupt event

19.8 Initialization/Application Information

19.8.1 Initialization

All initialization necessary for the SerDes module to start operation is done automatically based on RCW, with the exception of the following protocols:

19.8.1.1 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 for PCSs connected to FM MAC.

19.8.1.2 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 Section 19.8.4.1, "Lane Reset and Reconfiguration, prior to initializing the SGMII PCS:

- SRDSxLNmGCR1[REIDL_TH]
- SRDSxLNmGCR1[REIDL_EX_SEL]
- SRDSxLNmGCR1[REIDL_ET_MSB]
- SRDSxLNmGCR1[ISLEW_RCTL]
- SRDSxLNmGCR1[OSLEW_RCTL]
- SRDSxLNmRECR0[GK2OVD]
- SRDSxLNmRECR0[GK3OVD]
- SRDSxLNmRECR0[GK2OVD_EN]
- SRDSxLNmRECR0[GK3OVD_EN]
- SRDSxLNmTECR0[TEQ_TYPE]
- SRDSxLNmTECR0[RATIO_PST1Q]
- SRDSxLNmTECR0[AMP_RED]

See Section 19.6.2.7.2, "SRDSx Lane m General Control Register 1 (SRDSxLNmGCR1), Section 19.6.2.7.4, "SRDSx Lane m Receive Equalization Control Register 0 (SRDSxLNmRECR0) and Section 19.6.2.7.6, "SRDSx Lane m Transmit Equalization Control Register 0 (SRDSxLNmTECR0) for details.

The PCS initialization sequence for 2.5G SGMII is the same as for 1G SGMII with auto-negotiation disabled (see Section 19.8.1.1, "1G SGMII). Auto-negotiation is not supported for 2.5G SGMII.

19.8.1.3 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 Section 19.8.4.1, "Lane Reset and Reconfiguration, prior to performing 1000Base-KX auto-negotiation:

- SRDSxLNmGCR1[REIDL_TH]
- SRDSxLNmGCR1[REIDL_EX_SEL]
- SRDSxLNmGCR1[REIDL_ET_MSB]
- SRDSxLNmTECR0[AMP_RED]

See Section 19.6.2.7.2, "SRDSx Lane m General Control Register 1 (SRDSxLNmGCR1) and Section 19.6.2.7.6, "SRDSx Lane m Transmit Equalization Control Register 0 (SRDSxLNmTECR0) 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 SRDSxPLLnCR0[DLYDIV_SEL]=01, for the PLLn that is the clock source for the SGMII PCS (see TPLL_LES in Section 19.6.2.7.1, "SRDSx Lane m General Control Register 0 (SRDSxLNmGCR0)).
1. Enable the 1000Base-KX AN module by setting SRDSxPCCR1[SGMIIp_KX]=1, for each SGMIIp that will run in 1000Base-KX rather than SGMII mode.
2. Initialize SGMII IF Mode register to 0x0008:
   — SGMII_EN=0
   — USE_SGMII_AN=0
   — SGMII_SPEED=10
3. Initialize 1000Base-KX (Clause 45) AN Advertisement Register 1:
   — TX_NONCE=unique value per device
4. Read 1000Base-KX (Clause 45) AN Status Register to clear any previous status
5. Initialize 1000Base-KX (Clause 45) AN Control Register to 0x1200:
   — AN_ENAB=1
   — RST_AN=1

19.8.1.4 XFI/10GBase-KR

All default SerDes settings are for XFI. F

For 10GBase-KR, the following settings need to be modified, following the procedure in Section 19.8.4.1, "Lane Reset and Reconfiguration:

- SRDSxLNmTECR0[TEQ_TYPE]
- SRDSxLNmTECR0[SGN_PREQ]
- SRDSxLNmTECR0[RATIO_PREQ]
- SRDSxLNmTECR0[RATIO_PST1Q]
- SRDSxLNmTECR0[AMP_RED]

See Section 19.6.2.7.6, “SRDSx Lane m Transmit Equalization Control Register 0 (SRDSxLNmTECR0) for details.

19.8.1.5 S-RIO Long Reach

The default SerDes settings for S-RIO are for short reach.

For long reach, the following settings need to be updated following the procedure in Section 19.8.4.1, “Lane Reset and Reconfiguration:

- SRDSxLNmTECR0[AMP_RED]

See Section 19.6.2.7.6, “SRDSx Lane m Transmit Equalization Control Register 0 (SRDSxLNmTECR0) for details.

19.8.2 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 SRDSxLNmGCR0[RX_PD]=1 and SRDSxLNmGCR0[RRST_B]=0.

Power down the Tx portion of the lanes by setting SRDSxLNmGCR0[TX_PD]=1 and SRDSxLNmGCR0[TRST_B]=0.

The Rx and Tx portions of the lanes may be independently powered down for uni-directional lanes or links (e.g. Tx-only Aurora lanes).

19.8.3 Frequency Negotiation

PCI Express and SATA auto-negotiate frequency in hardware. For SATA and PCIe 2.5/5 Gbaud, that auto-negotiation involves overriding the by-n rate select function of the SerDes lane.For PCIe 8 Gbaud, the auto-negotiation also involves a combination of switching PLL selects, and/or reconfiguring a PLL. The current state of the by-n rate selects and PLL select per lane are reflected in SRDSxLNnGCR0, as defined in Section 19.6.2.7.1, “SRDSx Lane m General Control Register 0 (SRDSxLNmGCR0).” The current state of the PLL full-rate frequency select is reflected in SRDSx112, as defined in Section 19.6.2.1.2, “SRDSx PLL Control Register 0 (SRDSxPLLnCR0).”

Note: SGMII can operate at 10 MBps, 100 Mbps, or 1000 Mbps or 2500 Mbps, but the frequency of operation of the serial link is 1.25 GHz for 10/100/1000, and 3.125 GHz for 2500. The lower effective data rates on SGMII 10/100 are implemented by replicating symbols 10 or 100 times.

Other protocols either have a static frequency configuration, or change frequencies via a software reconfiguring sequence. See Section 19.8.4.1, “Lane Reset and Reconfiguration,” and Section 19.8.4.3, “PLL Reset and Reconfiguration,” for details on the software reconfiguring sequences.

19.8.4 Soft Reset and Reconfiguring Procedures

19.8.4.1 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 SRDSxLNmGCR0[TRST_B]=0 and SRDSxLNmGCR0[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 SRDSxLNmGCR0[TRST_B]=1 and SRDSxLNmGCR0[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 SRDSxLNmGCR0[1STLANE]=1. All lanes either to the left (if SRDSxLNpGCR1[TRSTDIR]=0) or to the right (if SRDSxLNpGCR1[TRSTDIR]=1) of the master source clock lane until the next lane with SRDSxLNmGCR0[1STLANE]=1, or the end of the SerDes, are grouped with the master source clock lane.

It is recommended to disable any controller connected to a lane being reconfigured prior to starting the reconfiguration sequence.

19.8.4.2 Lane Enable After Powerdown

To enable a previously powered down lane, set SRDSxLNmGCR0[nX_PD]=0 (n=R or T), wait 15 us, then set SRDSxLNmGCR0[nRST]=1.

Note that if a Tx lane m with SRDSxLNmGCR0[1STLANE]=1 (master Tx clock lane) is powered down or reset, all Tx lanes to the left of it (p<m) with SRDSxLNpGCR1[TRSTDIR]=0, or all Tx lanes to the right of it (p>m) with SRDSxLNpGCR1[TRSTDIR]=1, cannot be used.

19.8.4.3 PLL Reset and Reconfiguration

To reconfigure a PLL, perform the following sequence:

1. Disable all lanes using the PLL to be reconfigured by setting SRDSxPLLnRSTCTL[SDRST_B]=0
2. Wait at least 50 ns
1. Disable the PLL by setting SRDSxPLLnRSTCTL[SDEN]=0 and SRDSxPLLnRSTCTL[PLLRST_B]=0
2. Wait at least 100 ns
3. Change the desired per-PLL settings
4. Reset the PLL by setting SRDSxPLLnRSTCTL[RSTREQ]=1
5. Set SRDSxPLLnRSTCTL[SDEN]=1, SRDSxPLLnRSTCTL[PLLRST_B]=1, and SRDSxPLLnRSTCTL[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 Section 19.8.4.1, "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.

19.8.5 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 20
PCI Express Interface Controller

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.

20.1 Introduction

The PCI Express controller provides the mechanism to communicate with PCI Express devices. Figure 20-1 is a high-level block diagram of the PCI Express controller.

20.1.1 Overview

The PCI Express controller connects the internal platform to a 8.0-GHz serial interface.

The remainder of this chapter refers to a single PCI Express controller offering up to a ×8 link interface. Notes are included to indicate variations for multiple instantiations.

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 an 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.

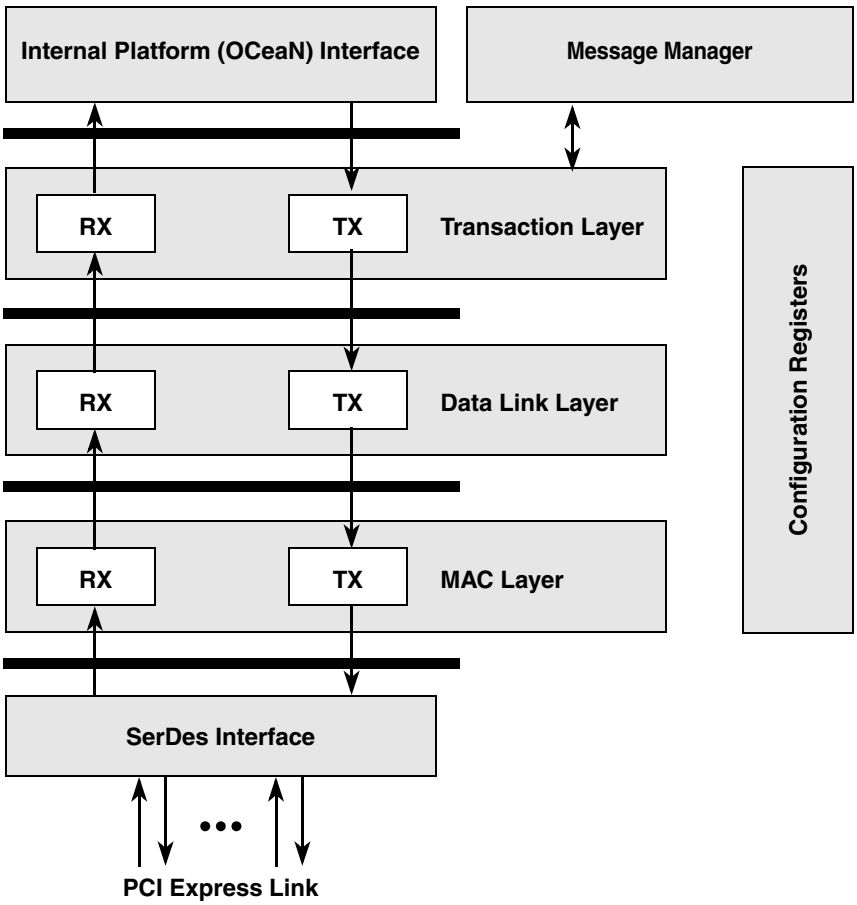


Figure 20-1. PCI Express Controller Block Diagram

As an initiator, the PCI Express controller supports memory read and write operations with a maximum transaction size of 256 bytes. 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. When configured 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.

20.1.1.1 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 register (PEX_CSR0) to check the status of link training before issuing external requests.

20.1.1.2 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.

20.1.2 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
- ×8, ×4, ×2, and ×1 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 three inbound general-purpose translation windows, one 64-bit MSI window (RC only), and one configuration window which can be alternately configured as a general purpose window.
- Supports 32- or 64-bit MSI interrupts
- Supports four outbound translation windows and one default window
- Supports twelve non-posted and eight posted PCI Express inbound transactions
- Supports up to ten internal platform transactions
- 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 memory protection on a per transaction basis (inbound) using LIODN
- Supports separate LIODN base register for Inbound MSI ATMU hit.
- Supports SR-IOV 1.1 spec version with 2 PFs and 64 VFs per PF (total of 128 VFs)
- Supports Functional Level Reset (FLR)
- Supports Alternative Routing ID (ARI)
- Supports MSI-X for PF/VF with 8 MSI-X vector per PF or VF.
- Supports Expansion ROM.
- Supports MSI-X trap operation.

20.1.3 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 20-1. POR Parameters for PCI Express Controller

RCW Parameter	Description	Section/Page
Host/Agent HOST_AGT_PEX	Selects between root complex (RC) and endpoint (EP) modes	
SerDes Protocol Select SRDS_PRTCL_S*n*	Determines the link width and operational generation possibilities (for example 1.0, 2.0, 3.0)	
SerDes frequency divider for PCI Express SRDS_DIV_PEX_S*n*	Determines the link speed	

20.1.3.1 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.

20.1.3.2 Link Width

The initial link width is determined by the SerDes protocol configuration field (RCW[SRDS_PRTCL_S*n*]). See Reset Configuration Word (RCW) for more information. The specific configurations are detailed in SerDes Lane Assignments and Multiplexing.

20.1.3.3 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.

20.1.3.4 LIODN

Logical I/O Device Number (LIODN) is a mechanism where the memory subsystem identifies the LIODN that is assigned to an inbound PCI Express memory transaction based on its transaction ID (that is, RID), looks up its memory access permissions in a table, and performs the appropriate actions based on the outcome of the table look-up.

20.1.3.5 EP SR-IOV

Single Root I/O Virtualization (SR-IOV) allows for different software images to share IO resources that exist on an EP device. With SR-IOV supported in EP, different devices or different software tasks can share IO resources such as Gigabit Ethernet controllers with minimum code modifications. Additional benefits include savings in hardware cost for additional and/or dedicated IO resources, space and power consumption. A total of two PFs and 128 VFs are supported. Each PF will have its own dedicated 8-Kbyte memory-mapped register space. Unless stated otherwise, all memory-mapped registers, except inbound and outbound ATMUs, are shared among the PFs.

20.1.3.6 MSI-X

MSI-X allows for EP device to send message interrupts to RC device independently for different Physical or Virtual functions as supported by EP SR-IOV. Each PF or VF will have eight MSI-X vectors allocated with a total of 256 total MSI-X vectors supported.

20.2 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* and TX*n*_B) and a differential pair for receiving (RX_P*n* and RX_N*n*) with an embedded data clock.

Table 20-2. PCI Express Interface Signals—Detailed Signal Descriptions

Signal	I/O	Description	
SD_RX[*n*]	I	Receive data. 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*] _B	I	Receive data inverted. The receive data signals carry PCI Express packet information.	
		State Meaning	Asserted/Negated—Represents inverted 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*]	O	Transmit data. The transmit data signals carry PCI Express packet information.	
		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*]_B	O	Transmit data, inverted. Represent the inverted form of the transmit data signals	
		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*.

20.3 Memory Map/Register Definitions

The PCI Express interface supports the following register types:

- Memory-mapped registers—these registers control PCI Express address translation, PCI error management, and PCI Express configuration register access. These registers are described in Section 20.3.1, "PCI Express Memory Mapped Registers," and its subsections. The memory mapped registers are accessed by reading and writing to an address comprised of the base address (specified in CCSRBAR) plus the block base address, plus the offset of the specific register to be accessed. Note that all memory-mapped registers must only be accessed as 32-bit quantities. Each PF in EP occupies an 8-Kbyte CCSR register space. The block base addresses in EP are as follows:
  - PF0 starts at 0x0000 address offset
  - PF1 starts at 0x2000 address offset
- PCI Express configuration registers contained within the PCI Express configuration space—these registers are specified by the PCI Express specification for every PCI Express device. These registers are described in Section 20.3.8, "PCI Express Configuration Space Access," and its subsections. Each PF/VF will have its own configuration registers as specified by the PCI Express SR-IOV specification.

20.3.1 PCI Express Memory Mapped Registers

The PCI Express memory mapped registers are accessed by reading and writing to an address comprised of the base address (specified in the CCSRBAR on the local side or the PEXCSRBAR on the PCI Express side) plus the block base address, plus the offset of the specific register to be accessed. Note that all

memory-mapped registers (except the PCI Express configuration data register, PEX_CONFIG_DATA) must only be accessed as 32-bit quantities.

Table 20-3 lists the memory-mapped registers. In this table and in the register figures and field descriptions, the following access definitions apply:

- Reserved fields are always ignored for the purposes of determining access type.
- R/W, R, and W(read/write, read only, and write only) indicate that all the non-reserved fields in a register have the same access type. All PFs share the same register.
- 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.
- P-R/W, P-R, P-W, P-Mixed indicate that each individual PF has its own register implemented
- PV-R/W, PV-R, PV-W, PV-Mixed indicate that each individual PF and VF have their own register implemented
- Special is used when no other category applies. In this case the register figure and field description table should be read carefully.

Table 20-3. PCI Express Memory-Mapped Register Map

PCI Express Controller				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
PCI Express Configuration Access Registers				
0x000	PEX_CONFIG_ADDR—PCI Express configuration address register	R/W	All zeros	20.3.2.1/20-16
0x004	PEX_CONFIG_DATA—PCI Express configuration data register	R/W	All zeros	20.3.2.2/20-18
0x008	Reserved	—	—	
0x00C	PEX_OTB_CPL_TOR—PCI Express outbound completion timeout register	R/W	0x0013_FFFF	20.3.2.3/20-18
0x010	PEX_CONF_RTY_TOR—PCI Express configuration retry timeout register	R/W	0x0400_FFFF	20.3.2.4/20-19
0x014	PEX_CONFIG—PCI Express configuration register	R/W	0x00nn_0028	20.3.2.5/20-20
0x018	PEX_INT_STAT—PCI Express interrupt status register	RO	0x0000_0000	20.3.2.6/20-21
0x01C	Reserved	—	—	
PCI Express Power Management Event & Message Registers				
0x020	PEX_PME_MES_DR—PCI Express PME & message detect register	P-w1c	All zeros	20.3.3.1/20-23
0x024	PEX_PME_MES_DISR—PCI Express PME & message disable register	P-R/W	All zeros	20.3.3.2/20-24
0x028	PEX_PME_MES_IER—PCI Express PME & message interrupt enable register	P-R/W	All zeros	20.3.3.3/20-26
0x02C	PEX_PMCR—PCI Express power management command register	P-R/W	All zeros	20.3.3.4/20-27
0x030–0x03C	Reserved	—	—	
PCI Express LIODN Registers				
0x040	PEX_LBR—PCI Express LIODN base register	R/W	All zeros	20.3.3.5/20-27

Table 20-3. PCI Express Memory-Mapped Register Map (continued)

PCI Express Controller				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
0x044	PEXMSI_LBR—PCI Express MSI LIODN base register	R/W	All zeros	20.3.3.6/20-28
0x48–0x4C	Reserved	—	—	
PCI Express Utility Registers				
0x050	PEX_AOR—PCI Express address offset register	R/W	All zeros	20.3.3.7/20-29
	Reserved	—	—	
0x058	PEX_UDR—PCI Express upper data register	R/W	All zeros	20.3.3.8/20-32
0x05C	PEX_LDR—PCI Express lower data register	R/W	All zeros	20.3.3.9/20-32
0x060-0xBF4	Reserved	—	—	
PCI Express IP Block Revision Registers				
0xBF8	IP block revision register 1 (PEX_IP_BLK_REV1)	R	0x0208_0300	20.3.4.1/20-33
0xBFC	IP block revision register 2 (PEX_IP_BLK_REV2)	R	All zeros	20.3.4.2/20-33
PCI Express ATMU Registers				
Outbound Window 0 (Default)				
0xC00	PEXOTAR0—PCI Express outbound translation address register 0 (default)	• R/W in RC • P-R/W in EP for PF0 only • Reserved for all other PF	All zeros	20.3.5.1.5/20-41
0xC04	PEXOTEAR0—PCI Express outbound translation extended address register 0 (default)	• R/W in RC • P-R/W in EP for PF0 only • Reserved for all other PF	All zeros	20.3.5.1.6/20-42
0xC08–0xC0C	Reserved	—	—	
0xC10	PEXOWAR0—PCI Express outbound window attributes register 0 (default)	• Mixed in RC • P-Mixed in EP for PF0 only • Reserved for all other PF	0x8004_4027	20.3.5.1.8/20-43
0xC14–0xC1C	Reserved	—	—	
Outbound Window 1				

Table 20-3. PCI Express Memory-Mapped Register Map (continued)

PCI Express Controller				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
0xC20	PEXOTAR1—PCI Express outbound translation address register 1	R/W in RC P-R/W in EP	All zeros	20.3.5.1.5/20-41
0xC24	PEXOTEAR1—PCI Express outbound translation extended address register 1	R/W in RC P-R/W in EP	All zeros	20.3.5.1.6/20-42
0xC28	PEXOWBAR1—PCI Express outbound window base address register 1	R/W in RC P-R/W in EP	All zeros	20.3.5.1.7/20-42
0xC2C	Reserved	—	—	
0xC30	PEXOWAR1—PCI Express outbound window attributes register 1	R/W in RC P-R/W in EP	0x0004_4027	20.3.5.1.8/20-43
0xC34–0xC3C	Reserved	—	—	
Outbound Window 2				
0xC40	PEXOTAR2—PCI Express outbound translation address register 2	R/W in RC P-R/W in EP	All zeros	20.3.5.1.5/20-41
0xC44	PEXOTEAR2—PCI Express outbound translation extended address register 2	R/W in RC P-R/W in EP	All zeros	20.3.5.1.6/20-42
0xC48	PEXOWBAR2—PCI Express outbound window base address register 2	R/W in RC P-R/W in EP	All zeros	20.3.5.1.7/20-42
0xC4C	Reserved	—	—	
0xC50	PEXOWAR2—PCI Express outbound window attributes register 2	R/W in RC P-R/W in EP	0x0004_4027	20.3.5.1.8/20-43
0xC54–0xC5C	Reserved	—	—	
Outbound Window 3				
0xC60	PEXOTAR3—PCI Express outbound translation address register 3	R/W in RC P-R/W in EP	All zeros	20.3.5.1.5/20-41
0xC64	PEXOTEAR3—PCI Express outbound translation extended address register 3	R/W in RC P-R/W in EP	All zeros	20.3.5.1.6/20-42
0xC68	PEXOWBAR3—PCI Express outbound window base address register 3	R/W in RC P-R/W in EP	All zeros	20.3.5.1.7/20-42
0xC6C	Reserved	—	—	

Table 20-3. PCI Express Memory-Mapped Register Map (continued)

PCI Express Controller				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
0xC70	PEXOWAR3—PCI Express outbound window attributes register 3	R/W in RC P-R/W in EP	0x0000_0000	20.3.5.1.8/20-43
0xC74–0xC7C	Reserved	—	—	
Outbound Window 4				
0xC80	PEXOTAR4—PCI Express outbound translation address register 4	R/W in RC P-R/W in EP	All zeros	20.3.5.1.5/20-41
0xC84	PEXOTEAR4—PCI Express outbound translation extended address register 4	R/W in RC P-R/W in EP	All zeros	20.3.5.1.6/20-42
0xC88	PEXOWBAR4—PCI Express outbound window base address register 4	R/W in RC P-R/W in EP	All zeros	20.3.5.1.7/20-42
0xC8C	Reserved	—	—	
0xC90	PEXOWAR4—PCI Express outbound window attributes register 4	R/W in RC P-R/W in EP	0x0004_4027	20.3.5.1.8/20-43
0xC94–0xCBC	Reserved	—	—	
PF MSI-X Trap Outbound Window Address Registers (EP only)				
0xCC0-0xCC4	Reserved	—	—	
0xCC8	PEXMSIX_TPOWBAR—PCI Express MSIX trap outbound window base address register	P-R/W	All zeros	20.3.5.2/20-46
0xCCC	Reserved	—	—	
0xCD0	PEXMSIX_TPOWAR—PCI Express MSIX trap outbound window attribute register	P-R/W	0x0004_400B	20.3.5.3/20-47
0xCD4-0xCDC	Reserved	—	—	
Expansion ROM Inbound Window				
0xCE0	PEXEPROMITAR—PCI Express Expansion ROM inbound translation address register	• R/W in RC • P-R/W in EP for PF0 • P-RO for all other PF	All zeros	20.3.6.1/20-59
0xCE4–0xCEC	Reserved	—	—	

Table 20-3. PCI Express Memory-Mapped Register Map (continued)

PCI Express Controller				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
0xCF0	PEXEPROMIWAR—PCI Express Expansion ROM inbound window attribute register	• R/W in RC • P-R/W in EP for PF0 • P-RO for all other PF	0x20F4_4017	20.3.6.2/20-60
0xCF4–0xCFC	Reserved	—	—	
Inbound MSI Window				
0xD00	PEXMSIITAR—PCI Express MSI inbound translation address register	R/W in RC only	0xnnnn_nnnn	20.3.6.2.1/20-62
0xD04	Reserved	—	—	
0xD08	PEXMSIIWBAR—PCI Express MSI inbound window base address register	R/W in RC only	All zeros	20.3.6.2.2/20-63
0xD0C	PEXMSIIWBEAR—PCI Express MSI inbound window base extended address register	R/W in RC only	All zeros	20.3.6.2.3/20-64
0xD10	PEXMSIIWAR—PCI Express MSI inbound window attributes register	R/W in RC only	0x00n4_40nn	20.3.6.2.4/20-64
0xD14–0xD9C	Reserved	—	—	
Inbound Window 3				
0xD80	PEXITAR3—PCI Express inbound translation address register 3	R/W in RC P-R/W in EP	0x0000_0000	20.3.5.4.5/20-54
0xD84	Reserved	—	—	
0xD88	PEXIWBAR3—PCI Express inbound window base address register 3	R/W in RC only	0x0000_0000	20.3.5.4.6/20-55
0xD8C	PEXIWBEAR3—PCI Express inbound window base extended address register 3	R/W in RC only	0x0000_0000	20.3.5.4.7/20-56
0xD90	PEXIWAR3—PCI Express inbound window attributes register 3	R/Win RC P-R/W in EP	0x20F4_4027	20.3.5.4.8/20-56
0xD94–0xD9C	Reserved	—	—	
Inbound Window 2				
0xDA0	PEXITAR2—PCI Express inbound translation address register 2	R/W in RC P-R/W in EP	0x0000_0000	20.3.5.4.5/20-54
0xDA4	Reserved	—	—	
0xDA8	PEXIWBAR2—PCI Express inbound window base address register 2	R/W in RC only	0x0000_0000	20.3.5.4.6/20-55

Table 20-3. PCI Express Memory-Mapped Register Map (continued)

PCI Express Controller				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
0xDAC	PEXIWBEAR2—PCI Express inbound window base extended address register 2	R/W in RC only	0x0000_0000	20.3.5.4.7/20-56
0xDB0	PEXIWAR2—PCI Express inbound window attributes register 2	R/W in RC P-R/W in EP	0x20F4_4027	20.3.5.4.8/20-56
0xDB4–0xDBC	Reserved	—	—	
Inbound Window 1				
0xDC0	PEXITAR1—PCI Express inbound translation address register 1	R/W in RC P-R/W in EP	0x0000_0000	20.3.5.4.5/20-54
0xDC4	Reserved	—	—	
0xDC8	PEXIWBAR1—PCI Express inbound window base address register 1	R/W in RC only	0x0000_0000	20.3.5.4.6/20-55
0xDCC	Reserved	—	—	
0xDD0	PEXIWAR1—PCI Express inbound window attributes register 1	R/W in RC P-R/W in EP	0x20F4_4027	20.3.5.4.8/20-56
0xDD4–0xDDC	Reserved	—	—	
Inbound Window 0				
0xDE0	PEXITAR0—PCI Express inbound translation address register 0	R/W in RC P-R/W in EP	0x0000_0000	20.3.5.4.5/20-54
0xDE4	Reserved	—	—	
0xDE8	Reserved	—	—	
0xDEC	Reserved	—	—	
0xDF0	PEXIWAR0—PCI Express inbound window attributes register 0	R/W in RC P-R/W in EP	0x80n4_40nn	20.3.5.4.8/20-56
0xDF4–0xDFC	Reserved	—	—	
PCI Express Error Management Registers				
0xE00	PEX_ERR_DR—PCI Express error detect register	w1c	All zeros	20.3.7.1/20-66
0xE04	Reserved	—	—	—
0xE08	PEX_ERR_EN—PCI Express error interrupt enable register	R/W	All zeros	20.3.7.2/20-70
0xE0C	Reserved	—	—	—
0xE10	PEX_ERR_DISR—PCI Express error disable register	R/W	All zeros	20.3.7.3/20-72
0xE14–0xE1C	Reserved	—	—	—
0xE20	PEX_ERR_CAP_STAT—PCI Express error capture status register	Mixed	All zeros	20.3.7.4/20-74

Table 20-3. PCI Express Memory-Mapped Register Map (continued)

PCI Express Controller				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
0xE24	Reserved	—	—	—
0xE28	PEX_ERR_CAP_R0—PCI Express error capture register 0	R/W	All zeros	20.3.7.5/20-75
0xE2C	PEX_ERR_CAP_R1—PCI Express error capture register 1	R/W	All zeros	20.3.7.6/20-76
0xE30	PEX_ERR_CAP_R2—PCI Express error capture register 2	R/W	All zeros	20.3.7.7/20-78
0xE34	PEX_ERR_CAP_R3—PCI Express error capture register 3	R/W	All zeros	20.3.7.8/20-80
0xE38–0xF10	Reserved	—	—	
0xF14	PEX_CSR0—PEX control/status register 0	R	See register description	20.3.7.9/20-81
0xF18	PEX_CSR1—PEX control/status register 1	R/W	See register description	20.3.7.10/20-82
0xF1C–0xFFC	Reserved	—	—	
PCI Express VF(s) ATMU Registers (EP only)				
Outbound Window 1				
0x1000–0x1004	Reserved	—	—	
0x1008	PEXVFOWBAR1—PCI Express VF outbound window base address register 1	P-R/W	All zeros	20.3.5.1.3/20-39
0x100C	Reserved	—	—	
0x1010	PEXVFOWAR1—PCI Express VF outbound window attributes register 1	P-R/W	0x0004_401F	20.3.5.1.4/20-39
0x1014–0x101C	Reserved	—	—	
Outbound Window 2				
0x1020–0x1024	Reserved	—	—	
0x1028	PEXVFOWBAR2—PCI Express VF outbound window base address register 2	P-R/W	All zeros	20.3.5.1.3/20-39
0x102C	Reserved	—	—	
0x1030	PEXVFOWAR2—PCI Express VF outbound window attributes register 2	P-R/W	0x0004_401F	20.3.5.1.4/20-39
0x1034–0x103C	Reserved	—	—	
Outbound Window 3				
0x1040–0x1044	Reserved	—	—	
0x1048	PEXVFOWBAR3—PCI Express VF outbound window base address register 3	P-R/W	All zeros	20.3.5.1.3/20-39
0x104C	Reserved	—	—	

Table 20-3. PCI Express Memory-Mapped Register Map (continued)

PCI Express Controller				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
0x1050	PEXVF OWAR3—PCI Express VF outbound window attributes register 3	P-R/W	0x0004_401F	20.3.5.1.4/20-39
0x1054–0x105C	Reserved	—	—	
Outbound Window 4				
0x1060–0x1064	Reserved	—	—	
0x1068	PEXVFOWBAR4—PCI Express VF outbound window base address register 4	P-R/W	All zeros	20.3.5.1.3/20-39
0x106C	Reserved	—	—	
0x1070	PEXVFOWAR4—PCI Express VF outbound window attributes register 4	P-R/W	0x0004_401F	20.3.5.1.4/20-39
0x1074–0x107C	Reserved	—	—	
Inbound Window 3				
0x1080	PEXVFIWTAR3—PCI Express VF inbound window translation address register 3	P-R/W	All zeros	20.3.5.4.3/20-51
0x1084–0x108C	Reserved	—	—	
0x1090	PEXVFIWAR3—PCI Express VF inbound window attributes register 3	P-R/W	0x20F4_401F	20.3.5.4.4/20-52
0x1094–0x109C	Reserved	—	—	
Inbound Window 2				
0x10A0	PEXVFIWTAR2—PCI Express VF inbound window translation address register 2	P-R/W	All zeros	20.3.5.4.3/20-51
0x10A4–0x10AC	Reserved	—	—	
0x10B0	PEXVFIWAR2—PCI Express VF inbound window attributes register 2	P-R/W	0x20F4_401F	20.3.5.4.4/20-52
0x10B4–0x10BC	Reserved	—	—	
Inbound Window 1				
0x10C0	PEXVFIWTAR1—PCI Express VF inbound window translation address register 1	P-R/W	All zeros	20.3.5.4.3/20-51
0x10C4–0x10CC	Reserved	—	—	
0x10D0	PEXVFIWAR1—PCI Express VF inbound window attributes register 1	P-R/W	0x20F4_401F	20.3.5.4.4/20-52
0x10D4–0x10DC	Reserved	—	—	

Table 20-3. PCI Express Memory-Mapped Register Map (continued)

PCI Express Controller				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
Inbound Window 0				
0x10E0	PEXVFIWTAR0—PCI Express VF inbound window translation address register 0	P-R/W	All zeros	20.3.5.4.3/20-51
0x10E4–0x10EC	Reserved	—	—	
0x10F0	PEXVFIWAR0—PCI Express VF inbound window attributes register 0	P-R/W	0x80n4_40nn	20.3.5.4.4/20-52
0x10F4–0x17FC	Reserved	—	—	
Outbound VF Translation Address Registers				
Outbound Window 1				
0x1800	PEXVFOW1TAR1—PCI Express VF outbound window 1 translation address register 1	P-R/W	All zeros	20.3.5.1.1/20-38
0x1804	PEXVFOW1TEAR1—PCI Express VF outbound window 1extended translation address register 1	P-R/W	All zeros	20.3.5.1.2/20-38
0x1808-0x19F4	PEXVFOW1TAR2-63, PEX_VFOW1TEAR2-63—PCI Express VF outbound window 1 (extended) translation address register 2-63	P-R/W	All zeros	20.3.5.1.1/20-38 20.3.5.1.2/20-38
0x19F8	PEXVFOW1TAR64—PCI Express VF outbound window 1 translation address register 64	P-R/W	All zeros	20.3.5.1.1/20-38
0x19FC	PEXVFOW1TEAR64—PCI Express VF outbound window 1 extended translation address register 64	P-R/W	All zeros	20.3.5.1.2/20-38
Outbound Window 2				
0x1A00	PEXVFOW2TAR1—PCI Express VF outbound window 2 translation address register 1	P-R/W	All zeros	20.3.5.1.1/20-38
0x1A04	PEXVFOW2TAR1—PCI Express VF outbound window 2 extended translation address register 1	P-R/W	All zeros	20.3.5.1.2/20-38
0x1B08-0x13F4	PEXVFOW2TAR2-63, PEXVFOW2TEAR2-63—PCI Express VF outbound window 2 (extended) translation address register 2-63	P-R/W	All zeros	20.3.5.1.1/20-38 20.3.5.1.2/20-38
0x1BF8	PEXVFOW2TAR64—PCI Express VF outbound window 2 translation address register 64	P-R/W	All zeros	20.3.5.1.1/20-38
0x1BFC	PEXVFOW2TEAR64—PCI Express VF outbound window 2 extended translation address register 64	P-R/W	All zeros	20.3.5.1.2/20-38
Outbound Window 3				
0x1C00	PEXVFOW3TAR1—PCI Express VF outbound window 3 translation address register 1	P-R/W	All zeros	20.3.5.1.1/20-38
0x1C04	PEXVFOW3TAR1—PCI Express VF outbound window 3 extended translation address register 1	P-R/W	All zeros	20.3.5.1.2/20-38
0x1C08-0x1DF4	PEXVFOW3TAR2-63, PEXVFOW3TEAR2-63—PCI Express VF outbound window 3 (extended) translation address register 2-63	P-R/W	All zeros	20.3.5.1.1/20-38 20.3.5.1.2/20-38
0x1DF8	PEXVFOW3TAR64—PCI Express VF outbound window 3 translation address register 64	P-R/W	All zeros	20.3.5.1.1/20-38

Table 20-3. PCI Express Memory-Mapped Register Map (continued)

PCI Express Controller				
Offset	**Register**	**Access**	**Reset**	**Section/Page**
0x1DFC	PEXVFOW3TEAR64—PCI Express VF outbound window 3 extended translation address register 64	P-R/W	All zeros	20.3.5.1.2/20-38
Outbound Window 4				
0x1E00	PEXVFOW4TAR1—PCI Express VF outbound window 4 translation address register 1	P-R/W	All zeros	20.3.5.1.1/20-38
0x1E04	PEXVFOW4TAR1—PCI Express VF outbound window 4 extended translation address register 1	P-R/W	All zeros	20.3.5.1.2/20-38
0x1E08-0x1FF4	PEXVFOW4TAR2-63, PEXVFOW4TEAR2-63—PCI Express VF outbound window 4 (extended) translation address register 2-63	P-R/W	All zeros	20.3.5.1.1/20-38 20.3.5.1.2/20-38
0x1FF8	PEXVFOW4TAR64—PCI Express VF outbound window 4 translation address register 64	P-R/W	All zeros	20.3.5.1.1/20-38
0x1FFC	PEXVFOW4TEAR64—PCI Express VF outbound window 4 extended translation address register 64	P-R/W	All zeros	20.3.5.1.2/20-38

20.3.2 PCI Express Configuration Access Registers

20.3.2.1 PCI Express Configuration Address Register (PEX_CONFIG_ADDR)

The PCI Express configuration address register, shown in Figure 20-2, contains address information for accesses to PCI Express internal and external configuration registers for Root Complex (RC), or End Point (EP) without SR-IOV.

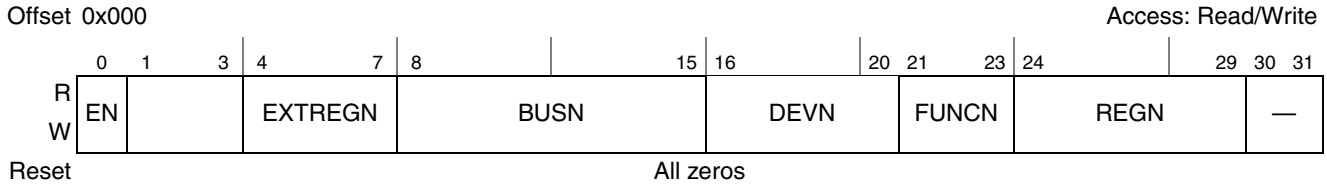


Figure 20-2. PCI Express Configuration Address Register (PEX_CONFIG_ADDR)

The fields of the PCI Express configuration address register for Root Complex (RC), or End Point (EP) without SR-IOV support are described in Table 20-4.

Table 20-4. PEX_CONFIG_ADDR Field Descriptions for RC or EP without SR-IOV

Bits	Name	Description
0	EN	Enable. This bit allows a PCI Express configuration access when PEX_CONFIG_DATA is accessed. If this bit is cleared, writing to PEX_CONFIG_DATA has no effect and reading PEX_CONFIG_DATA returns unknown data.
1–2	—	Reserved
3	—	Reserved
4–7	EXTREGN	Extended register number. This field allows access to extended PCI Express configuration space (that is, the registers in the offset range from 0x100 to 0xFFF).

Table 20-4. PEX_CONFIG_ADDR Field Descriptions for RC or EP without SR-IOV

Bits	Name	Description
8–15	BUSN	Bus number. PCI bus number to access
16–20	DEVN	Device number. Device number to access on specified bus
21–23	FUNCN	Physical Function number. Function to access within specified device
24–29	REGN	Register number. 32-bit register to access within specified device
30–31	—	Reserved

Root complex (RC) and End Point (EP without SR-IOV) configuration headers contain 4096 bytes of address space. To access a register within the header, both the extended register number and the register number fields are concatenated to form the 4-byte aligned address of the register. That is, the register address is extended register number || register number || 0b00.

The PCI Express configuration address register, shown in Figure 20-3, contains address information for accesses to PCI Express internal and external configuration registers for End Point (EP) with SR-IOV.

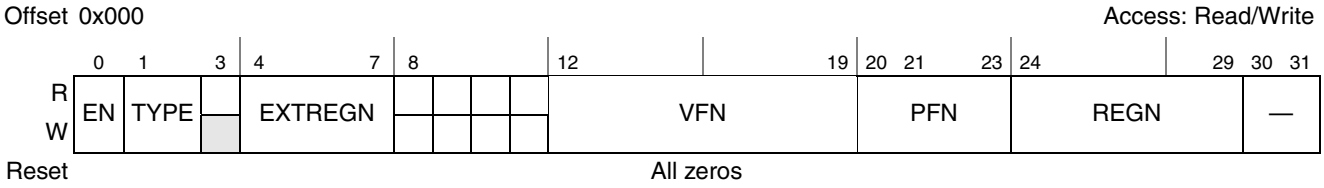


Figure 20-3. PCI Express Configuration Address Register (PEX_CONFIG_ADDR) for EP with SR-IOV

The fields of the PCI Express configuration address register for End Point (EP) with SR-IOV support are described in Table 20-5.

Table 20-5. PEX_CONFIG_ADDR Field Descriptions for EP with SR-IOV

Bits	Name	Description
0	EN	Enable. This bit allows a PCI Express configuration access when PEX_CONFIG_DATA is accessed. If this bit is cleared, writing to PEX_CONFIG_DATA has no effect and reading PEX_CONFIG_DATA returns unknown data.
1–2	TYPE	00, Configuration Register Accesses for RC and EP without SR-IOV 01, Configuration Register Accesses to PF registers for EP with SR-IOV 10, Reserved 11, Configuration Register Accesses to VF registers for EP with SR-IOV
3	—	Reserved
4–7	EXTREGN	Extended register number. This field allows access to extended PCI Express configuration space (that is, the registers in the offset range from 0x100 to 0xFFF).
8–11	—	Reserved
12–19	VFN	Virtual Function number minus 1. 64-255 is reserved.
20–23	PFN	Physical Function number minus 1. 2-15 is reserved.

Table 20-5. PEX_CONFIG_ADDR Field Descriptions for EP with SR-IOV

Bits	Name	Description
24–29	REGN	Register number. 32-bit register to access within specified device
30–31	—	Reserved

End Point (EP without SR-IOV) configuration headers contain 4096 bytes of address space for each PF. Access Type (TYPE), Physical Function (PF) number and Virtual Function (VF) number are required for the PCI-Express core to differentiate accesses to PF and VF configuration registers. To access a register within the header, both the extended register number and the register number fields are concatenated to form the 4-byte aligned address of the register. That is, the register address is extended register number || register number || 0b00.

20.3.2.2 PCI Express Configuration Data Register (PEX_CONFIG_DATA)

The PCI Express configuration data register, show in Figure 20-4, is a 32-bit port for internal and external configuration access. Note that accesses of 1, 2, or 4 bytes to the PCI Express configuration data register are allowed. Also note that accesses to the little-endian PCI Express configuration space must be properly formatted. See Section 20.4.1.2.3, "Byte Order for Configuration Transactions," for more information.

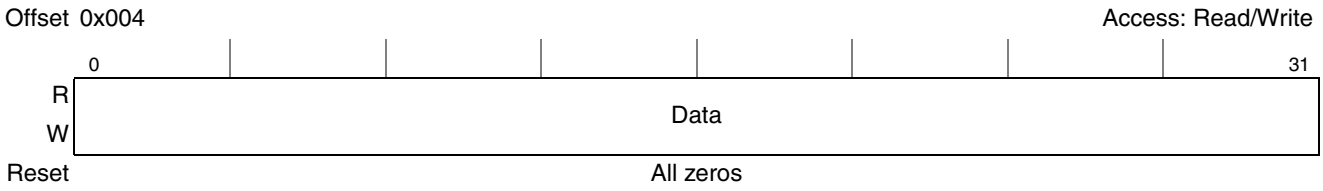


Figure 20-4. PCI Express Configuration Data Register (PEX_CONFIG_DATA)

The fields of the PCI Express configuration data register are described in Table 20-6.

Table 20-6. PEX_CONFIG_DATA Field Descriptions

Bits	Name	Description
0–31	Data	A read or write to this register starts a PCI Express configuration cycle if the PEX_CONFIG_ADDR enable bit is set (PEX_CONFIG_ADDR[EN] = 1).

20.3.2.3 PCI Express Outbound Completion Timeout Register (PEX_OTB_CPL_TOR)

The PCI Express outbound completion timeout register, shown in Figure 20-5, contains the maximum wait time for a response to come back as a result of an outbound non-posted request before a timeout condition

occurs. This timeout register is also used to timeout requests that were not acknowledged (PEX_ERR_DET[PAT]) which is a fatal error.

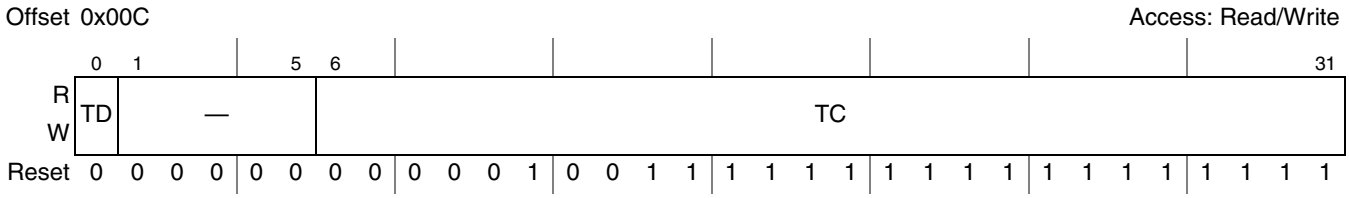


Figure 20-5. PCI Express Outbound Completion Timeout Register (PEX_OTB_CPL_TOR)

The fields of the PCI Express outbound completion timeout register are described in Table 20-7.

Table 20-7. PEX_OTB_CPL_TOR Field Descriptions

Bits	Name	Description
0	TD	Timeout disable. This bit controls the enabling/disabling of the timeout function. 0 Enable completion timeout 1 Disable completion timeout
1–5	—	Reserved
6–31	TC	Timeout counter. This is the value that is used to load the response counter of the completion timeout. The completion timeout is determined by TC and the setting of the completion timeout disable bit (CPL_TOD) in Device Control 2 Register per PF. When Device Control 2[CPL_TOD] is cleared, one TC unit is 8× the PCI Express controller clock period; that is, one TC unit is 20 ns at 400 MHz, and 30 ns at 266.66 MHz. The following are examples of timeout periods based on different TC settings: 0x00_0000Reserved 0x10_FFFF22.28 ms at 400 MHz controller clock; 33.34 ms at 266.66 MHz controller clock 0xFF_FFFF335.54 ms at 400 MHz controller clock; 503.31 ms at 266.66 MHz controller clock 0x3FF_FFFF 1.34 s at 400 MHz controller clock; 2.01 s at 266.66 MHz controller clock When Device Control 2[CPL_TOD] is set, the timeout value is based on TC and Device Control 2[CPL_TO_VAL]. When completion timeout value (CPL_TO_VAL) is: 0000 the timeout value is TC 0001 the timeout value is TC/64 0010 the timeout value is TC/4 0101 the timeout value is TC 0110 the timeout value is TC×4 1001 the timeout value is TC×16 1010 the timeout value is TC×64 For all other values of completion timeout the timeout value is TC. **Note:** The maximum computed timeout value based on TC can not exceed 1.34s at 400 MHz and 2.01s at 266.66 MHz.

20.3.2.4 PCI Express Configuration Retry Timeout Register (PEX_CONF_RTY_TOR)

The PCI Express configuration retry timeout register, shown in Figure 20-6, contains the maximum time period during which retries of configuration transactions which resulted in a CRS response occur.

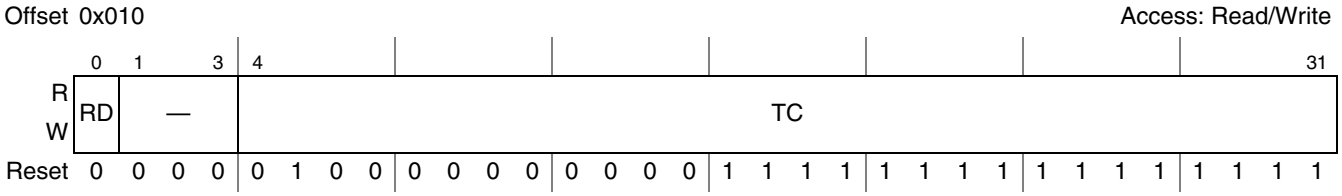


Figure 20-6. PCI Express Configuration Retry Timeout Register (PEX_CONF_RTY_TOR)

The fields of the PCI Express configuration retry timeout register are described in Table 20-8.

Table 20-8. PEX_CONF_RTY_TOR Field Descriptions

Bits	Name	Description
0	RD	Retry disable. This bit disables the retry of a configuration transaction that receives a CRS status response packet. 0 Enable retry of a configuration transaction in response to receiving a CRS status response until the timeout counter (defined by the PEX_CONF_RTY_TOR[TC] field) has expired. 1 Disable retry of a configuration transaction regardless of receiving a CRS status response.
1–3	—	Reserved
4–31	TC	Timeout counter. This is the value that is used to load the CRS response counter. One TC unit is 8× the PCI Express controller clock period; that is, one TC unit is 20 ns at 400 MHz and 30 ns at 266.66 MHz. Timeout period based on different TC settings: 0x000_0000 Reserved 0x400_FFFF 1.34 s at 400 MHz controller clock, 2.02 s at 266.66 MHz controller clock 0xFFF_FFFF 5.37 s at 400 MHz controller clock, 8.05 s at 266.66 MHz controller clock

20.3.2.5 PCI Express Configuration Register (PEX_CONFIG)

The PCI Express configuration register, shown in Figure 20-7, contains various control switches for the controller.

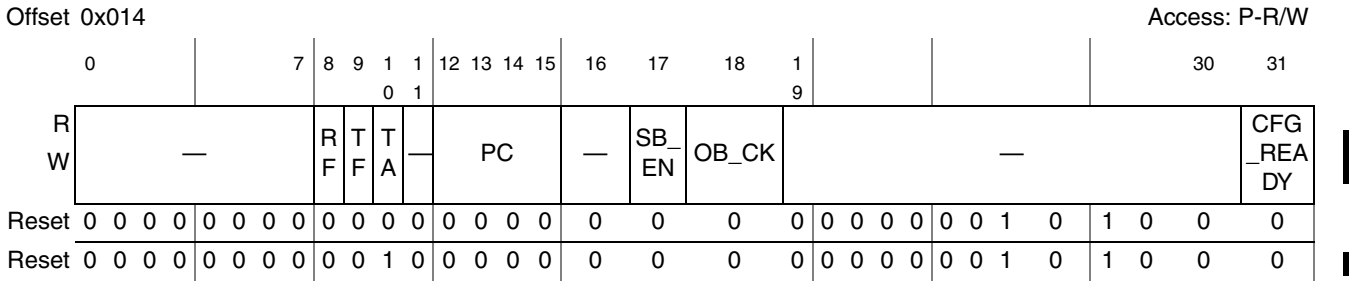


Figure 20-7. PCI Express Configuration Register (PEX_CONFIG)

The fields of the PCI Express configuration register are described in Table 20-9.

Table 20-9. PEX_CONFIG Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8	RF	Rx Flip: When set allows manual lane reversal for receive lanes. For use when automatic lane reversal does not occur because lane 0 is not detected. If used, recommend this bit to be initialized using PBL prior to link training.
9	TF	Tx Flip: When set allows manual lane reversal for transmit lanes. For use when automatic lane reversal does not occur because lane 0 is not detected. If used, recommend this bit to be initialized using PBL prior to link training.
10	TA	SR-IOV: Tx Abort: When set allows logic to terminate any outbound transactions that is targeted to a device that does not set its bus master enable bit or its D-state is non zero.When not set will make logic hold on to outbound transaction that is targeted to a device that does not set its bus master enable bit or its D-state is non zero. Non-SR-IOV: This bit is reserved.
11	—	Reserved
12–15	PC	Reserved. Default value may be non-zeros. Do not alter default value.
16	—	Reserved
17	SB_EN	Southbridge enable. Enables the ability to operate with certain southbridge devices that require being configured on bus 0. Note that this bit is for use in RC mode only; this bit must be cleared in EP mode. This bit should also be set for configuring devices that are ARI capable. 0 Configuration transactions targeting bus 0 are handled internally and are not allowed to pass to the external link. 1 Configuration transactions targeting bus 0 are allowed to pass to the external link.
18	OB_CK	Outbound transaction address checking enable. 0 Disable checking for all outbound transaction addresses (memory and I/O) against the base/limit registers. There is no checking of outbound addresses. This setting is compatible with the previous generation PCI Express controller behavior. 1 Enable checking for all outbound addresses (memory and I/O) against the base/limit registers. If an outbound transaction address does not hit into the base/limit registers, it will cause an error.
19–30	—	Reserved. (Some bits may be set by default. Software is required to preserve the setting of reserved bits)
31	CFG_READY	Configuration ready. Only applicable when in EP mode. Default setting is SOC driven . Modification of this bit by software requires a read-modify-write operation. 1 Allow all inbound configuration transactions to be processed normally 0 Retry al inbound configuration transaction

20.3.2.6 PCI Express Interrupt Status Register (PEX_INT_STAT)

The PCI Express interrupt status register, shown in Figure 20-8, shows the current interrupt sources that are raising an interrupt to the core. This register is a read only register. Clearing the interrupt source will clear the corresponding bit in the interrupt status register.

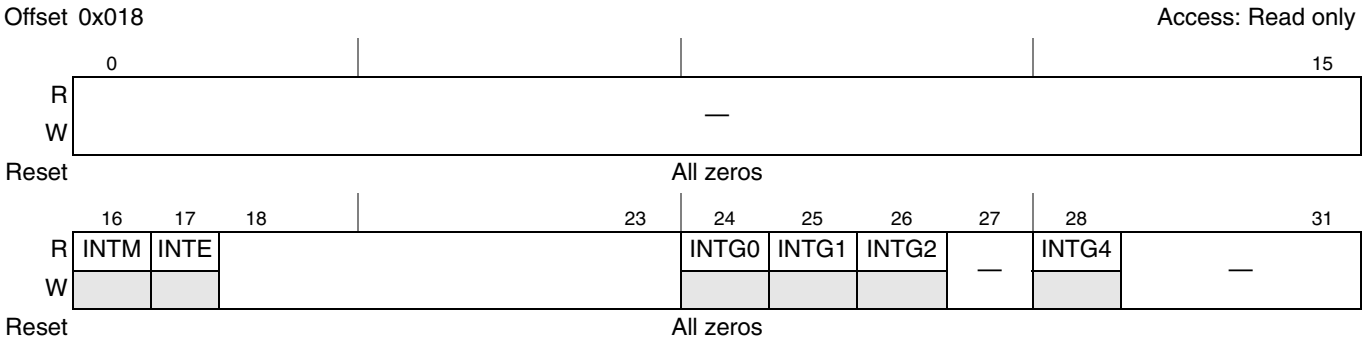


Figure 20-8. PCI Express Interrupt Source Register (PEX_INT_STAT)

The fields of the PCI Express PME and message detect register are described in Table 20-10.

Table 20-10. PEX_INT_STAT Descriptions

Bits	Name	Description
0–15	—	Reserved
16	INTM	Interrupt from PEX_PME_MES_DR register. 1 Interrupt is pending 0 No interrupt
17	INTE	Interrupt from PEX_ERR_DR register. 1 Interrupt is pending 0 No interrupt
18–23	—	Reserved
24	INTG0	Interrupt from group 0. This includes interrupt generated by: The INTx assertion disable bit in the Command register is 0 and the PME interrupt enable bit in the Root Control register is set to 1 and the PME Status bit in the Root Status register is set to 1. 1 Interrupt is pending 0 No interrupt
25	INTG1	Interrupt from group 1. This includes interrupts generated when a reported error condition causes a bit to be set in the Root Error Status register and the associated error message reporting enable bit is set in the Root Error Command register. 1 Interrupt is pending 0 No interrupt
26	INTG2	Interrupt from group 2. This includes interrupts generated from Command Status Register (Master Abort, Target Abort, etc..) 1 Interrupt is pending 0 No interrupt
27	—	Reserved

Table 20-10. PEX_INT_STAT Descriptions

Bits	Name	Description
28	INTG4	Interrupt from group 4. This includes interrupts generated when the Link Autonomous Bandwidth Status register (Link Status register bit 15) is updated and the Link Autonomous Bandwidth Interrupt Enable (Link Control register bit 11) is set or the Link Bandwidth Management Status register (Link Status register bit 14) is updated and the Link Bandwidth Management Interrupt Enable (Link Control register bit 10) is set or the Link Equalization Request bit in the Link Status 2 Register has been set. 1 Interrupt is pending 0 No interrupt
29–31	—	Reserved

20.3.3 PCI Express Power Management Event and Message Registers

20.3.3.1 PCI Express PME and Message Detect Register (PEX_PME_MES_DR)

The PCI Express PME and message detect register, shown in Figure 20-9, logs inbound messages and PME events that are detected by the PCI Express controller. This register is a write-1-to-clear type register.

Figure 20-9. PCI Express PME and Message Detect Register (PEX_PME_MES_DR)

The fields of the PCI Express PME and message detect register are described in Table 20-11.

Table 20-11. PEX_PME_MES_DR Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16	PTO	PME turn off. This bit indicates the detection of a PME_Turn_Off message. This bit is only valid in EP mode. 1 A PME_Turn_Off_message is detected 0 No PME_Turn_Off message detected
17	—	Reserved. Note that during normal operation, this bit may be set (falsely). The bit may be ignored and cleared (w1c) without consequence.

Table 20-11. PEX_PME_MES_DR Field Descriptions (continued)

Bits	Name	Description
18	ENL23	Entered L2/L3 ready state. This bit indicates that the PCI Express controller has entered L2/L3 state. This is only valid in RC mode. 1 L2/L3 ready state has been entered 0 L2/L3 ready state has not been entered
19	EXL23	Exit L2/L3 ready state. This bit indicates that the PCI Express controller has exited the L2/L3 state. This is only valid in RC mode. 1 Exit L2/L3 state has been detected 0 Exit L2/L3 state not detected
20	—	Reserved. Note that during normal operation, this bit may be set (falsely). The bit may be ignored and cleared (w1c) without consequence.
21	HRD	Hot reset detected. This bit indicates that the PCI Express controller has detected a hot reset condition on the link. The controller is reset and cleans up all outstanding transactions. Link retraining takes place once hot reset state is exited. This is valid only in EP mode. 1 Hot reset request has been detected 0 Hot reset request not detected
22	LDD	Link down detected. This bit indicates that a link down condition has been detected. The controller is reset and then cleans up all outstanding transactions. Link retraining takes place once the controller has cleaned itself up. Note that for EP, this bit and HRD are typically set when a hot reset event is detected. 1 Link down has been detected 0 Link down not detected
23	AILDD	Ack-timeout induced link down detected. This bit indicates that a link down condition has been detected due to an internal timeout condition. The controller is reset and then cleans up all outstanding transactions. Link retraining takes place once the controller has cleaned itself up. No LDD will be set if this bit is set. 1 Ack timeout link down has been detected 0 Ack timeout link down not detected
24	LUD	Link up detected. This bit indicates that a link up condition has been detected. It means that link training has been successful. 1 Link up detected 0 Link up not detected
25–31	—	Reserved. Note that during normal operation, these bits may be set (falsely). The bits may be ignored and cleared (w1c) without consequence.

20.3.3.2 PCI Express PME and Message Disable Register (PEX_PME_MES_DISR)

The PCI Express PME and message disable register, shown in Figure 20-10, when set, prevents the detection of the corresponding bits in the PCI Express PME and message detect register.

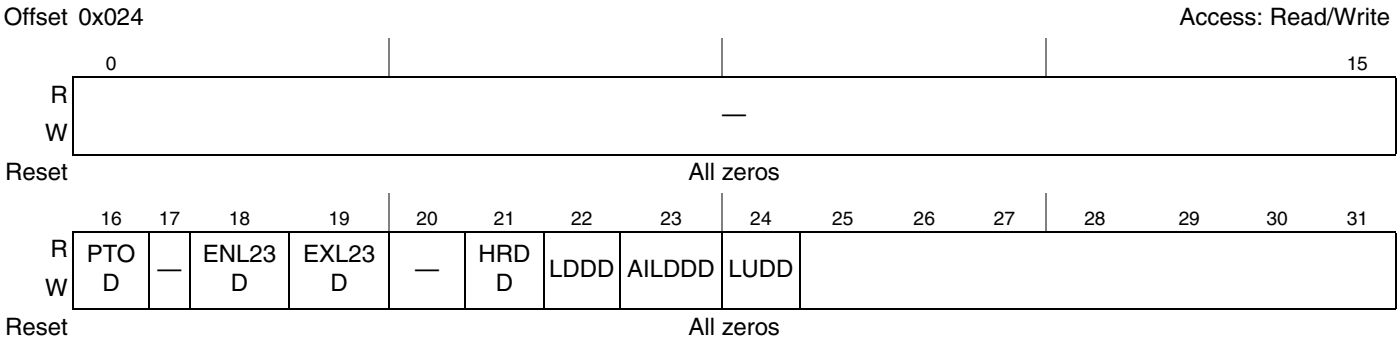


Figure 20-10. PCI Express PME and Message Disable Register (PEX_PME_MES_DISR)

The fields of the PCI Express PME and message disable register are described in Table 20-12.

Table 20-12. PEX_PME_MES_DISR Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16	PTOD	PME turn off disable. When set, this bit disables the setting of PEX_PME_MES_DR[PTO] bit. 1 Disable PME_Turn_Off_message detection 0 Enable PME_Turn_Off message detection
17	—	Reserved
18	ENL23D	Entered_L2/L3 ready disable. When set, this bit disables the setting of PEX_PME_MES_DR[ENL23] bit. 1 Disable Entered_L2/L3 ready state detection 0 Enable Entered_L2/L3 ready state detection
19	EXL23D	Exited_L2/L3 ready disable. When set, this bit disables the setting of PEX_PME_MES_DR[EXL23] bit. 1 Disable Exited_L2/L3 ready state detection 0 Enable Exited_L2/L3 ready state detection
20	—	Reserved
21	HRDD	Hot reset detected disable. When set, this bit disables the setting of PEX_PME_MES_DR[HRD] bit. 1 Disable hot reset state detection 0 Enable hot reset state detection
22	LDDD	Link down detected disable. When set, this bit disables the setting of PEX_PME_MES_DR[LDD] bit. 1 Disable link down state detection 0 Enable link down state detection
23	AILDDD	Ack time-out induced link down detected disable. When set, this bit disables the setting of PEX_PME_MES_DR[AILDD] bit. 1 Disable ack time-out link down state detection 0 Enable ack time-out link down state detection
24	LUDD	Link up detected disable. When set, this bit disables the setting of PEX_PME_MES_DR[LUD] bit. 1 Disable link up detection 0 Enable link up detection
25-31	—	Reserved

20.3.3.3 PCI Express PME and Message Interrupt Enable Register (PEX_PME_MES_IER)

The PCI Express PME and message interrupt enable register, shown in Figure 20-11, allows for the detection of a message or a PME event to generate an interrupt, provided that the corresponding bit in the PCI Express PME and message detect register is set.

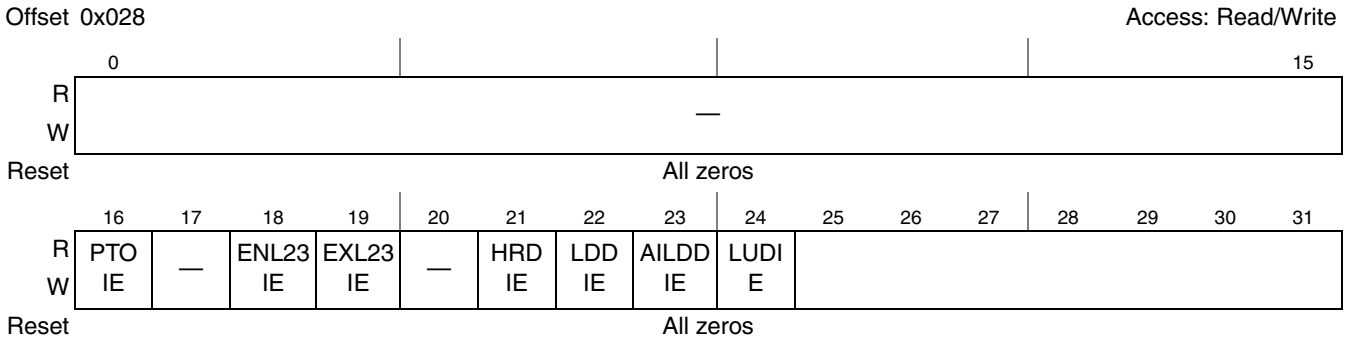


Figure 20-11. PCI Express PME and Message Interrupt Enable Register (PEX_PME_MES_IER)

Table 20-13 shows the fields of the PCI Express PME and message interrupt enable register.

Table 20-13. PEX_PME_MES_IER Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16	PTOIE	PME turn off interrupt enable. When set and PEX_PME_MES_DR[PTO]=1 generates an interrupt. 1 Enable PME_Turn_Off_message interrupt generation 0 Disable PME_Turn_Off message interrupt generation
17	—	Reserved
18	ENL23IE	Entered L2/L3 ready interrupt enable. When set and PEX_PME_MES_DR[ENL23]=1 generates an interrupt. 1 Enable Entered_L2/L3 ready state interrupt generation 0 Disable Entered_L2/L3 ready state interrupt generation
19	EXL23IE	Exited L2/L3 ready interrupt enable. When set and PEX_PME_MES_DR[EXL23]=1 generates an interrupt. 1 Enable Exited_L2/L3 ready state interrupt generation 0 Disable Exited_L2/L3 ready state interrupt generation
20	—	Reserved
21	HRDIE	Hot reset detected interrupt enable. When set and PEX_PME_MES_DR[HRD]=1 generates an interrupt. 1 Enable hot reset state interrupt generation 0 Disable hot reset state interrupt generation
22	LDDIE	Link down detected interrupt enable. When set and PEX_PME_MES_DR[LDD]=1 generates an interrupt. 1 Enable link down state interrupt generation 0 Disable link down state interrupt generation
23	AILDDIE	Ack time-out induced link down detected interrupt enable. When set and PEX_PME_MES_DR[AILDD]=1 generates an interrupt. 1 Enable ack time-out link down state interrupt generation 0 Disable ack time-out link down state interrupt generation

Table 20-13. PEX_PME_MES_IER Field Descriptions (continued)

Bits	Name	Description
24	LUDIE	Link up detected interrupt enable. When set and PEX_PME_MES_DR[LUD]=1 generates an interrupt. 1 Enable link up detected interrupt generation 0 Disable link up detected interrupt generation
25-31	—	Reserved

20.3.3.4 PCI Express Power Management Command Register (PEX_PMCR)

The PCI Express power management command register, shown in Figure 20-12, provides software a mechanism to allow the PCI Express controller to get back to L0 link state.

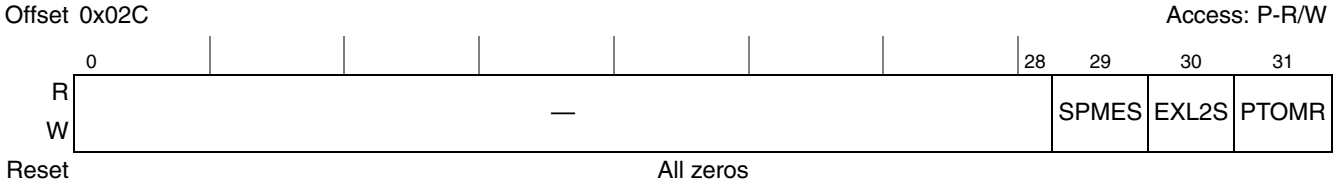


Figure 20-12. PCI Express Power Management Command Register (PEX_PMCR)

The fields of the PCI Express power management command register are described in Table 20-14.

Table 20-14. PEX_PMCR Field Descriptions

Bits	Name	Description
0–28	—	Reserved
29	SPMES	Set PME status. This sets the PME status bit and if PME is enabled (see Section 20.3.10.3, "PCI Express Power Management Status and Control Register—0x44," on page 20-115 for more information) it transmits a PM_PME message upstream. This bit should not be used when in RC mode. This bit is self-clearing.
30	EXL2S	Exit L2 state. When set exits the link state out of L2/L3 ready state in order to send new requests. The request is only made when entered_L2/L3 ready state is active. This bit is self-clearing. When the link has exited L2/L3 ready state, the status bit Exit_L2/L3 ready state is set. This bit should not be used when in EP mode. This bit is shared among PFs. In RC mode, Exit L2 clears both sticky and non-sticky register bits.
31	PTOMR	PME_Turn_Off message request. When set broadcasts a PME turn_off message. This bit should not be used when in EP mode. This bit is self-clearing. This bit is unique to each PF (that is, per PF).

20.3.3.5 PCI Express LIODN Base Register (PEX_LBR)

The PCI Express LIODN base register, shown in Figure 20-13, provides the base LIODN that all transactions to memory (except for Inbound MSI window hits) must use. The actual LIODN that is

associated with a transaction sent to memory is PEX_LBR + LIODN offset that is derived from the LIODN permission table. See Section 20.4.10, "LIODN Permission Table," for more information.

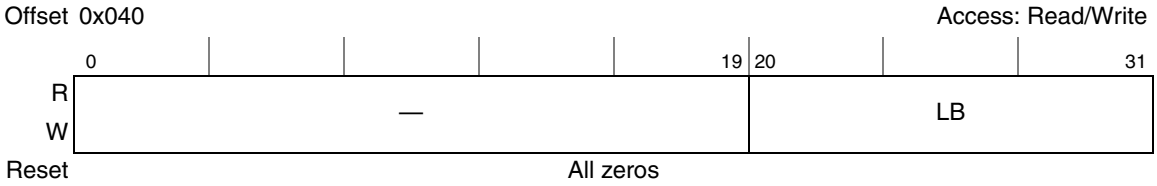


Figure 20-13. PCI Express LIODN Base Register (PEX_LBR)

The fields of the PCI Express LIODN base register are described in Table 20-15.

Table 20-15. PEX_LBR Field Descriptions

Bits	Name	Description
0-19	—	Reserved
20-31	LB	LIODN Base. This field contains the 12-bit LIODN base that is used for permission identification for all inbound transactions toward mermory.

20.3.3.6 PCI Express MSI LIODN Base Register (PEXMSI_LBR)

The PCI Express MSI LIODN base register, shown in Figure 20-14, provides the base LIODN that Inbound MSI transactions to memory must use. The actual LIODN that is associated with a Inbound MSI transaction sent to memory is PEXMSI_LBR + LIODN offset that is derived from the LIODN permission table. See Section 20.4.10, "LIODN Permission Table," for more information.

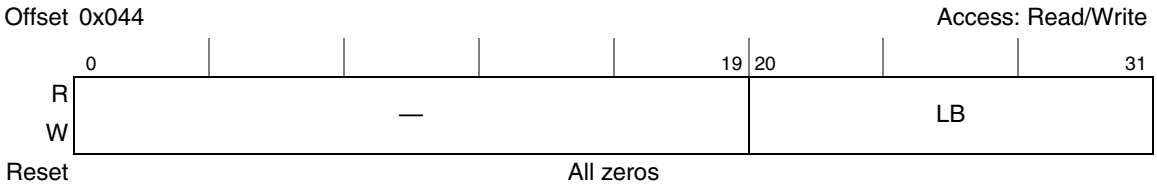


Figure 20-14. PCI Express LIODN Base Register (PEX_LBR)

The fields of the PCI Express MSI LIODN base register are described in Table 20-16.

Table 20-16. PEXMSI_LBR Field Descriptions

Bits	Name	Description
0-19	—	Reserved
20-31	LB	LIODN Base. This field contains the 12-bit LIODN base that is used for permission identification for all inbound transactions toward mermory.

20.3.3.7 PCI Express Address Offset Register (PEX_AOR)

The PCI Express address offset register, shown in Figure 20-15, provides address index into the LIODN permission table, or MSI-X/PBA table.

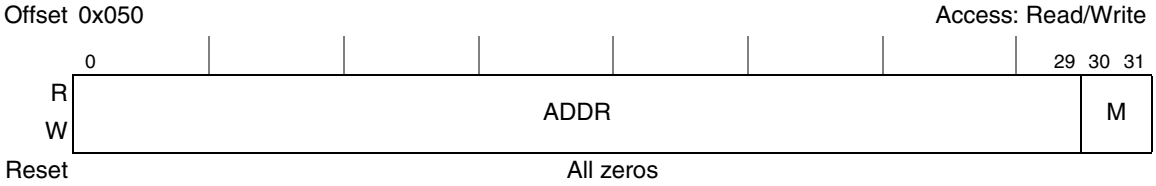


Figure 20-15. PCI Express Address Offset Register (PEX_AOR)

The fields of the PCI Express address offset register are described in Table 20-17.

Table 20-17. PEX_AOR Field Descriptions

Bits	Name	Description
0–29	ADDR	Address. This field takes on different meanings depending on the setting of M (Mode) field.
30-31	M	Mode. Depending on setting will take on different meaning. 00 Reserved 01 LIODN. Part of the ADDR fields is used to access LIODN permission table 10 MSI-X Vector Table. Part of the ADDR field is used to access MSI-X table structure. 11 MSI-X PBA Array. Part of the ADDR field is used to access MSI-X PBA structure.

20.3.3.7.1 PEX_AOR[M]=01

When PEX_AOR[M] = 01, this register contains the address offset into the LIODN permission table.

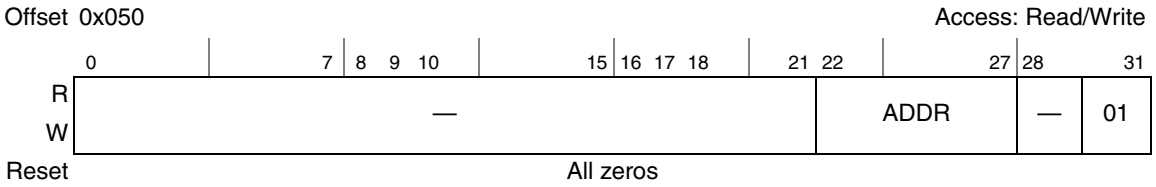


Figure 20-16. PCI Express Address Offset Register when M=01

The fields of the PCI Express address offset register are described in Table 20-18 when M=01.

Table 20-18. PEX_AOR Field Descriptions when M=01

Bits	Name	Description
0–21	—	Reserved.
22-27	ADDR	Address. This field provides the address offset into the LIODN table. Each entry in the table is 64-bits.
28-29	—	Reserved.
30-31	M	Mode = 01

20.3.3.7.2 PEX_AOR[M]=10

When PEX_AOR[M] = 10, this register contains the address offset into the MSI-X vector table structure. Note that data will be byte-swapped.

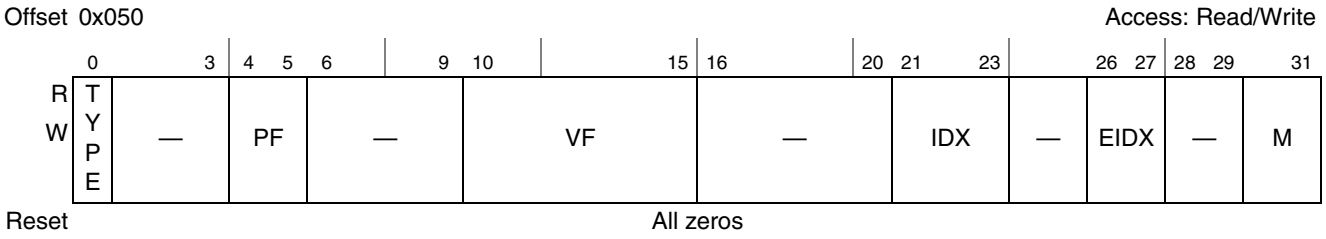


Figure 20-17. PCI Express Address Offset Register when M=10

The fields of the PCI Express address offset register are described in Table 20-19 when M=10.

Table 20-19. PEX_AOR Field Descriptions when M=10

Bits	Name	Description
0	TYPE	0, Access to PF MSI-X vector table for EP with SR-IOV. 1, Access to VF MSI-X vector table for EP with SR-IOV.
1–3	—	Reserved.
4-5	PF	Physical Function
6-9	—	Reserved.
10-15	VF	Virtual Function
16-20	—	Reserved.
21-23	IDX	MSI-X Entry Index in each VF.
24-25	—	Reserved.
26-27	EIDX	Extended index. This field provides which 4-byte entity within the MSI-X table entry to access. 00 First 4 bytes (lower 32-bit vector address) 01 Second 4 bytes (upper 32-bit vector address) 10 Third 4-bytes (vector data) 11 Fourth 4 bytes (vector control)
28-29	—	Reserved.
30–31	M	Mode=10

To access a MSI-X vector table, the PF, VF, IDX, EIDXare concatenated to form the 4-byte aligned address of the register within the MSI-X vector table. That is, the register address is

PF || VF || IDX || EIDX || 0b00.

20.3.3.7.3 PEX_AOR[M]=11

When PEX_AOR[M] = 11, this register contains the address offset into the MSI-X PBA structure. Note that data will be byte-swapped.

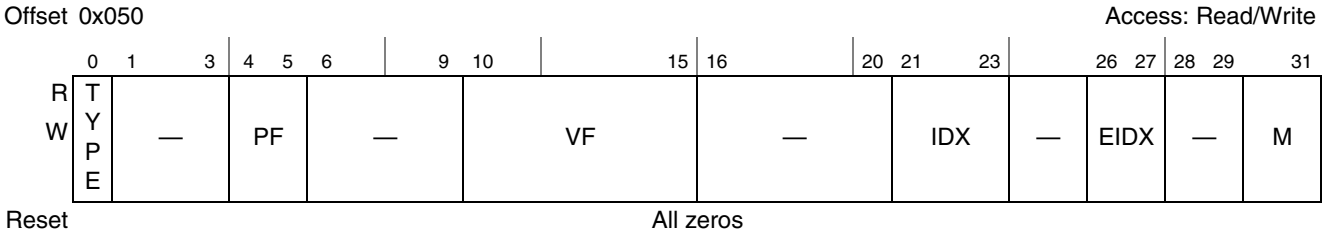


Figure 20-18. PCI Express Address Offset Register when M=11

The fields of the PCI Express address offset register are described in Table 20-20 when M=11.

Table 20-20. PEX_AOR Field Descriptions when M=11

Bits	Name	Description
0	TYPE	0, Access to PF MSI-X vector table for EP with SR-IOV. 1, Access to VF MSI-X vector table for EP with SR-IOV.
1–3	—	Reserved.
4-5	PF	Physical Function
6-9	—	Reserved.
10-15	VF	Virtual Function
16-20	—	Reserved.
21-23	IDX	MSI-X Entry Index in each VF.
24-25	—	Reserved.
26-27	EIDX	Extended index. This field provides which 4-Byte entity within the MSI-X PBA structure to access. 00 - PBA bits for 0-31 MSI-X vector of the function. 01 - PBA bits for 32-63 MSI-X vector of the function. 10 - PBA bits for 64-95 MSI-X vector of the fucnction. 11 - PBA bits for 96-127 MSI-X vector of the function.
28-29	—	Reserved.
30–31	M	Mode=11

To access a MSI-X PBA structure, the PF, VF, IDX, EIDXare concatenated to form the 4-byte aligned address of the register within the MSI-X PBA structure. That is, the register address is

PF || VF || IDX || EIDX || 0b00.

The 4-byte data returned is the pending bit status for 32 MSI-X vectors corresponding to the EIDX field.

20.3.3.8 PCI Express Upper Data Register (PEX_UDR)

The PCI Express upper data register, shown in Figure 20-19, allows software to read/write into the upper 32-bit of the LIODN permission table's entry.

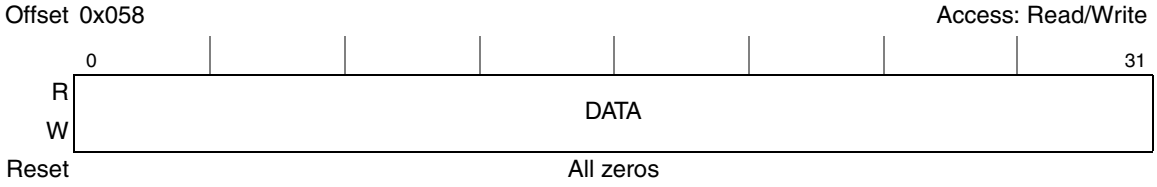


Figure 20-19. PCI Express Upper Data Register (PEX_UDR)

The fields of the PCI Express upper data register are described in Table 20-21.

Table 20-21. PEX_UDR Field Descriptions

Bits	Name	Description
0–31	DATA	PEX_AOR[M] = 01, upper 32-bit entity of the LIODN permission table's entry. PEX_AOR[M] = 00, 10, 11 are reserved.

20.3.3.9 PCI Express Lower Data Register (PEX_LDR)

The PCI Express lower data register, shown in Figure 20-20, allows software to read/write into the lower 32-bits of the LIODN permission table's entry or MSI-X vector/PBA table's entry.

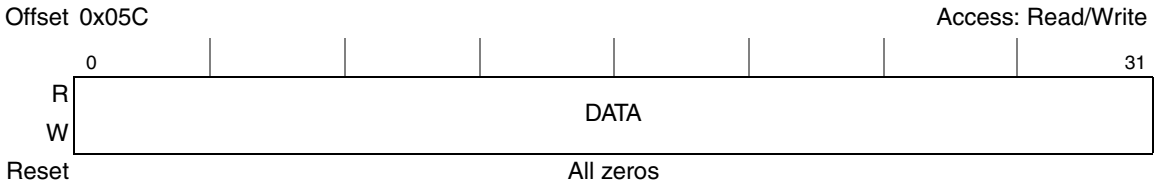


Figure 20-20. PCI Express Lower Data Register (PEX_LDR)

The fields of the PCI Express lower data register are described in Table 20-22.

Table 20-22. PEX_LDR Field Descriptions

Bits	Name	Description
0–31	DATA	PEX_AOR[M] = 00, reserved. PEX_AOR[M] = 01, lower 32-bit entity of the LIODN permission table's entry. PEX_AOR[M] = 10, 4-byte data of the MSI-X vector corresponding to PEX_AOR[ADDR] field. PEX_AOR[M] = 11, 4-byte data of the MSI-X PBA data corresponding to PEX_AOR[ADDR] field.

20.3.4 PCI Express IP Block Revision Registers

20.3.4.1 IP Block Revision Register 1 (PEX_IP_BLK_REV1)

The IP block revision register 1 is shown in Figure 20-21.

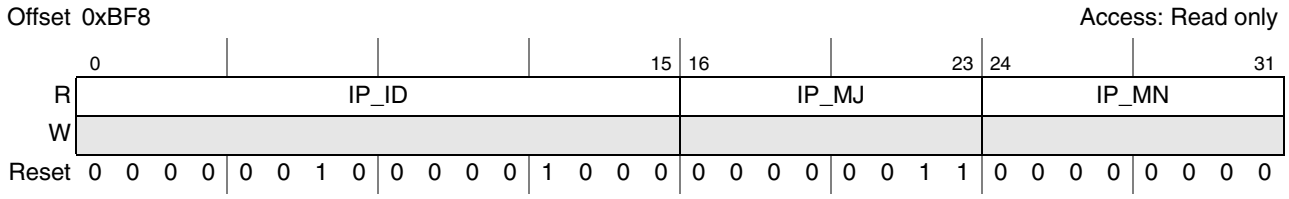


Figure 20-21. IP Block Revision Register 1

Table 20-23 contains descriptions of the fields of the IP block revision register 1.

Table 20-23. PCI Express IP Block Revision Register 1 Field Descriptions

Bits	Name	Description
0–15	IP_ID	Block ID
16–23	IP_MJ	Block Major Revision
24–31	IP_MN	Block Minor Revision

20.3.4.2 IP Block Revision Register 2 (PEX_IP_BLK_REV2)

The IP block revision register 2 is shown in Figure 20-22.

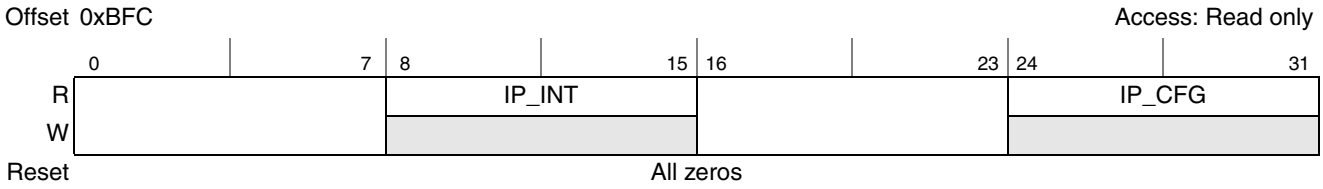


Figure 20-22. IP Block Revision Register 2

Table 20-24 contains descriptions of the fields of the IP block revision register 2.

Table 20-24. PCI Express IP Block Revision Register 2 Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8–15	IP_INT	Block integration option
16–23	—	Reserved
24–31	IP_CFG	Block configuration option

20.3.5 PCI Express ATMU Registers

Software should initialize the appropriate PCI Express ATMU registers and associated configuration registers to allow inbound and outbound flow of traffic to be routed properly to and from memory.

Table 20-25 describes the available programming options depending on whether the device is in RC or EP mode.

Table 20-25. ATMU and configuration registers

		Non SR-IOV device		SR-IOV device	
		RC	EP	RC	EP (per PF)
Inbound	ATMU	• PEXITAR0-3 • PEXIBAR1-3 • PEXIBEAR2-3 • PEXIWAR0-3 • PEXMSIITAR • PEXMSIIWAR • PEXEPROMITAR • PEXEPROMIWAR	• PEXITAR0-3 • PEXIWAR0-3 • PEXEPROMITAR • PEXEPROMIWAR	• PEXITAR0-3 • PEXIBAR1-3 • PEXIBEAR2-3 • PEXIWAR0-3 • PEXMSIITAR • PEXMSIIWAR • PEXEPROMITAR • PEXEPROMIWAR	• PEXITAR0-3 • PEXIWAR0-3 • PEXEPROMITAR • PEXEPROMIWAR • PEXVFITAR0-3 • PEXVFIWAR0-3
	Configuration	• PEXCSRBAR (Type 1 header, offset 0x10) • PCI Express Memory Base Register (Type 1 header, offset 0x20) • PCI Express Memory Limit (Type 1 header, offset 0x22) • PCI Express Prefetchable Memory Base Register (Type 1 header, offset 0x24) • PCI Express Prefetchable Memory Limit Register (Type 1 header, offset 0x26) • PCI Express Expansion ROM Base Register (Type 1 header, offset 0x38)	• PEXCSRBAR (Type 0 header, offset 0x10) • PCI Express Memory Base Registers (Type 0 header, offset 0x14-0x24) • PCI Express Expansion ROM Base Reigister (Type 0 header, offset 0x30)	• PEXCSRBAR (Type 1 header, offset 0x10) • PCI Express Memory Base Register (Type 1 header, offset 0x20) • PCI Express Memory Limit (Type 1 header, offset 0x22) • PCI Express Prefetchable Memory Base Register (Type 1 header, offset 0x24) • PCI Express Prefetchable Memory Limit Register (Type 1 header, offset 0x26) • PCI Express Expansion ROM Base Register (Type 1 header, offset 0x38)	• PEXCSRBAR (Type 0 header, offset 0x10) • PCI Express Memory Base Registers (Type 0 header, offset 0x14-0x24) • PCI Express Expansion ROM Base Reigister (Type 0 header, offset 0x30) • PCI Express SR-IOV VF Memory Base Registers (Type 0 header, offset 0x174-0x188)

Table 20-25. ATMU and configuration registers

Outbound	**ATMU**	• PEXOBAR1-4 • PEXOTAR1-4 • PEXOTEAR1-4 • PEXOWAR1-4	• PEXOBAR1-4 • PEXOTAR1-4 • PEXOTEAR1-4 • PEXOWAR1-4	• PEXOBAR1-4 • PEXOTAR1-4 • PEXOTEAR1-4 • PEXOWAR1-4	• PEXOBAR1-4 • PEXOTAR1-4 • PEXOTEAR1-4 • PEXOWAR1-4 • PEXVFOBAR1-4 • PEXVFOTAR1-64 • PEXVFOTEAR1-64 • PEXVFOWAR1-4
	Configuration	• PCI Express Memory Limit (Type 1 header, offset 0x22) • PCI Express Prefetchable Memory Base Register (Type 1 header, offset 0x24) • PCI Express Prefetchable Memory Limit Register (Type 1 header, offset 0x26) • PCI Express Expansion ROM Base Register (Type 1 header, offset 0x38)		• PCI Express Memory Limit (Type 1 header, offset 0x22) • PCI Express Prefetchable Memory Base Register (Type 1 header, offset 0x24) • PCI Express Prefetchable Memory Limit Register (Type 1 header, offset 0x26) • PCI Express Expansion ROM Base Register (Type 1 header, offset 0x38)	

20.3.5.1 PCI Express Outbound ATMU Registers

The outbound address translation windows must be aligned based on the granularity selected by the size fields. For SR-IOV VF windows, the base address must be aligned based on the total VF size calculated by multiplying the size field with the number of active VFs. Outbound transactions should hit into one of outbound address ranges defined by PEXOWARx/PEXVFOWWARx. Outbound window misses use the default outbound register set (outbound ATMU window 0). Overlapping outbound windows (1–4) are not supported and will cause undefined behavior. Note that for RC mode, all outbound transactions post ATMU must hit either into the memory base/limit range or the prefetchable memory base/limit range defined in the PCI Express type 1 header. For EP mode, there is no such requirement. There is also an MSI-X trap window which generates an MSI-X transaction when hit into. This window is only applicable for SR-IOV device operating in EP mode.

Note that in RC mode, when PEX_CONFIG[OB_CK] is set, the translated address of an outbound transaction must fall into either the memory base/limit range, the prefetchable memory base/limit range, or I/O base/limit range. The transaction will be terminated if the translated address does not fall within these ranges.

Figure 20-23 shows the outbound transaction flow.

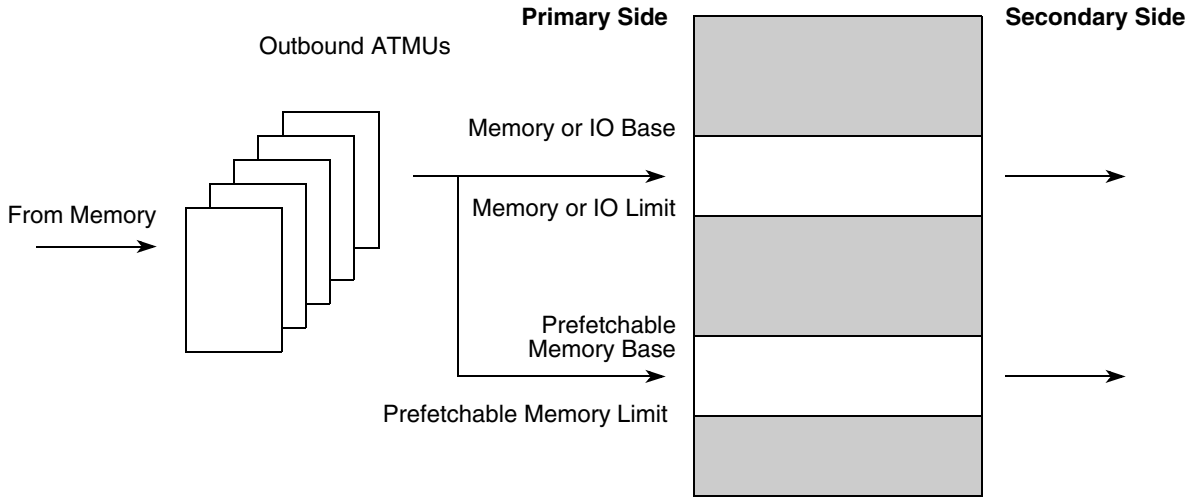


Figure 20-23. RC Outbound Transaction Flow

For SR-IOV implementation in EP mode, there needs to be a mechanism to map outbound addresses into a particular VF/PF space. This is done through the outbound ATMU translation windows. Each PF will have its own ATMU unit in separate 8K memory-mapped register regions. When an outbound transaction is passing through the ATMU windows, all ATMU units are compared. Depending on which ATMU region of address the outbound transaction is hitting into, a particular VF/PF is chosen. In case if no hit occurs, then the default outbound window that corresponds to PF 0 is chosen.

All outbound ATMU registers dedicated to PF address decode occupy memory-mapped offset 0xC00-0xCFC. Note that this condition only applies if in EP mode.

All the VFs in a PF share a set of the VF Outbound Window Base Address Register (PEXVFOWBAR1-4) and VF Outbound Window Attribute Register (PEXVFOWAR1-4) in each PF at memory-mapped offset 0x820-0x8FC.

However, each VF in a PF has its own VF Outbound Window Translation Address Registers (PEXVFOWnTAR1-64) and VF Outbound Window Extended Translation Address Registers (PEXVFOWnTEAR1-64).

Note that the size of a particular BAR for all the VFs in a PF is determined by PEXVFOWARn[OWS]. The total BAR memory size is determined by PEXVFOWARn[OWS] times the total number of VFs supported (i.e 64). For example, a VF bar 0 in PF 0 is configured to have a size of 4k and starting address is 0x0000_0000. Then the address mapping to a VF is done as follow.

- Address from 0x0000_0000 to 0x0000_0FFF maps to PF 0, VF 1, Bar0
- Address from 0x0000_1000 to 0x0000_1FFF maps to PF 0, VF 2, Bar0
- Address from 0x0000_2000 to 0x0000_2FFF maps to PF 0, VF 3, Bar0
- ...

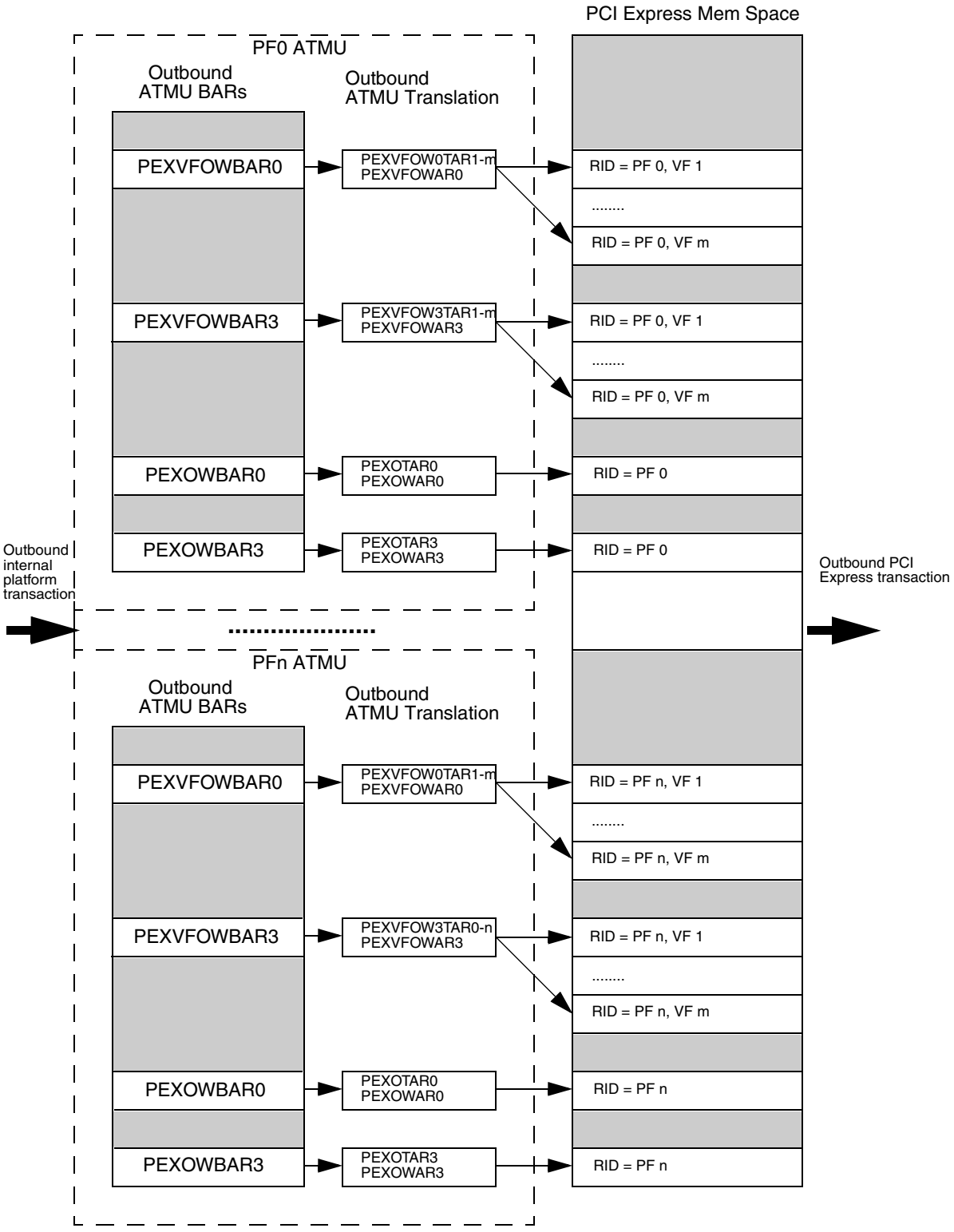


Figure 20-24. SR-IOV Outbound Transaction Flow - EP only

20.3.5.1.1 PCI Express VF Outbound Window Translation Address Registers (PEXVFOW*n*TAR*m*)—EP Mode Only

The PCI Express VF outbound translation address registers, shown in Figure 20-25, select the starting addresses in the system address space for window hits within the PCI Express outbound address translation windows. The new translated address is created by concatenating the transaction offset to this translation address. n indicates the Outbound Window number and m indicates the Virtual Function number.

Offset Window 1: 0x1800-0x19FC incrementing by 8 (for example, 0x1800, 0x1808, 0x1810,...) Access: P-R/W
Window 2: 0x1A00-0x1BFC incrementing by 8 (for example, 0x1A00, 0x1A08, 0x1A10,...)
Window 3: 0x1C00-0x1DFC incrementing by 8 (for example, 0x1C00, 0x1C08, 0x1C10,...)
Window 4: 0x1E00-0x1FFC incrementing by 8 (for example, 0x1E00, 0x1E08, 0x1E10,...)

	0	11	12	31
R W	TEA		TA	
Reset	All zeros			

Figure 20-25. PCI Express VF Outbound Window n Translation Address Registers m (PEXVFOW*n*TAR*m*)

Table 20-26 describes the fields of the PCI Express VF outbound translation address registers.

Table 20-26. PEXVFOWnTAR*m* Field Descriptions

Bits	Name	Description
0–11	TEA	Translation extended address. System address which indicates the starting point of the outbound translated address. The translation address must be aligned based on the total VF size calculated from the size field. Corresponds to PCI Express address bits [43:32] (bit 32 is the lsb).
12–31	TA	Translation address. System address which indicates the starting point of the outbound translated address. The translation address must be aligned based on the total VF size calculated from the size field. This corresponds to PCI Express address bits [31:12].

20.3.5.1.2 PCI Express VF Outbound Window Translation Extended Address Registers (PEXVFOW*n*TEAR*m*)—EP Mode Only

The PCI Express VF outbound translation extended address registers, shown in Figure 20-26, contain the most-significant bits of a 64 bit translation address.

Offset Window 1: 0x1804-0x19FC incrementing by 8 (for example, 0x1804, 0x180C, 0x1814,...) Access: P-R/W
Window 2: 0x1A04-0x1BFC incrementing by 8 (for example, 0x1A04, 0x1A0C, 0x1A14,...)
Window 3: 0x1C04-0x1DFC incrementing by 8 (for example, 0x1C04, 0x1C0C, 0x1C14,...)
Window 4: 0x1E04-0x1FFC incrementing by 8 (for example, 0x1E04, 0x1E0C, 0x1E14,...)

	0	11	12	31
R W	—		TEA	
Reset	All zeros			

Figure 20-26. PCI Express VF Outbound Window n Translation Extended Address Registers m (PEXVFOW*n*TEAR*m*)

Table 20-27 describes the fields of the PCI Express VF outbound translation extended address registers.

Table 20-27. PCI Express VF Outbound Window n Extended Address Translation Register *m* Field Descriptions

Bits	Name	Description
0–11	—	Reserved
12–31	TEA	Translation extended address. System address which indicates the starting point of the outbound translated address. The translation address must be aligned based on the total VF size calculated from the size field. Corresponds to PCI Express address bits [63:44].

20.3.5.1.3 PCI Express VF Outbound Window Base Address Registers (PEXVFOWBAR*n*)—EP Mode Only

The PCI Express VF outbound window base address registers, shown in Figure 20-27, select the base address for the windows which are translated to the external address space. Addresses for outbound transactions are compared to these windows. If such a transaction does not fall within one of these spaces the transaction is forwarded through a default register set. These registers are unique per PF.

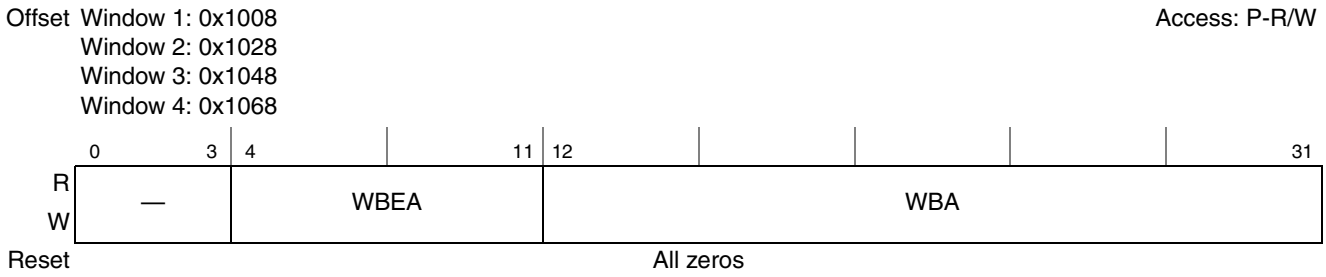


Figure 20-27. PCI Express VF Outbound Window Base Address Registers (PEXVFOWBAR*n*)

Table 20-28 describes the fields of the PCI Express VF outbound window base address registers.

Table 20-28. PCI Express VF Outbound Window Base Address Register *n* Field Descriptions

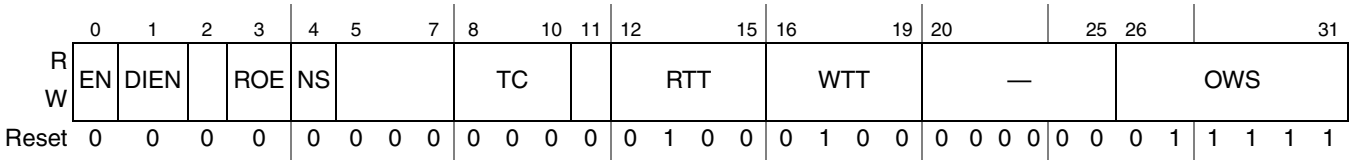
Bits	Name	Description
0–3	—	Reserved
4–11	WBEA	Window base extended address. Source address which is the starting point for the outbound translation window. The window must be aligned based on the total VF window size calculated from the window size (PEXVFOWARn[OWS] × number of VFs). Correspond to internal platform address bits [0:7]. (where 0 is the msb of the internal platform address)
12–31	WBA	Window base address. Source address which is the starting point for the outbound translation window. The window must be aligned based on the total VF window size calculated from the window size bits. This corresponds to internal platform address bits [8:27].

20.3.5.1.4 PCI Express VF Outbound Window Attributes Registers (PEXVFOWAR*n*)—EP Mode Only

The PCI Express VF outbound window attributes registers, shown in Figure 20-28, define the window sizes to translate and other attributes for the translations. Figure 20-28 shows the PCI Express VF outbound window attributes registers (PEXVFOWAR*n*). These registers are unique per PF.

Offset Window 1: 0x1010
Window 2: 0x1030
Window 3: 0x1050
Window 4: 0x1070

Access: P-R/W



	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	EN		DIEN		ROE	NS			TC				RTT				WTT				—						OWS					
Reset	0		0		0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1

Figure 20-28. PCI Express VF Outbound Window Attributes Registers (PEXVFOWAR*n*)

Table 20-29 describes the fields of the PCI Express VF outbound window attributes registers.

Table 20-29. PEXVFOWAR*n* Field Descriptions

Bits	Name	Description
0	EN	Enable. This bit enables this address translation window. For the default window, this bit is read-only and always hardwired to 1. 0 Disable outbound translation window 1 Enable outbound translation window
1	DIEN	Data invariance enable. This bit controls the data ordering policy of all outbound transactions passing through this window. See Section 20.4.1.2, "Byte Ordering," for more information. 0 Disable data invariance mode. Outbound transactions use address invariance ordering. 1 Enable data invariance mode. Outbound transactions use data invariance ordering. **Note:** The data invariance feature is only supported for use with specific interface cards that require this non-default byte ordering policy.
2	—	Reserved
3	ROE	Relaxed ordering enable. This bit when set and the PCI Express device control register[Enable Relaxed] bit is set enables the Relaxed Ordering bit for the packet. This bit only applies to memory transactions. 0 Default ordering 1 Relaxed ordering
4	NS	No snoop enable. This bit when set and the PCI Express device control register[Enable No Snoop] bit is set enables the no snoop bit for the packet. This bit only applies to memory transactions. 0 Snoopable 1 No snoop
5–7	—	Reserved
8–10	TC	Traffic class. This field indicates the traffic class of the outbound packet. This field only applies to memory transaction. All other transaction types should set the TC field to 0. 000 TC0 001 TC1 010 TC2 011 TC3 100 TC4 101 TC5 110 TC6 111 TC7 **Note:** Traffic class settings are passed through to the PCI Express link, but no specific actions are taken in the device based on traffic class.
11	—	Reserved

Table 20-29. PEXVFOWAR*n* Field Descriptions (continued)

Bits	Name	Description
12–15	RTT	Read transaction type. Read transaction type to run on the PCI Express link 0100 Memory read All other settings reserved
16–19	WTT	Write transaction type. Write transaction type to run on the PCI Express link. 0100 Memory write All other settings reserved
20–25	—	Reserved
26–31	OWS	Outbound window size. Outbound translation window size N which is the encoded $2^{(N + 1)}$-byte window size. The smallest window size is 4 Kbytes. The window size represents the size of one VF in a PF. To determine the total VF size per PF, it is OWS*total num VF active in a PF. 000000 Reserved ... 001011 4-Kbyte window size 001100 8-Kbyte window size ... 011111 4-Gbyte window size 100000 8-Gbyte window size 100001 16-Gbyte window size 100010 32-Gbyte window size 100011 64-Gbyte window size ... 100111 1-Tbyte window size 101000 Reserved ... 111111 Reserved

20.3.5.1.5 PCI Express Outbound Translation Address Registers (PEXOTAR*n*)

The PCI Express outbound translation address registers, shown in Figure 20-29, select the starting addresses in the system address space for window hits within the PCI Express outbound address translation windows. The new translated address is created by concatenating the transaction offset to this translation address. PEXOTAR0 exists in RC mode or for PF0-only in EP mode. PEXOTAR1-4 are unique per PF.

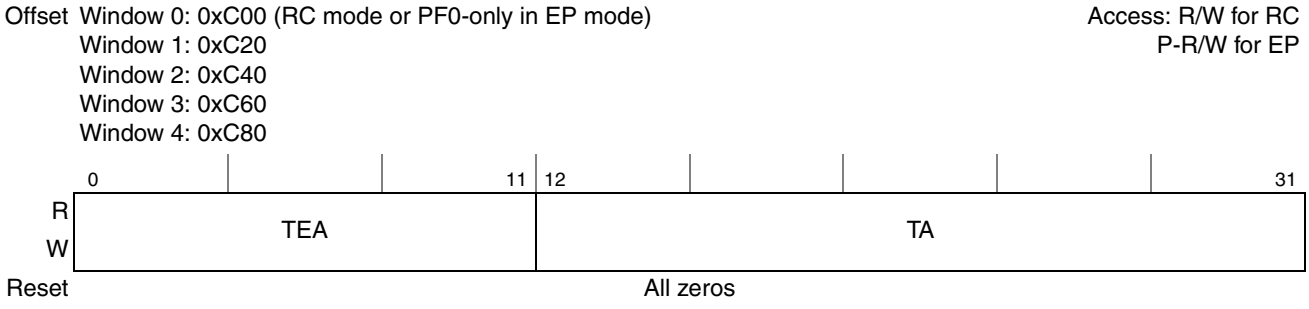


Figure 20-29. PCI Express Outbound Translation Address Registers (PEXOTAR*n*)

Table 20-30 describes the fields of the PCI Express outbound translation address registers.

Table 20-30. PEXOTAR*n* Field Descriptions

Bits	Name	Description
0–11	TEA	Translation extended address. System address which indicates the starting point of the outbound translated address. The translation address must be aligned based on the size field. Corresponds to PCI Express address bits [43:32] (bit 32 is the lsb).
12–31	TA	Translation address. System address which indicates the starting point of the outbound translated address. The translation address must be aligned based on the size field. This corresponds to PCI Express address bits [31:12].

20.3.5.1.6 PCI Express Outbound Translation Extended Address Registers (PEXOTEAR*n*)

The PCI Express outbound translation extended address registers, shown in Figure 20-30, contain the most-significant bits of a 64 bit translation address. PEXOTEAR0 exists in RC mode or for for PF0-only in EP mode. PEXOTEAR1-4 are unique per PF.

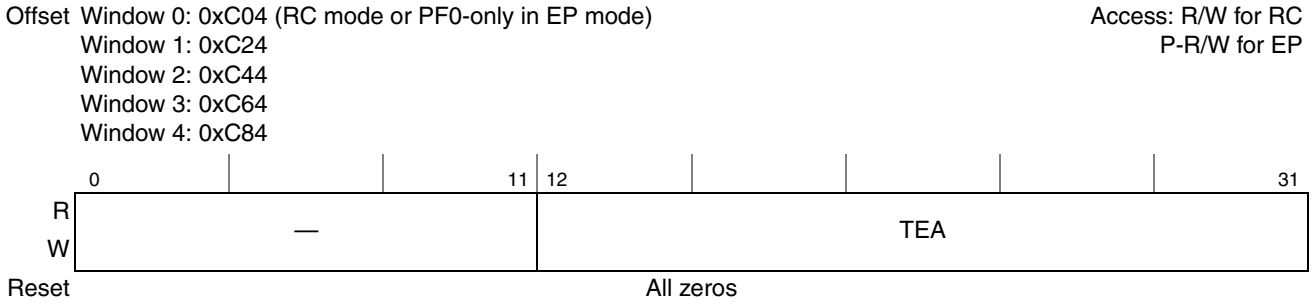


Figure 20-30. PCI Express Outbound Translation Extended Address Registers (PEXOTEAR*n*)

Table 20-31 describes the fields of the PCI Express outbound translation extended address registers.

Table 20-31. PCI Express Outbound Extended Address Translation Register *n* Field Descriptions

Bits	Name	Description
0–11	—	Reserved
12–31	TEA	Translation extended address. System address which indicates the starting point of the outbound translated address. The translation address must be aligned based on the size field. Corresponds to PCI Express address bits [63:44].

20.3.5.1.7 PCI Express Outbound Window Base Address Registers (PEXOWBAR*n*)

The PCI Express outbound window base address registers, shown in Figure 20-31, select the base address for the windows which are translated to the external address space. Addresses for outbound transactions are compared to these windows. If such a transaction does not fall within one of these spaces the transaction is forwarded through a default register set. These registers are unique per PF.

Offset Window 1: 0xC28
Window 2: 0xC48
Window 3: 0xC68
Window 4: 0xC88

Access: R/W for RC
P-R/W for EP

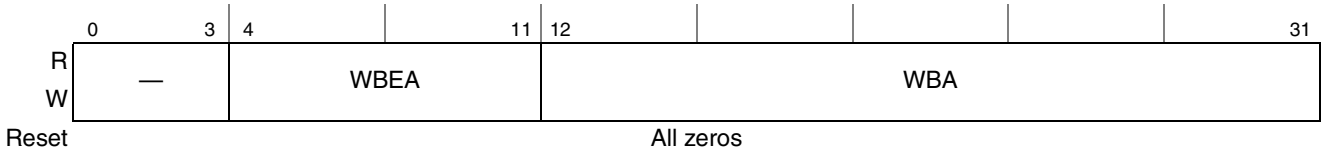


Figure 20-31. PCI Express Outbound Window Base Address Registers (PEXOWBAR*n*)

Table 20-32 describes the fields of the PCI Express outbound window base address registers.

Table 20-32. PCI Express Outbound Window Base Address Register *n* Field Descriptions

Bits	Name	Description
0–37	—	Reserved
4–11	WBEA	Window base extended address. Source address which is the starting point for the outbound translation window. The window must be aligned based on the size selected in the window size bits. Correspond to internal platform address bits [0:7]. (where 0 is the msb of the internal platform address)
12–31	WBA	Window base address. Source address which is the starting point for the outbound translation window. The window must be aligned based on the size selected in the window size bits. This corresponds to internal platform address bits [8:27].

20.3.5.1.8 PCI Express Outbound Window Attributes Registers (PEXOWAR*n*)

The PCI Express outbound window attributes registers, shown in Figure 20-32 and Figure 20-33, define the window sizes to translate and other attributes for the translations. 1 Tbyte is the largest window size allowed. Figure 20-32 shows the outbound window attributes register 0 (PEXOWAR0). PEXOWAR0 only exists for PF0. In order to properly update the fields of this register, there should not be back-to-back writes. At least 2 pipe clocks delay are needed in between write accesses when changing fields of this register.

Offset 0xC10

Access: Mixed for RC
P-Mixed for EP

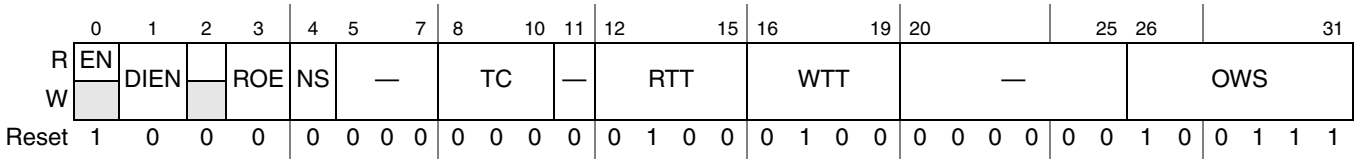


Figure 20-32. PCI Express Outbound Window Attributes Register 0 (PEXOWAR0)

Figure 20-33 shows the PCI Express outbound window attributes registers 1, 2, 4 (PEXOWAR*n*). These registers are unique per PF.

Offset Window 1: 0xC30
Window 2: 0xC50
Window 4: 0xC90

Access: R/W for RC
P-R/W for EP

	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	EN	DIEN		ROE	NS				TC				RTT				WTT				—						OWS					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	1	1	1

Figure 20-33. PCI Express Outbound Window Attributes Registers 1, 2, 4 (PEXOWAR*n*)

Figure 20-34 shows the PCI Express outbound window attributes registers 3 (PEXOWAR3). This register is unique per PF.

Offset Window 3: 0xC70

Access: R/W for RC
P-R/W for EP

	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	EN	DIEN		ROE	NS				TC				RTT				WTT				—						OWS					
Reset	All zeros																															

Figure 20-34. PCI Express Outbound Window Attributes Register 3 (PEXOWAR3)

Table 20-33 describes the fields of the PCI Express outbound window attributes registers.

Table 20-33. PEXOWAR*n* Field Descriptions

Bits	Name	Description
0	EN	Enable. This bit enables this address translation window. For the default window, this bit is read-only and always hardwired to 1. 0 Disable outbound translation window 1 Enable outbound translation window
1	DIEN	Data invariance enable. This bit controls the data ordering policy of all outbound transactions passing through this window. See Section 20.4.1.2, "Byte Ordering," for more information. 0 Disable data invariance mode. Outbound transactions use address invariance ordering. 1 Enable data invariance mode. Outbound transactions use data invariance ordering. **Note:** The data invariance feature is only supported for use with specific interface cards that require this non-default byte ordering policy.
2	—	Reserved
3	ROE	Relaxed ordering enable. This bit when set and the PCI Express device control register[Enable Relaxed] bit is set enables the Relaxed Ordering bit for the packet. This bit only applies to memory transactions. 0 Default ordering 1 Relaxed ordering
4	NS	No snoop enable. This bit when set and the PCI Express device control register[Enable No Snoop] bit is set enables the no snoop bit for the packet. This bit only applies to memory transactions. 0 Snoopable 1 No snoop
5–7	—	Reserved

Table 20-33. PEXOWAR*n* Field Descriptions (continued)

Bits	Name	Description
8–10	TC	Traffic class. This field indicates the traffic class of the outbound packet. This field only applies to memory transaction. All other transaction types should set the TC field to 0. 000 TC0 001 TC1 010 TC2 011 TC3 100 TC4 101 TC5 110 TC6 111 TC7 **Note:** Traffic class settings are passed through to the PCI Express link, but no specific actions are taken in the device based on traffic class.
11	—	Reserved
12–15	RTT	Read transaction type. Read transaction type to run on the PCI Express link 0000 Reserved 0000 Reserved 0010 Configuration read. Supported only when in RC mode and size of less than or equal to 4 bytes and not crossing 4-byte address boundary. 0100 Memory read ... Reserved 1000 IO read. Supported only when in RC mode and size of less than or equal to 4 bytes and not crossing 4-byte address boundary. ... Reserved 1111 Reserved
16–19	WTT	Write transaction type. Write transaction type to run on the PCI Express link. 0000 Reserved 0001 Reserved 0010 Configuration write. Supported only when in RC mode and size of less than or equal to 4 bytes and not crossing 4-byte address boundary. 0100 Memory write 0101 Message write. Only support 4-byte size access on a 4-byte address boundary. ... Reserved 1000 IO Write. Supported only when in RC mode and size of less than or equal to 4 bytes and not crossing 4-byte address boundary. ... Reserved 1111 Reserved

Table 20-33. PEXOWAR*n* Field Descriptions (continued)

Bits	Name	Description
20–25	—	Reserved
26–31	OWS	Outbound window size. Outbound translation window size N which is the encoded $2^{(N+1)}$-byte window size. The smallest window size is 4 Kbytes. Note that for the default window (window 0), the outbound window size may be programmed less than the 1-Tbyte maximum. However, accesses that miss all other windows and hit outside the default window is aliased to the default window. 000000 Reserved ... 001011 4-Kbyte window size 001100 8-Kbyte window size ... 011111 4-Gbyte window size 100000 8-Gbyte window size 100001 16-Gbyte window size 100010 32-Gbyte window size 100011 64-Gbyte window size ... 100111 1-Tbyte window size ... 111111 Reserved

20.3.5.2 PCI Express MSI-X Trap Outbound Window Base Address Register (PEXMSIX_TPOWBAR) - EP only

The PCI Express MSI-X trap outbound window base address register, shown in Figure 20-35, provides address offset for MSI-X trap operation. When MSI-X trap operation is enabled and a 4-byte write transaction falls into the MSI-X trap window starting address, hardware converts the write transaction to a MSI-X operation. The MSI-X vector information is included in the write data.

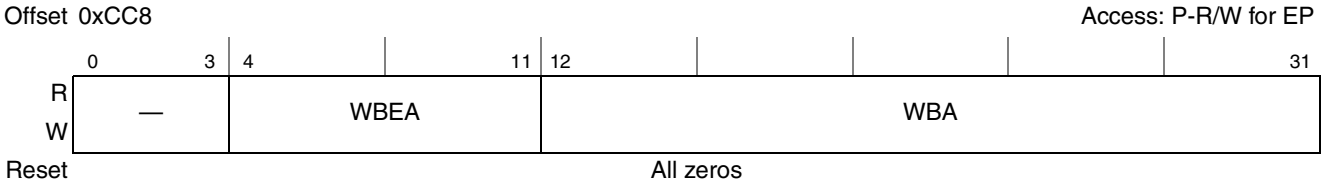


Figure 20-35. PCI Express MSI-X Trap Outbound Window Base Address Registers (PEXMSIX_TPOWBAR)

Table 20-34 describes the fields of the PCI Express MSI-X Trap outbound window base address registers.

Table 20-34. PEXMSIX_TPOWBAR Field Descriptions

Bits	Name	Description
0–3	—	Reserved
4–11	WBEA	Window base extended address. Source address which is the starting point for the outbound translation window. The window must be aligned based on the size selected in the window size bits. Correspond to internal platform address bits [0:7]. (where 0 is the msb of the internal platform address)
12–31	WBA	Window base address. Source address which is the starting point for the outbound translation window. The window must be aligned based on the size selected in the window size bits. This corresponds to internal platform address bits [8:27].

20.3.5.3 PCI Express MSI-X Trap Outbound Window Attribute Register (PEXMSIX_TPOWAR) - EP only

The PCI Express MSI-X trap outbound window attribute register, provides controls for MSI-X trap operation. These registers are unique per PF.

Offset 0xCD0 Access: P-R/W for EP

	0	1	2	3	4	5	7	8	10	11	12	15	16	19	20	25	26	31
R W	EN	DIEN	—										WTT		—		OWS	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	1	0	1	1

Figure 20-36. PCI Express MSI-X Trap Outbound Window Attributes Register (PEXMSIX_TPOWAR)

Table 20-35 describes the fields of the PCI Express outbound window attributes registers.

Table 20-35. PEXMSIX_TPOWAR Field Descriptions

Bits	Name	Description
0	EN	Enable. This bit enables this address translation window. For the default window, this bit is read-only and always hardwired to 1. 0 Disable outbound translation window 1 Enable outbound translation window
1	DIEN	Data invariance enable. This bit controls the data ordering policy of all outbound transactions passing through this window. See Section 20.4.1.2, "Byte Ordering," for more information. 0 Disable data invariance mode. Outbound transactions use address invariance ordering. 1 Enable data invariance mode. Outbound transactions use data invariance ordering. **Note:** The data invariance feature is only supported for use with specific interface cards that require this non-default byte ordering policy.
2–15	—	Reserved
16–19	WTT	Write transaction type. Write transaction type to run on the PCI Express link. 0000 Reserved 0001 Reserved 0010 Reserved 0100 Memory write 0101 Reserved ... Reserved 1000 Reserved ... Reserved 1111 Reserved
20–25	—	Reserved
26–31	OWS	Outbound window size. Outbound translation window size N which is the encoded $2^{(N + 1)}$-byte window size. The largest window size is 4 Kbytes. 000000 Reserved ... 001011 4-Kbyte window size 001100 Reserved ... 111111 Reserved

20.3.5.4 PCI Express Inbound ATMU Registers

There are differences between RC and EP implementations of inbound ATMU registers as described in the following sections.

20.3.5.4.1 EP Inbound ATMU Implementation

All base address registers (BARs) reside in the PCI Express type 0 configuration header space which is accessible through the PEX_CONFIG_ADDR/PEX_CONFIG_DATA mechanism. Note that host software must program these BAR using configuration type 0 cycles. There are 4 inbound BARs.

- Default inbound window BAR0 at configuration address 0x10 (32-bit). Also known as PEXCSRBAR. This is a fixed 16-Mbyte window used for inbound memory transactions that access memory-mapped registers. This window can also be used as either a fixed 16-Mbyte window used for inbound memory transactions that access memory-mapped registers or it can be programmed as a general purpose window for local memory access.
- Inbound window BAR1 at configuration address 0x14 (32-bit) This window is generic when not supporting MSI-X (that is, non SR-IOV device); this window is reserved for MSI-X table access for SR-IOV device.
- Inbound window BAR2 at configuration address 0x18-0x1c (64-bit)
- Inbound window BAR3 at configuration address 0x20-0x24 (64-bit)
- Inbound ROM BAR at configuration address 0x30

The PCI Express controller does not implement a shadow of the inbound BARs in the memory-mapped register set. However, when there is a hit to the BAR(s), the PCI Express controller uses the corresponding translation and attribute registers (PEXITARn and PEXIWARn) in the memory-mapped register set for the translation. If the transaction hits multiple BARs, then the lowest-numbered BAR is used.

For SR-IOV implementation in EP mode, there are 2 separate sets of inbound ATMU registers. One set is defined for PF translation and another is defined for VF translation.

All inbound ATMU registers that are dedicated to VF address decode occupy memory-mapped offset 0x1200-0x13FC. All inbound ATMU registers dedicated to PF address decode occupy memory-mapped offset 0xD00-DFC. Note that this condition only applies if in EP mode.

All PF BARs reside in the PCI Express type 0 config header space and all VF BARs reside in SR-IOV Capabilities structure which is accessible through the PEX_CONFIG_ADDR/PEX_CONFIG_DATA mechanism. There are 4 inbound VF BARs.

- Default inbound window VF BAR0 at configuration address SR-IOV Capabilities offset + 0x24 (32-bit). Also known as PEXCSRBAR. This window can also be used as either a fixed 1-Mbyte or 16-Mbyte window for inbound memory transactions that access memory-mapped registers or it can be programmed as a general purpose window for local memory access.
- Inbound window VF BAR1 at configuration address SR-IOV Capabilities offset + 0x28 (32-bit). Reserved for MSI-X table access.
- Inbound window VF BAR2 at configuration address SR-IOV Capabilities offset + 0x2c-0x30 (64-bit)

- Inbound window VF BAR3 at configuration address SR-IOV Capabilities offset + 0x34-0x38 (64-bit)

The PCI Express controller does not implement a shadow of the inbound VF BARs in the memory-mapped register set. However, when there is a hit to the VF BAR, the PCI Express controller uses the corresponding translation and attribute registers (PEXVFIWnTARm and PEXVFIWARn) in the memory-mapped register set for the translation. Similarly, when there is a hit to the PF BAR, the PCI Express controller uses the corresponding translation and attribute registers (PEXIWnTARm and PEXIWARn) in the memory-mapped register set for the translation.

Each VF BAR defines the size of a single memory aperture for the first VF. If there are multiple VFs, then the memory aperture for subsequent VFs lined up in subsequently memory addresses. The size of the memory aperture for each VF is defined in PEXVFIWAR[IWS]. The total BAR memory size is total of VF per PF (that is, 64) * PEXVFIWAR[IWS] size.

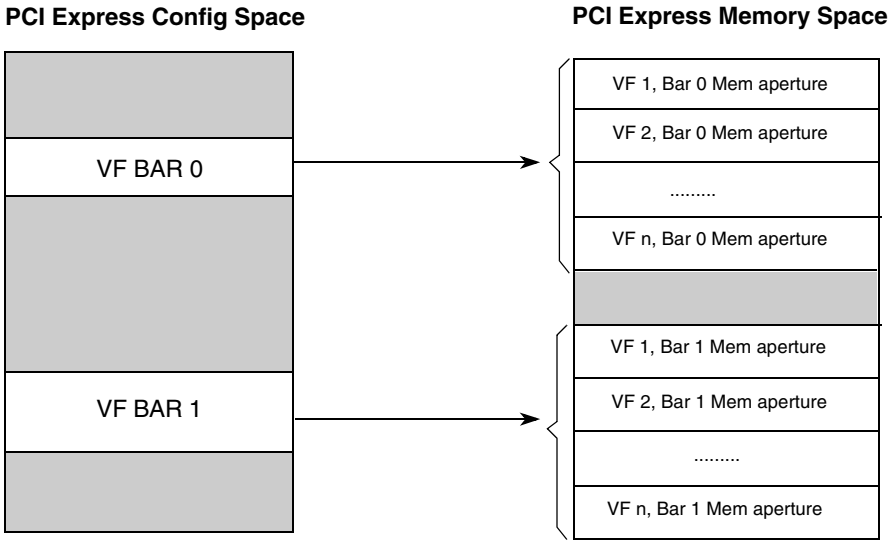


Figure 20-37. EP SR-IOV VF BARs Inbound Address - EP only

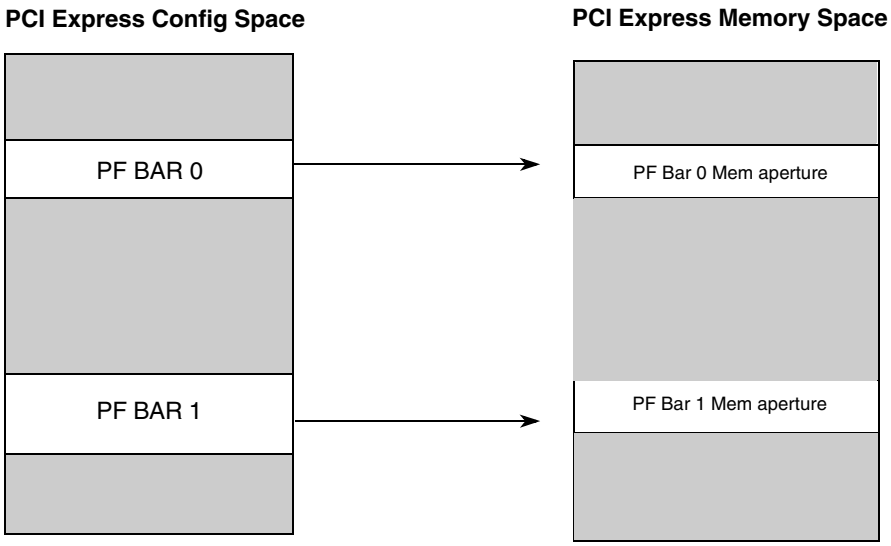


Figure 20-38. EP SR-IOV PF BARs Inbound Address

20.3.5.4.2 RC Inbound ATMU Implementation

In RC mode, the PEXIWBAR[1–3] registers reside outside of the type 1 header; PEXIWBAR0 resides in the Type 1 header (at offset 0x10). PEXIWBAR0 can be used as either a fixed 16-Mbyte window for inbound memory transactions that access memory-mapped registers or it can be programmed as a general purpose window for local memory access. In addition, in RC mode, there is a 64-bit MSI BAR that decodes 64-bit MSI writes and an inbound expansion ROM base address in the type 1 header at offset 0x38.

If the transaction hits any window, the translation is performed and then the transaction is sent to memory. If there is no hit to any one of the BARs, then an UR completion is returned for non-posted transactions. All posted transactions with no BAR hit are ignored.

Figure 20-39 shows the inbound transaction flow in RC mode.

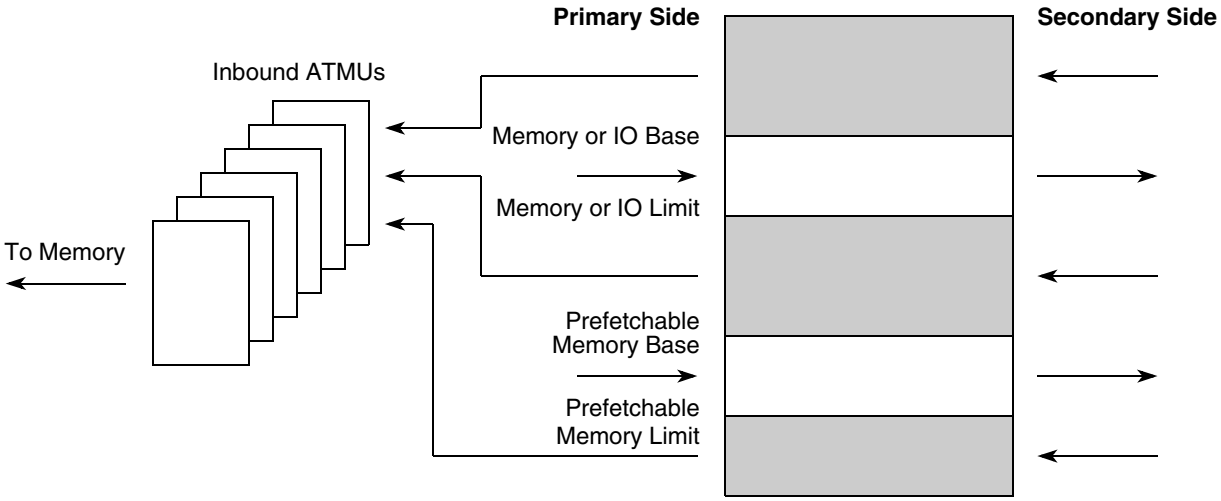


Figure 20-39. RC Inbound Transaction Flow

20.3.5.4.3 PCI Express VF Inbound Translation Address Registers (PEXVFIWTAR*n*)—EP Mode Only

The PCI Express VF inbound translation address registers, shown in Figure 20-40, contain the translated internal platform address to be used for transactions that are hitting into VF space. n indicates the Inbound Window number. The aperture of the size of the VF translated address region is the same as the size of the VF base address region. These registers are unique per PF.

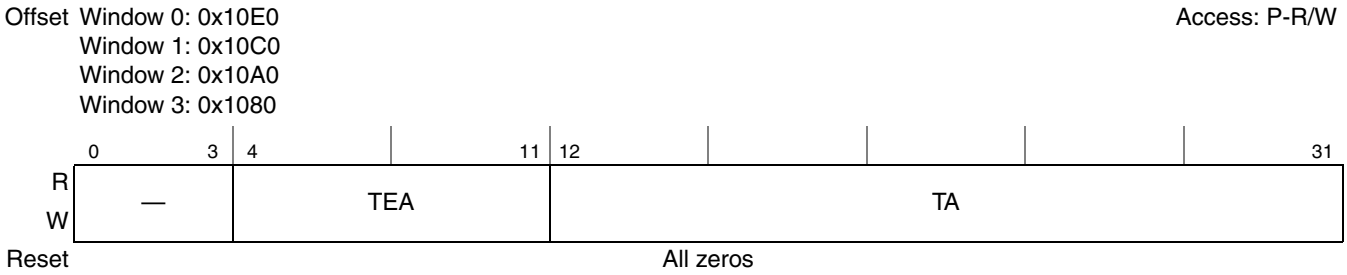


Figure 20-40. PCI Express VF Inbound Window Translation Address Registers (PEXVFIWTAR*n*)

Table 20-36 describes the fields of the PCI Express VF inbound translation address registers.

Table 20-36. PEXVFIWTAR*n* Field Descriptions

Bits	Name	Description
0–3	—	Reserved

Table 20-36. PEXVFIWTAR*n* Field Descriptions (continued)

Bits	Name	Description
4–11	TEA	Translation extended address. Target address which indicates the starting point of the inbound translated address. The translation address must be aligned based on the total VF size calculated from PEXVFIWARn[IWS] × the number of VFs. Corresponds to internal platform address bits [0:7] where bit 0 is the msb of the internal platform address. When PEXVFIWARn[TRGT] field is set for CCSRBAR, this field becomes CCSRBAR address. Otherwise, it takes on current programmed value.
12–31	TA	Translation address. Target address which indicates the starting point of the inbound translated address. The translation address must be aligned based on the total VF size calculated from the size field. This corresponds to internal platform address bits [8:27]. When PEXVFIWnARw[TRGT] field is set for CCSRBAR, this field becomes CCSRBAR address. Otherwise, it takes on current programmed value.

20.3.5.4.4 PCI Express VF Inbound Window Attributes Registers (PEXVFIWAR*n*)—EP Mode Only

The PCI Express VF inbound window attributes registers, shown in Figure 20-41, define the window size of a single VF to translate and other attributes for the translations. The total window size is determine by the number of VF supported × IWS. Total window size must not exceed 1 Tbyte. These registers are unique per PF.

Offset Window 0: 0x10F0
Window 1: 0x10D0
Window 2: 0x10B0
Window 3: 0x1090

Access: P-R/W

	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	EN		DIEN		PF		—				TRGT				RTT				WTT				—						IWS			
Reset W1–3	0		0		1		0		0	0	0	0	1	1	1	1	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1
W0	1		0		0		0		0	0	0	0	n	n	n	n	0	1	0	0	0	1	0	0	0	0	0	0	n	n	n	n	n	n

Figure 20-41. PCI Express VF Inbound Window Attributes Registers (PEXVFIWAR*n*)

Table 20-37 describes the fields of the PCI Express VF inbound window attributes registers.

Table 20-37. PEXVFIWAR*n* Field Descriptions

Bits	Name	Description
0	EN	Enable. This bit controls the enabling/disabling of the translation window. This field defaults to 1 for window 0. 0 Disable inbound window translation 1 Enable inbound window translation
1	DIEN	Data invariance enable. This bit controls the data ordering policy of all inbound transactions passing through this window. See Section 20.4.1.2, "Byte Ordering," for more information. 0 Disable data invariance mode. Inbound transactions use address invariance ordering. 1 Enable data invariance mode. Inbound transactions use data invariance ordering. **Note:** The data invariance feature is only supported for use with specific interface cards that require this non-default byte ordering policy.

Table 20-37. PEXVFIWAR*n* Field Descriptions (continued)

Bits	Name	Description
2	PF	Prefetchable. This bit indicates that the address space is prefetchable. This bit corresponds to the prefetchable bit in the BAR in the PCI Express type 0 header. This bit drives the BAR's prefetchable bit in EP mode. When PEXVFIWARn[TRGT] field is set for CCSRBAR, this field is set to 0. Otherwise, it takes on current programmed value. 0 Not prefetchable 1 Prefetchable
3–7	—	Reserved
8–11	TRGT	Target interface. This field defaults to CCSRBAR for window 0. Target interface. If this field is set to anything other than local memory space, the attributes for the transaction must be assigned in a corresponding outbound window at the target. 0000 PCI Express 1—PCI Express controller 1 should not use this encoding 0001 PCI Express 2—PCI Express controller 2 should not use this encoding 0010 PCI Express 3—PCI Express controller 3 should not use this encoding 0011–0111 Reserved 1000 Serial RapidIO 1 1001 Serial RapidIO 2 1010–1110 Reserved 1111 Local memory space xxxx Route to CCSRBAR. **Note:** PEXVFIWAR*n*[TRGT] must not be set to target the initiating interface itself. For example, PEXVFIWAR*n* for PCI Express controller 1 should never target PCI Express 1. **Note:** If this field is set to anything other than local memory space, the attributes for the transaction must be assigned in a corresponding outbound window at the target. **Note:** Inbound write transactions on one PCI Express port must not translate to outbound configuration or I/O write transactions on another PCI Express port.
12–15	RTT	Read transaction type. Read transaction type to send to target interface. When PEXVFIWARn[TRGT] field is set for CCSRBAR, this field is set to 4. Otherwise, it takes on current programmed value. If the transaction is not going to local memory space 0000 Reserved ... 0100 Read 0101 Reserved 0110 Reserved 0111 Reserved 1000 Reserved ... 1111 Reserved If the transaction is going to local memory space 0000 Reserved ... 0100 Read, do not snoop local processor 0101 Read, snoop local processor 0110 Reserved 0111 Read, snoop local processor, unlock L2 cache line 1000 Reserved ... 1111 Reserved

Table 20-37. PEXVFIWAR*n* Field Descriptions (continued)

Bits	Name	Description
16–19	WTT	Write transaction type. Write transaction type to send to target interface. When PEXVFIWARn[TRGT] field is set for CCSRBAR, this field is set to 4. Otherwise, it takes on current programmed value. If the transaction is not going to local memory space 0000 Reserved ... 0100 Write 0101 Reserved 0110 Reserved 0111 Reserved 1000 Reserved ... 1111 Reserved If the transaction is going to local memory space 0000 Reserved ... 0100 Write, do not snoop local processor 0101 Write, snoop local processor 0110 Write, snoop local processor, allocate L2 cache line 0111 Write, snoop local processor, allocate and lock L2 cache line 1000 Reserved ... 1111 Reserved
20–25	—	Reserved
26–31	IWS	Inbound window size. Inbound translation window size N which is the encoded $2^{(N + 1)}$-bytes window size for a single VF. The total window size is determined by number of VF * IWS. The smallest window size is 4 Kbytes. For EP mode, this field directly controls the size of the BARs. When PEXVFIWARn[TRGT] field is set for CCSRBAR, this field can be either 1M or 16M. Otherwise, it takes on current programmed value. Care must be taken such that VF×IWS does not exceed the maximum memory size (1Tbyte) supported and is not in conflict with any other memory region. For PEXVFIWAR1 in EP mode with SR-IOV, this field should be set to 8-Kbyte window size. 000000 Reserved ... 001010 Reserved 001011 4-Kbyte window size 001100 8-Kbyte window size ... 011111 4-Gbyte window size 100000 8-Gbyte window size 100001 16-Gbyte window size 100010 See requirement above ... 111111 Reserved

20.3.5.4.5 PCI Express Inbound Translation Address Registers (PEXITAR*n*)

The PCI Express inbound translation address registers, shown in Figure 20-42, contain the translated internal platform address to be used. These registers are unique per PF.

Offset Window 0: 0xDE0
Window 1: 0xDC0
Window 2: 0xDA0
Window 3: 0xD80

Access: R/W for RC
P-R/W for EP

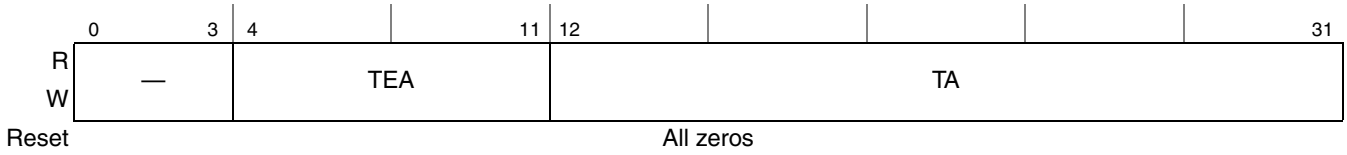


Figure 20-42. PCI Express Inbound Translation Address Registers (PEXITAR*n*)

Table 20-38 describes the fields of the PCI Express inbound translation address registers.

Table 20-38. PCI Express Inbound Translation Address Registers Field Descriptions

Bits	Name	Description
0–3	—	Reserved
4–11	TEA	Translation extended address. Target address which indicates the starting point of the inbound translated address. The translation address must be aligned based on the size field. Corresponds to internal platform address bits [0:7] where bit 0 is the msb of the internal platform address. When PEXIWARn[TRGT] field is set for CCSRBAR, this field becomes CCSRBAR address. Otherwise, it takes on the current programmed value.
12–31	TA	Translation address. Target address which indicates the starting point of the inbound translated address. The translation address must be aligned based on the size field. Corresponds to internal platform address bits [8:27] where bit 0 is the msb of the internal platform address. When PEXIWARn[TRGT] field is set for CCSRBAR, this field becomes CCSRBAR address. Otherwise, it takes on the current programmed value.

20.3.5.4.6 PCI Express Inbound Window Base Address Registers (PEXIWBAR*n*)—RC Mode Only

The PCI Express inbound window base address registers, shown in Figure 20-43, select the base address for the windows which are translated to an alternate target address space. In root complex (RC) mode, addresses for inbound transactions are compared to these windows. In RC mode, PEXIWBAR0 is located in the PCI Express type 1 configuration header space and PEXIWBAR[1–3] registers are implemented as described in this section. In endpoint (EP) mode, these registers are not implemented in the memory-mapped space. Reading these registers in EP mode returns all zeros and writing to these offsets has no consequences. All base address registers in EP are located in the PCI Express type 0 configuration header space. Note that PEXIWBAR1 only supports 32-bit PCI Express address space. These registers are unique per PF.

Offset Window 1: 0xDC8
Window 2: 0xDA8
Window 3: 0xD88

Access: R/W for RC
P-R/W for EP

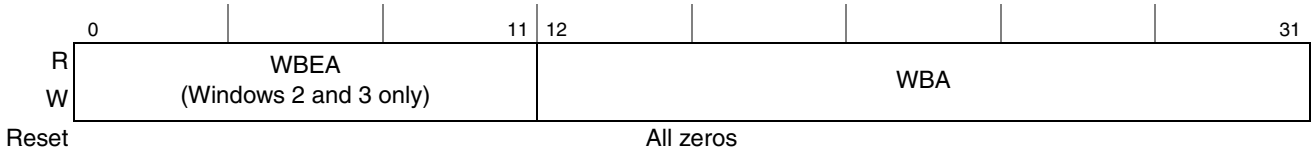


Figure 20-43. PCI Express Inbound Window Base Address Registers (PEXIWBAR*n*)

Table 20-39 describes the fields of the PCI Express inbound window base address registers.

Table 20-39. PCI Express Inbound Window Base Address Register Field Descriptions

Bits	Name	Description
0–11	WBEA	Window base extended address. This field corresponds to PCI Express address bits [43:32]. Note that the extended address is supported for windows 2 and 3 only; for PEXIWBAR1, these bits are reserved and must be 0.
12–31	WBA	Window base address. Source address which is the starting point for the inbound translation window. The window must be aligned based on the size selected in the window size bits. This corresponds to PCI Express address bits [31:12].

20.3.5.4.7 PCI Express Inbound Window Base Extended Address Registers (PEXIWBEAR*n*)—RC Mode Only

The PCI Express inbound window base extended address registers, shown in Figure 20-44, contain the most-significant bits of a 64 bit base address. This register is only available in RC mode. These registers are unique per PF.

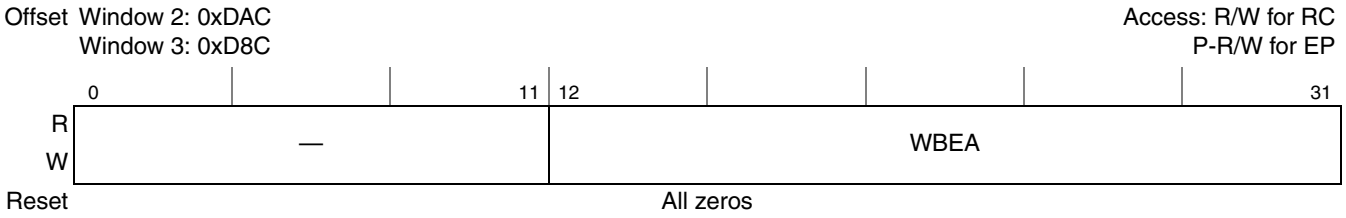


Figure 20-44. PCI Express Inbound Window Base Extended Address Registers (PEXIWBEAR*n*)

Table 20-40 describes the fields of the PCI Express inbound window base extended address registers.

Table 20-40. PCI Express Inbound Window Base Extended Address Register Field Descriptions

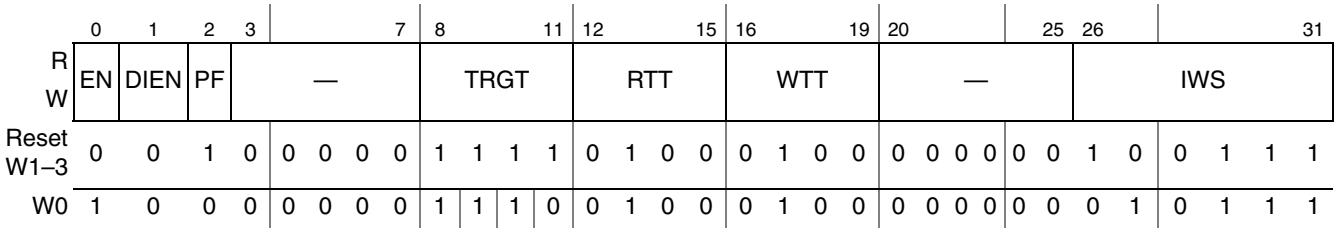
Bits	Name	Description
0–11	—	Reserved
12–31	WBEA	Window base extended address. This field corresponds to PCI Express address bits [63:44]

20.3.5.4.8 PCI Express Inbound Window Attributes Registers (PEXIWAR*n*)

The PCI Express inbound window attributes registers, shown in Figure 20-45, define the window sizes to translate and other attributes for the translations. 1 Tbyte is the largest window size allowed. These registers are unique per PF.

Offset Window 0: 0xDF0
Window 1: 0xDD0
Window 2: 0xDB0
Window 3: 0xD90

Access: R/W for RC
P-R/W for EP



	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	EN		DIEN		PF		—		TRGT				RTT				WTT				—						IWS					
Reset W1–3	0		0		1		0		0	0	0	0	1	1	1	1	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0	1	1	1
W0	1		0		0		0		0	0	0	0	1	1	1	0	0	1	0	0	0	1	0	0	0	0	0	0	0	1	0	1	1	1

Figure 20-45. PCI Express Inbound Window Attributes Registers (PEXIWAR*n*)

Table 20-41 describes the fields of the PCI Express inbound window attributes registers.

Table 20-41. PCI Express Inbound Window Attributes Registers Field Descriptions

Bits	Name	Description
0	EN	Enable. This bit controls the enabling/disabling of the translation window. This field defaults to 1 for window 0. 0 Disable inbound window translation 1 Enable inbound window translation
1	DIEN	Data invariance enable. This bit controls the data ordering policy of all inbound transactions passing through this window. See Section 20.4.1.2, "Byte Ordering," for more information. 0 Disable data invariance mode. Inbound transactions use address invariance ordering. 1 Enable data invariance mode. Inbound transactions use data invariance ordering. **Note:** The data invariance feature is only supported for use with specific interface cards that require this non-default byte ordering policy.
2	PF	Prefetchable. This bit indicates that the address space is prefetchable. This bit corresponds to the prefetchable bit in the BAR in the PCI Express type 0 header. This bit drives the BAR's prefetchable bit in EP mode.When PEXIWARn[TRGT] field is set for CCSRBAR, this field is set to 0. Otherwise, it takes on current programmed value. 0 Not prefetchable 1 Prefetchable
3–7	—	Reserved
8–11	TRGT	Target interface. This field defaults to CCSRBAR for window 0. Target interface. If this field is set to anything other than local memory space, the attributes for the transaction must be assigned in a corresponding outbound window at the target. 0000 PCI Express 1—PCI Express controller 1 should not use this encoding 0001 PCI Express 2—PCI Express controller 2 should not use this encoding 0010 PCI Express 3—PCI Express controller 3 should not use this encoding 0011–0111 Reserved 1000 Serial RapidIO 1 1001 Serial RapidIO 2 1110 Route to CCSRBAR. 1010–1101 Reserved 1111 Local memory space **Note:** PEXIWAR*n*[TRGT] must not be set to target the initiating interface itself. For example, PEXIWAR*n* for PCI Express controller 1 should never target PCI Express 1. **Note:** If this field is set to anything other than local memory space, the attributes for the transaction must be assigned in a corresponding outbound window at the target. **Note:** Inbound write transactions on one PCI Express port must not translate to outbound configuration or I/O write transactions on another PCI Express port.

Table 20-41. PCI Express Inbound Window Attributes Registers Field Descriptions (continued)

Bits	Name	Description
12–15	RTT	Read transaction type. Read transaction type to send to target interface. When PEXIWARn[TRGT] field is set for CCSRBAR, this field is set to 4. Otherwise, it takes on current programmed value. If the transaction is not going to local memory space 0000 Reserved ... 0100 Read 0101 Reserved 0110 Reserved 0111 Reserved 1000 Reserved ... 1111 Reserved If the transaction is going to local memory space 0000 Reserved ... 0100 Read, do not snoop local processor 0101 Read, snoop local processor 0110 Reserved 0111 Read, snoop local processor, unlock L2 cache line 1000 Reserved ... 1111 Reserved
16–19	WTT	Write transaction type. Write transaction type to send to target interface. When PEXIWARn[TRGT] field is set for CCSRBAR, this field is set to 4. Otherwise, it takes on current programmed value. If the transaction is not going to local memory space 0000 Reserved ... 0100 Write 0101 Reserved 0110 Reserved 0111 Reserved 1000 Reserved ... 1111 Reserved If the transaction is going to local memory space 0000 Reserved ... 0100 Write, do not snoop local processor 0101 Write, snoop local processor 0110 Write, snoop local processor, allocate L2 cache line 0111 Write, snoop local processor, allocate and lock L2 cache line 1000 Reserved ... 1111 Reserved

Table 20-41. PCI Express Inbound Window Attributes Registers Field Descriptions (continued)

Bits	Name	Description
20–25	—	Reserved
26–31	IWS	Inbound window size. Inbound translation window size N which is the encoded $2^{(N+1)}$-bytes window size. The smallest window size is 4 Kbytes. For EP mode, this field directly controls the size of the BARs. When PEXIWARn[TRGT] field is set for CCSRBAR, this field can be either 1M or 16M. Otherwise, it takes on current programmed value. For PEXIWAR1 in EP mode with SR-IOV, this field should be set to 8-Kbyte window size. 000000 Reserved ... 001010 Reserved 001011 4-Kbyte window size 001100 8-Kbyte window size ... 011111 4-Gbyte window size 100000 8-Gbyte window size 100001 16-Gbyte window size 100010 32-Gbyte window size 100011 64-Gbyte window size ... 100111 1-Tbyte window size ... 111111 Reserved

20.3.6 PCI Express Expansion ROM Inbound ATMU Registers

20.3.6.1 PCI Express Expansion ROM Inbound Translation Address Register (PEXEPROMITAR)

The PCI Express Expansion ROM inbound translation address register, shown in Figure 20-46, contains the translated internal platform (OCeaN) address to be used for Expansion ROM. The base of the Expansion ROM window is defined in the PCIe core. The expansion ROM translation address should be initialized to an appropriate value by the pre-boot loader.

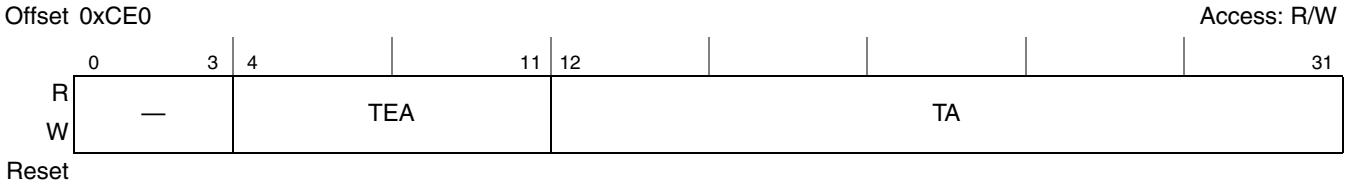


Figure 20-46. PCI Express Expansion ROM Inbound Translation Address Register (PEXEPROMITAR)

Table 20-42 describes the fields of the PCI Express Expansion ROM inbound translation address register.

Table 20-42. PEXEPROMITAR Field Descriptions

Bits	Name	Description
0–3	—	Reserved
4–11	TEA	Translation extended address. Target address which indicates the starting point of the inbound translated address. The translation address must be aligned based on the size field. Corresponds to internal platform address bits [0:7] where bit 0 is the msb of the internal platform address.
12–31	TA	Translation address. Target address which indicates the starting point of the inbound translated address.

20.3.6.2 PCI Express Expansion ROM Inbound Window Attributes Registers (PEXEPROMIWAR)

The PCI Express Expansion ROM inbound window attributes register, shown in Figure 20-47, defines the window attributes for the translations. he enable (EN) and inbound window size (IWS) attributes of the window are shadow copies of what is programmed in the Expansion ROM BAR Mask register in the SPEX core. The Expansion ROM window is limited to the lower 4-Gbyte space with maximim size of 16-Mbytes. The default window size is 16 Mbytes. It is expected that the pre-boot loader will program the PEXEPROMIWAR[EN], and PEXEPROMIWAR[IWS] fields and the Expansion ROM Bar Mask Register (at config offset 0x30) to have matching values prior to using the window.

Offset 0xCF0 Access: R/W

	0	1	2	3	7	8	11	12	15	16	19	20	25	26	31
R W	EN	DIEN	PF	—		TRGT		RTT		WTT		—		IWS	
Reset	0	0	0	0	0 0 0 0	1	1 1 1	0	1 0 0	0	1 0 0	0 0 0 0	0 0	0 1	0 1 1 1

Figure 20-47. PCI Express Expansion ROM Inbound Window Attributes Register (PEXEPROMIWAR)

Table 20-43 describes the fields of the PCI Express Expansion ROM inbound window attributes register.

Table 20-43. PEXEPROMIWAR Field Descriptions

Bits	Name	Description
0	EN	Enable. This bit controls the enabling/disabling of the Expansion ROM translation window. Note that bit 0 of the Expansion ROM Base Address register in the PCI Express PCI-Compatible Configuration Header needs to be programmed with the same value as this enable bit. 0 Disable inbound window translation 1 Enable inbound window translation
1	DIEN	Data invariance enable. This bit controls the data ordering policy of all inbound transactions passing through this window. See Section 20.4.1.2, “Byte Ordering,” for more information. 0 Disable data invariance mode. Inbound transactions use address invariance ordering. 1 Enable data invariance mode. Inbound transactions use data invariance ordering.
2	PF	Prefetchable. This bit indicates that the address space is prefetchable. This bit corresponds to the prefetchable bit in the BAR in the PCI Express type 0 header. This bit drives the BAR’s prefetchable bit in EP mode. 0 Not prefetchable 1 Prefetchable
3–7	—	Reserved

Table 20-43. PEXEPROMIWAR Field Descriptions (continued)

Bits	Name	Description
8–11	TRGT	Target interface. 0000PCI Express 1—PCI Express controller 1 should not use this encoding 0001 PCI Express 2—PCI Express controller 2 should not use this encoding 0010 PCI Express 3—PCI Express controller 3 should not use this encoding 0011–0111 Reserved 1000 Serial RapidIO 1 1001 Serial RapidIO 2 1010–1110 Reserved 1111 Local memory space xxxx Route to CCSRBAR. **Note:** PEXMSIIWAR*n*[TRGT] must not be set to target the PCI Express interface itself. That is, PEXMSIIWAR*n* for PCI Express controller 1 should never target PCI Express 1. **Note:** If this field is set to anything other than local memory space, the attributes for the transaction must be assigned in a corresponding outbound window at the target. **Note:** Inbound write transactions on one PCI Express port must not translate to outbound configuration or I/O write transactions on another PCI Express port.
12–15	RTT	Read transaction type. Read transaction type to send to target interface. If the transaction is not going to local memory space 0000 Reserved ... 0100 Read 0101 Reserved 0110 Reserved 0111 Reserved 1000 Reserved ... 1111 Reserved If the transaction is going to local memory space 0000 Reserved ... 0100 Read, do not snoop local processor 0101 Read, snoop local processor 0110 Reserved 0111 Read, snoop local processor, unlock L2 cache line 1000 Reserved ... 1111 Reserved

Table 20-43. PEXEPROMIWAR Field Descriptions (continued)

Bits	Name	Description
16–19	WTT	Write transaction type. Write transaction type to send to target interface. If the transaction is not going to local memory space 0000 Reserved ... 0100 Write 0101 Reserved 0110 Reserved 0111 Reserved 1000 Reserved ... 1111 Reserved If the transaction is going to local memory space 0000 Reserved ... 0100 Write, do not snoop local processor 0101 Write, snoop local processor 0110 Write, snoop local processor, allocate L2 cache line 0111 Write, snoop local processor, allocate and lock L2 cache line 1000 Reserved ... 1111 Reserved
20–25	—	Reserved
26–31	IWS	Inbound window size. Inbound translation window size N which is the encoded $2^{(N + 1)}$-bytes window size. The smallest window size is 4 Kbytes. For EP mode, this field directly controls the size of the BARs. Note that IWS needs to match the Expansion ROM BAR mask register at configuration offset 0x30 and PEX_CSR1[CS2] needs to be set when programming the Expansion ROM BAR mask register. 000000 Reserved ... 001010 Reserved 001011 4-Kbyte window size 001100 8-Kbyte window size ... 010111 16-Mbyte window size 111000 Reserved ... 111111 Reserved

20.3.6.2.1 PCI Express MSI Inbound Translation Address Register (PEXMSIITAR)—RC Mode Only

The PCI Express MSI inbound translation address register, shown in Figure 20-48, contains the translated internal platform address to be used. The translated address is always mapped to CCSRBAR space similarly to PEXITAR0.

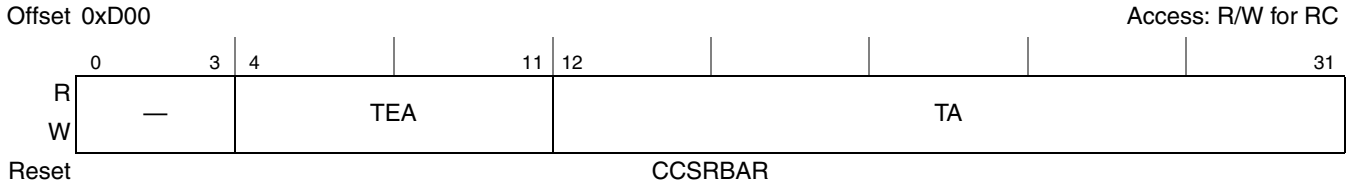


Figure 20-48. PCI Express MSI Inbound Translation Address Register (PEXMSIITAR)

Table 20-44 describes the fields of the PCI Express MSI inbound translation address register.

Table 20-44. PCI Express MSI Inbound Translation Address Register Field Descriptions

Bits	Name	Description
0–3	—	Reserved
4–11	TEA	Translation extended address. Target address which indicates the starting point of the inbound translated address. The translation address must be aligned based on the size field. Corresponds to internal platform address bits [0:7] where bit 0 is the msb of the internal platform address. Mapped to CCSRBAR. .
12–31	TA	Translation address. Target address which indicates the starting point of the inbound translated address. Mapped to CCSRBAR.

20.3.6.2.2 PCI Express MSI Inbound Window Base Address Register (PEXMSIIWBAR)—RC Mode Only

The PCI Express MSI inbound window base address register, shown in Figure 20-49, selects the base address for the windows which are translated to an alternate target address space. In root complex (RC) mode, addresses for the 64-bit MSI inbound transactions are compared to this window.

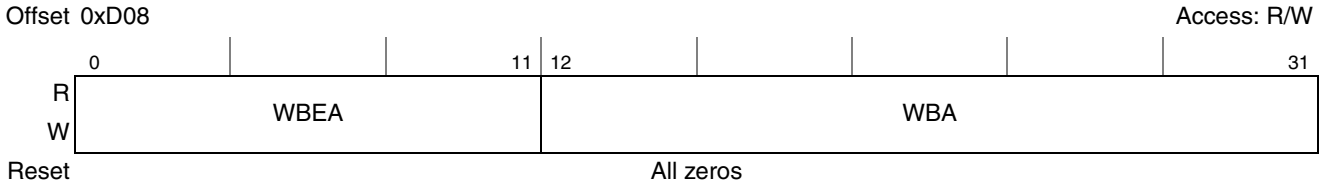


Figure 20-49. PCI Express MSI Inbound Window Base Address Register (PEXMSIIWBAR)

Table 20-45 describes the fields of the PCI Express MSI inbound window base address registers.

Table 20-45. PCI Express MSI Inbound Window Base Address Register Field Descriptions

Bits	Name	Description
0–11	WBEA	Window base extended address. This field corresponds to PCI Express address bits [43:32].
12–31	WBA	Window base address. Source address which is the starting point for the inbound translation window. The window must be aligned based on the size selected in the window size bits. This corresponds to PCI Express address bits [31:12].

20.3.6.2.3 PCI Express MSI Inbound Window Base Extended Address Registers (PEXMSIIWBEAR)—RC Mode Only

The PCI Express MSI inbound window base extended address register, shown in Figure 20-50, contains the most-significant bits of a 64-bit MSI base address.

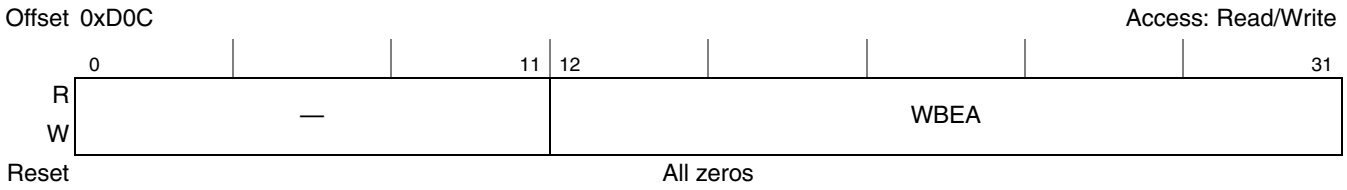


Figure 20-50. PCI Express MSI Inbound Window Base Extended Address Register (PEXMSIIWBEAR)

Table 20-46 describes the fields of the PCI Express MSI inbound window base extended address registers.

Table 20-46. PCI Express MSI Inbound Window Base Extended Address Register Field Descriptions

Bits	Name	Description
0–11	—	Reserved
12–31	WBEA	Window base extended address. This field corresponds to PCI Express address bits [63:44]

20.3.6.2.4 PCI Express MSI Inbound Window Attributes Registers (PEXMSIIWAR) —RC Mode Only

The PCI Express MSI inbound window attributes register, shown in Figure 20-51, defines the window sizes to translate and other attributes for the translations. The size is fixed at 16-M window.

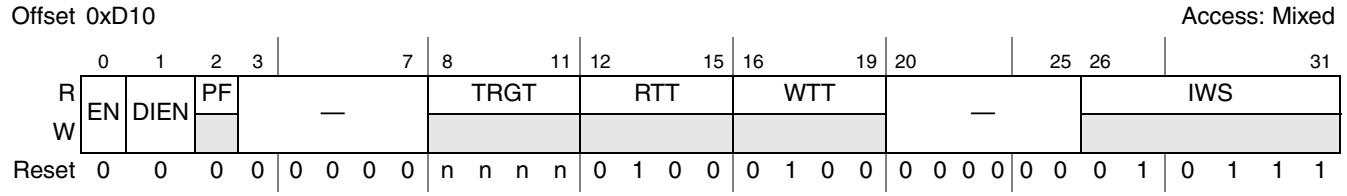


Figure 20-51. PCI Express MSI Inbound Window Attributes Register (PEXMSIIWAR)

Table 20-47 describes the fields of the PCI Express MSI inbound window attributes register.

Table 20-47. PCI Express MSI Inbound Window Attribute Register Field Descriptions

Bits	Name	Description
0	EN	Enable. This bit controls the enabling/disabling of the MSI translation window. 0 Disable inbound window translation 1 Enable inbound window translation
1	DIEN	Data invariance enable. This bit controls the data ordering policy of all inbound transactions passing through this window. See Section 20.4.1.2, "Byte Ordering," for more information. 0 Disable data invariance mode. Inbound transactions use address invariance ordering. 1 Enable data invariance mode. Inbound transactions use data invariance ordering.

Table 20-47. PCI Express MSI Inbound Window Attribute Register Field Descriptions (continued)

Bits	Name	Description
2	PF	Prefetchable. This bit indicates that the address space is prefetchable. This bit corresponds to the prefetchable bit in the BAR in the PCI Express type 0 header. This bit drives the BAR's prefetchable bit in EP mode. 0 Not prefetchable 1 Prefetchable
3–7	—	Reserved
8–11	TRGT	Target interface. 0000 PCI Express 1—PCI Express controller 1 should not use this encoding 0001 PCI Express 2—PCI Express controller 2 should not use this encoding 0010 PCI Express 3—PCI Express controller 3 should not use this encoding 0011–0111 Reserved 1000 Serial RapidIO 1 1001 Serial RapidIO 2 1010–1110 Reserved 1111 Local memory space xxxx Route to CCSRBAR. **Note:** PEXMSIIWAR*n*[TRGT] must not be set to target the PCI Express interface itself. That is, PEXMSIIWAR*n* for PCI Express controller 1 should never target PCI Express 1. **Note:** If this field is set to anything other than local memory space, the attributes for the transaction must be assigned in a corresponding outbound window at the target. **Note:** Inbound write transactions on one PCI Express port must not translate to outbound configuration or I/O write transactions on another PCI Express port.
12–15	RTT	Read transaction type. Read transaction type to send to target interface. If the transaction is not going to local memory space 0000 Reserved ... 0100 Read 0101 Reserved 0110 Reserved 0111 Reserved 1000 Reserved ... 1111 Reserved If the transaction is going to local memory space 0000 Reserved ... 0100 Read, do not snoop local processor 0101 Read, snoop local processor 0110 Reserved 0111 Read, snoop local processor, unlock L2 cache line 1000 Reserved ... 1111 Reserved

Table 20-47. PCI Express MSI Inbound Window Attribute Register Field Descriptions (continued)

Bits	Name	Description
16–19	WTT	Write transaction type. Write transaction type to send to target interface. If the transaction is not going to local memory space 0000 Reserved ... 0100 Write 0101 Reserved 0110 Reserved 0111 Reserved 1000 Reserved ... 1111 Reserved If the transaction is going to local memory space 0000 Reserved ... 0100 Write, do not snoop local processor 0101 Write, snoop local processor 0110 Write, snoop local processor, allocate L2 cache line 0111 Write, snoop local processor, allocate and lock L2 cache line 1000 Reserved ... 1111 Reserved
20–25	—	Reserved
26–31	IWS	Inbound window size. Inbound translation window size N which is the encoded $2^{(N + 1)}$-bytes window size. 010111 16-Mbyte window size

20.3.7 PCI Express Error Management Registers

20.3.7.1 PCI Express Error Detect Register (PEX_ERR_DR)

The PCI Express error detect register, shown in Figure 20-52, contains error status bits that are detected by hardware.The detected error bits are write-1-to-clear type registers. Reading from these registers occurs normally; however, write operations can clear but not set bits. A bit is cleared whenever the register is written, and the data in the corresponding bit location is a 1. For example, to clear bit 6 and not affect any other bits in the register, the value 0b0200_0000 is written to the register. When an error is detected the appropriate error bit is set. Subsequent errors sets the appropriate error bits in the error detection registers, but only the first error for a particular unit have any relevant information captured. The interrupt enable bits are used to allow or block the error reporting to the interrupt mechanism while the disable bits are used to prevent or allow the setting of the status bits.

Offset 0xE00 Access: w1c

	0	1–7	8	9	10	11	12	13	14	15
R	ME	—	PCT	PAT	PCAC	PNM	CDNSC	CRSNC	ICCA	IACA
W	w1c		w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c
Reset	All zeros									

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	CRST	MIS	IOIS	CIS	CIEP	IOIEP	OAC	IOIA	IMBA	IIOBA	LDDE	LME	FLRDE	UREP	TAE	—
W	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	w1c	
Reset	All zeros															

Figure 20-52. PCI Express Error Detect Register (PEX_ERR_DR)

Table 20-48 describes the fields of the PCI Express error detect register.

Table 20-48. PCI Express Error Detect Register Field Descriptions

Bits	Name	Description
0	ME	Multiple errors. Detecting multiple errors of the same type. An error is considered as multiple error when its detect bit is set and the same error is occurring again. Note that this bit does not track the ordering of when the error occurs. 1 Multiple errors were detected. 0 Multiple errors were not detected.
1–7	—	Reserved
8	PCT	PCI Express completion time-out. A completion time-out condition was detected for a non-posted, outbound PCI Express transaction. An error response is sent back to the requestor. Note that a completion timeout counter only starts when the non-posted request was able to send to the link partner. 1 A completion time-out on the PCI Express link was detected. 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. 0 No completion time-out on the PCI Express link detected.
9	PAT	PCI Express Ack time-out. An internal request timeout was seen. This is considered as a internal fatal error. Once this bit is set, all outstanding transactions will be canceled. No new transactions will be accepted until this bit is clear. It is expected that software needs to perform the proper clean-up operation (that is, hot reset) to get the link to function again before clearing this bit. 1 An Ack time-out on the PCI Express link was detected. 0 No Ack time-out on the PCI Express link detected.
10	PCAC	PCI Express completer abort (CA) completion. A completion with CA status was received. 1 A completion with CA status was detected. 0 No completion with CA status detected.
11	PNM	PCI Express no map. An inbound transaction that is not mapped to any inbound windows was detected. In RC mode, a posted transaction will be dropped silently and this bit will be set. A nonposted transaction will return a completion without data (Cpl) packet with a UR completion status to the requester and this bit is set. For EP mode, a Cpl packet with a UR completion status is sent back to the requester but does not set this bit.This bit is also set when an IO Write or IO Read transaction is received in RC mode and the transaction is not hitting into IO Base/Limit registers. 1 A no-map transaction was detected in RC mode. 0 No no-map transaction detected.

Table 20-48. PCI Express Error Detect Register Field Descriptions (continued)

Bits	Name	Description
12	CDNSC	Completion with data not successful. A completion with data packet was received with a non successful status (that is, UR, CA or CRS status). 1 Completion with data non successful packet was detected. 0 No completion with data non successful packet detected.
13	CRSNC	CRS non configuration. A completion was detected for a non configuration cycle and with CRS status. See Section 20.3.8.1.1, "PCI Express Configuration Access Register Mechanism," for more information. 1 CRS non configuration packet was detected. 0 No CRS non configuration packet detected.
14	ICCA	Invalid PEX_CONFIG_ADDR/PEX_CONFIG_DATA configuration access. Access to an illegal configuration space from PEX_CONFIG_ADDR/PEX_CONFIG_DATA was detected. 1 Invalid CONFIG_ADDR/PEX_CONFIG_DATA access detected 0 No invalid PEX_CONFIG_ADDR/PEX_CONFIG_DATA access detected
15	IACA	Invalid ATMU configuration access. Access to an illegal configuration space from an ATMU window was detected. 1 Invalid ATMU configuration access was detected 0 No invalid ATMU configuration access detected
16	CRST	CRS thresholded. An outbound configuration transaction was retried and thresholded due to a CRS completion status. An error response is sent back to the requestor. See Section 20.3.2.4, "PCI Express Configuration Retry Timeout Register (PEX_CONF_RTY_TOR)," for more information. 1 A CRS threshold condition was detected for an outbound configuration transaction 0 No CRS threshold condition detected
17	MIS	Message invalid size. An outbound message transaction that is greater than 4 bytes or crosses a 4-byte boundary was detected. See Section 20.4.1.9.1, "Outbound ATMU Message Generation," for more information. 1 An invalid size outbound message transaction was detected 0 No invalid size outbound message transaction detected
18	IOIS	I/O invalid size. An outbound I/O transaction that is greater than 4 bytes or crosses a 4-byte boundary was detected. 1 An invalid size outbound I/O transaction was detected 0 No invalid size outbound I/O transaction detected
19	CIS	Configuration invalid size. An outbound configuration transaction that is greater than 4 bytes or crosses a 4-byte boundary was detected. 1 An invalid size outbound configuration transaction was detected 0 No invalid size outbound configuration transaction detected
20	CIEP	Configuration invalid EP. An outbound ATMU configuration transaction request was seen when in EP mode. 1 An outbound configuration transaction while in EP was detected 0 No outbound configuration transaction in EP detected
21	IOIEP	I/O invalid EP. An outbound I/O transaction request was seen when in EP mode. 1 An outbound I/O transaction while in EP was detected 0 No outbound I/O transaction in EP detected
22	OAC	Outbound ATMU crossing. An outbound crossing ATMU transaction was detected. Does not detect crossing if transaction is hitting into a VF window. In addition, behavior is undefined if address is crossing 40-bit boundary. 1 An outbound transaction that hits in one window and crosses overing it was detected 0 No outbound ATMU crossing condition detected

Table 20-48. PCI Express Error Detect Register Field Descriptions (continued)

Bits	Name	Description
23	IOIA	I/O invalid address. An outbound I/O transaction with a translated address of greater than 4 Gbytes was detected. 1 A greater than 4-Gbyte I/O address was detected 0 No greater than 4-Gbyte I/O address detected
24	IMBA	Invalid memory base address. An outbound memory transaction with a translated address not hitting into the Memory Base/Limit register or Prefetchable Memory Base/Limit register. Only valid in RC mode. This feature is enabled by setting PEX_CONFIG[OB_CK]. 1 Invalid memory base address was detected 0 No invalid memory base address detected
25	IIOBA	Invalid I/O base address. An outbound I/O transaction with a translated address not hitting into the IO Base/Limit register or IO Upper Base/Limit register. Only valid in RC mode. This feature is enabled by setting PEX_CONFIG[OB_CK]. 1 Invalid I/O base address was detected 0 No invalid I/O base address detected
26	LDDE	Link down data error. An outbound read transaction was issued through an ATMU window when the link was down. 1 Link down data error detected 0 No Link down data error detected
27	LME	LIODN mapping error. An inbound transaction was not mapped to an LIODN entry in the LIODN permission table. 1 LIODN mapping error detected 0 No LIODN mapping error detected
28	FLRDE	FLR data error. A nonposted transaction was issued toward a PF/VF that has FLR event active. 1 Nonposted transaction issued while FLR event active detected 0 No nonposted transaction issued during FLR event
29	UREP	PCI Express Unsupported Request completion in EP. A completion with UR status was detected when in EP mode. 1 A completion with UR status was detected while in EP mode 0 No completion with UR status while in EP mode detected
30	TAE	Tx Abort error. A nonposted transaction was issued to a device that does not have its bus master enable bit set (EP mode only) or its D-state non-zero (EP/RC). 1 Nonposted transaction issued to devices that are not enabled or in low power state. 0 No nonposted transaction issued to devices that are not enabled or in low power state.
31	—	Reserved

20.3.7.2 PCI Express Error Interrupt Enable Register (PEX_ERR_EN)

The PCI Express error interrupt enable register, shown in Figure 20-53, allows interrupts to be generated when the corresponding PCI Express error detect register bits are set.

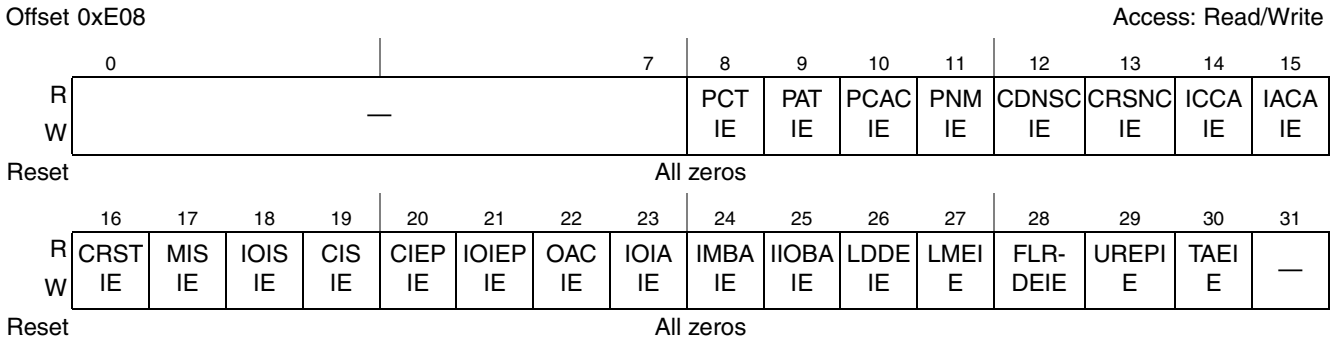


Figure 20-53. PCI Express Error Interrupt Enable Register (PEX_ERR_EN)

Table 20-49 describes the fields of the PCI Express error interrupt enable register.

Table 20-49. PCI Express Error Interrupt Enable Register Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8	PCTIE	PCI Express completion time-out interrupt enable. When set and PEX_ERR_DR[PCT]=1 generates an interrupt. 1 Enable PCI Express completion time-out interrupt generation 0 Disable PCI Express completion time-out interrupt generation
9	PATIE	PCI Express Ack time-out interrupt enable. When set and PEX_ERR_DR[PAT]=1 will generate an interrupt. 1 Enable PCI Express ack time-out interrupt generation 0 Disable PCI Express ack time-out interrupt generation
10	PCACIE	PCI Express CA completion interrupt enable. When set and PEX_ERR_DR[PCAC]=1 generates an interrupt. 1 Enable completion with CA status interrupt generation 0 Disable completion with CA status interrupt generation
11	PNMIE	PCI Express no map interrupt enable. When set and PEX_ERR_DR[PNM]=1 generates an interrupt. 1 Enable no map PCI Express packet interrupt generation 0 Disable no map PCI Express packet interrupt generation
12	CDNSCIE	Completion with data not successful interrupt enable. When this bit is set and PEX_ERR_DR[CDNSC] = 1 generates an interrupt. 1 Enable completion with data non successful interrupt generation 0 Disable completion with data non successful interrupt generation
13	CRSNCIE	CRS non configuration interrupt enable. When this bit is set and PEX_ERR_DR[CRSNC] = 1 generates an interrupt. 1 Enable CRS non configuration interrupt generation 0 Disable CRS non configuration interrupt generation
14	ICCAIE	Invalid PEX_CONFIG_ADDR/PEX_CONFIG_DATA configuration access interrupt enable. When set and PEX_ERR_DR[ICCA]=1 generates an interrupt. 1 Enable invalid PEX_CONFIG_ADDR/PEX_CONFIG_DATA access interrupt generation 0 Disable invalid PEX_CONFIG_ADDR/PEX_CONFIG_DATA access interrupt generation.

Table 20-49. PCI Express Error Interrupt Enable Register Field Descriptions (continued)

Bits	Name	Description
15	IACAIE	Invalid ATMU configuration access. When set and PEX_ERR_DR[IACA]=1 generates an interrupt. 1 Enable invalid ATMU configuration access interrupt generation 0 Disable invalid ATMU configuration access interrupt generation
16	CRSTIE	CRS thresholded interrupt enable. When set and PEX_ERR_DR[CRST]=1 generates an interrupt. 1 Enable CRS threshold interrupt generation 0 Disable CRS threshold interrupt generation
17	MISIE	Message invalid size interrupt enable. When set and PEX_ERR_DR[MIS]=1 generates an interrupt. 1 Enable invalid outbound message size interrupt generation 0 Disable invalid outbound message size interrupt generation
18	IOISIE	I/O invalid size interrupt enable. When set and PEX_ERR_DR[IOIS]=1 generates an interrupt. 1 Enable invalid outbound I/O size interrupt generation 0 Disable invalid outbound I/O size interrupt generation
19	CISIE	Configuration invalid size interrupt enable. When set and PEX_ERR_DR[CIS]=1 generates an interrupt. 1 Enable invalid outbound configuration size interrupt generation 0 Disable invalid outbound configuration size interrupt generation
20	CIEPIE	Configuration invalid EP interrupt enable. When set and PEX_ERR_DR[CIEP]=1 generates an interrupt. 1 Enable outbound configuration transaction while in EP mode interrupt generation 0 Disable outbound configuration transaction in EP mode interrupt generation
21	IOIEPIE	I/O invalid EP interrupt enable. When set and PEX_ERR_DR[IOIEP]=1 generates an interrupt. 1 Enable outbound I/O transaction EP mode interrupt generation 0 Disable outbound I/O transaction EP mode interrupt generation
22	OACIE	Outbound ATMU crossing interrupt enable. When set and PEX_ERR_DR[OAC]=1 generates an interrupt. 1 Enable outbound crossing ATMU interrupt generation 0 Disable outbound crossing ATMU interrupt generation
23	IOIAIE	I/O address invalid enable. When set and PEX_ERR_DR[IOIA]=1 generates an interrupt. 1 Enable greater than 4G I/O address interrupt generation 0 Disable greater than 4G I/O address interrupt generation
24	IMBAIE	Invalid memory base address interrupt enable. When set and PEX_ERR_DR[IMBA]=1 will generate an interrupt. 1 Enable memory base address interrupt generation 0 Disable memory base address interrupt generation
25	IIOBAIE	Invalid I/O base address interrupt enable. When set and PEX_ERR_DR[IIOBA]=1 will generate an interrupt. 1 Enable I/O base address interrupt generation 0 Disable I/O base address interrupt generation
26	LDDEIE	Link down data error interrupt enable. When set and PEX_ERR_DR[LDDEIE]=1 will generate an interrupt. 1 Enable link down data error interrupt generation 0 Disable link down data error interrupt generation
27	LMEIE	LIODN mapping error interrupt enable. When set and PEX_ERR_DR[LME]=1 will generate an interrupt. 1 Enable LIODN mapping error interrupt generation 0 Disable LIODN mapping error interrupt generation
28	FLRDEIE	FLR data error interrupt enable. When set and PEX_ERR_DR[FLRDE]=1 will generate an interrupt. 1 Enable nonposted transaction issued while FLR event interrupt generation. 0 Disable nonposted transaction issued during FLR event interrupt generation.

Table 20-49. PCI Express Error Interrupt Enable Register Field Descriptions (continued)

Bits	Name	Description
29	UREPIE	PCI Express Unsupported Request completion in EP. When set and PEX_ERR_DR[UREP]=1 will generate an interrupt. 1 Enable completion with UR while in EP mode interrupt generation. 0 Disable completion with UR while in EP mode interrupt generation.
30	TAEIE	Tx Abort error. When set and PEX_ERR_DR[TAE]=1 will generate an interrupt. 1 Enable nonposted transaction issued to devices that are not enabled or in low power state interrupt generation. 0 Disable nonposted transaction issued to devices that are not enabled or in low power state interrupt generation .
31	—	Reserved

20.3.7.3 PCI Express Error Disable Register (PEX_ERR_DISR)

The PCI Express error disable register, shown in Figure 20-54, controls the setting of the PCI Express error detect register's bits.

Offset 0xE10 Access: Read/Write

	0	1–7	8	9	10	11	12	13	14	15
R W	MED	—	PCTD	PATD	PCACD	PNMD	CDNSCD	CRSNCD	ICCAD	IACAD
Reset	All zeros									

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CRSTD	MISD	IOISD	CISD	CIEPD	IOIEPD	OACD	IOIAD	IMBAD	IIOBAD	LDDED	LMED	FLRDED	UREPED	TAED	—
Reset	All zeros															

Figure 20-54. PCI Express Error Disable Register (PEX_ERR_DISR)

Table 20-50 describes the fields of the PCI Express error disable register.

Table 20-50. PCI Express Error Disable Register Field Descriptions

Bits	Name	Description
0	MED	Multiple errors disable. When set disables the setting of PEX_ERR_DR[ME] bit. 1 Disable multiple errors detection 0 Enable multiple errors detection
1–7	—	Reserved
8	PCTD	PCI Express completion time-out disable. When set disables the setting of PEX_ERR_DR[PCT] bit. 1 Disable PCI Express completion time-out detection 0 Enable PCI Express completion time-out detection
9	PATD	PCI Express Ack time-out disable. When set will disable the setting of PEX_ERR_DR[PAT] bit. 1 Disable PCI Express ack time-out detection 0 Enable PCI Express ack time-out detection
10	PCACD	PCI Express CA completion disable. When set disables the setting of PEX_ERR_DR[PCAC] bit. 1 Disable completion with CA status detection 0 Enable completion with CA status detection

Table 20-50. PCI Express Error Disable Register Field Descriptions (continued)

Bits	Name	Description
11	PNMD	PCI Express no map disable. When set disables the setting of PEX_ERR_DR[PNM] bit. 1 Disable no map PCI Express packet detection 0 Enable no map PCI Express packet detection
12	CDNSCD	Completion with data not successful disable. When set disables the setting of PEX_ERR_DR[CDNSC] bit. 1 Disable completion with data not successful detection 0 Enable completion with data not successful detection
13	CRSNCD	CRS non configuration disable. When set disables the setting of PEX_ERR_DR[CRSNC] bit. 1 Disable CRS non configuration detection 0 Enable CRS non configuration detection
14	ICCAD	Invalid PEX_CONFIG_ADDR/PEX_CONFIG_DATA configuration access disable. When set disables the setting of PEX_ERR_DR[ICCA] bit. 1 Disable invalid PEX_CONFIG_ADDR/PEX_CONFIG_DATA access detection 0 Enable invalid PEX_CONFIG_ADDR/PEX_CONFIG_DATA access detection
15	IACAD	Invalid ATMU configuration access. When set disables the setting of PEX_ERR_DR[IACA] bit. 1 Disable invalid ATMU configuration access detection 0 Enable invalid ATMU configuration access detection
16	CRSTD	CRS thresholded disable. When set disables the setting of PEX_ERR_DR[CRST] bit. 1 Disable CRS threshold detection 0 Enable CRS threshold detection
17	MISD	Message invalid size disable. When set disables the setting of PEX_ERR_DR[MIS] bit. 1 Disable invalid outbound message size detection 0 Enable invalid outbound message size detection
18	IOISD	I/O invalid size disable. When set disables the setting of PEX_ERR_DR[IOIS] bit. 1 Disable invalid outbound I/O size detection 0 Enable invalid outbound I/O size detection
19	CISD	Configuration invalid size disable. When set disables the setting of PEX_ERR_DR[CIS] bit. 1 Disable invalid outbound configuration size detection 0 Enable invalid outbound configuration size detection
20	CIEPD	Configuration invalid EP disable. When set disables the setting of PEX_ERR_DR[CIEP] bit. 1 Disable outbound configuration transaction EP mode detection 0 Enable outbound configuration transaction EP mode detection
21	IOIEPD	I/O invalid EP disable. When set disables the setting of PEX_ERR_DR[IOEP] bit. 1 Disable outbound I/O transaction EP mode detection 0 Enable outbound I/O transaction EP mode detection
22	OACD	Outbound ATMU crossing disable. When set disables the setting of PEX_ERR_DR[OAC] bit. 1 Disable outbound crossing ATMU detection 0 Enable outbound crossing ATMU detection
23	IOIAD	I/O invalid address disable. When set disables the setting of PEX_ERR_DR[IOIA] bit. 1 Disable greater than 4G I/O address detection 0 Enable greater than 4G I/O address detection
24	IMBAIE	Invalid memory base address disable. When set will disable the setting of PEX_ERR_DR[IMBA]. 1 Disable memory base address detection 0 Enable memory base address detection

Table 20-50. PCI Express Error Disable Register Field Descriptions (continued)

Bits	Name	Description
25	IIOBAIE	Invalid I/O base address disable. When set will disable the setting of PEX_ERR_DR[IIOBA]. 1 Disable I/O base address detection 0 Enable I/O base address detection
26	LDDED	Link down data error disable. When set will disable the setting of PEX_ERR_DR[LDDE]. 1 Disable link down data error detection 0 Enable link down data error detection
27	LMED	LIODN mapping error disable. When set will disable the setting of PEX_ERR_DR[LME]. 1 Disable LIODN mapping error detection 0 Enable LIODN mapping error detection
28	FLRDED	FLR data error disable. When set will disable the setting of PEX_ERR_DR[FLRDE]. 1 Disable nonposted transaction issued during FLR detection 0 Enable nonposted transaction issued while FLR detection
29	UREPED	Completion with UR disable. When set will disable the setting of PEX_ERR_DR[UREP]. 1 Disable completion with UR while in EP mode detection 0 Enable completion with UR while in EP mode detection
30	TAED	Tx abort disable. When set will disable the setting of PEX_ERR_DR[TAE]. 1 Disable nonposted transaction issued to devices that are not enabled or in low power state detection 0 Enable nonposted transaction issued to devices that are not enabled or in low power state detection
31	—	Reserved

20.3.7.4 PCI Express Error Capture Status Register (PEX_ERR_CAP_STAT)

The PCI Express error capture status register, shown in Figure 20-55, allows vital error information to be captured when an error occurs. Note that no further error capturing is performed until the ECV bit is cleared.

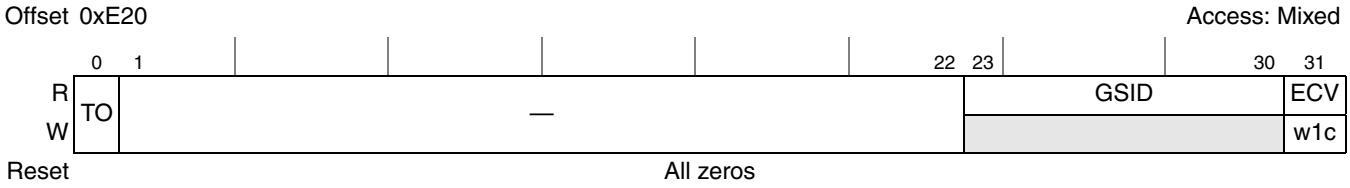


Figure 20-55. PCI Express Error Capture Status Register (PEX_ERR_CAP_STAT)

Table 20-51 describes the fields of the PCI Express error capture status register.

Table 20-51. PCI Express Error Capture Status Register Field Descriptions

Bits	Name	Description
0	TO	Transaction originator. This field Indicates whether the originator of the transaction is from PEX_CONFIG_ADDR/PEX_CONFIG_DATA. 1 Transaction originated from PEX_CONFIG_ADDR/PEX_CONFIG_DATA. 0 Transaction not originated from PEX_CONFIG_ADDR/PEX_CONFIG_DATA.
1–21	—	Reserved

Table 20-51. PCI Express Error Capture Status Register Field Descriptions (continued)

Bits	Name	Description
23–30	GSID	Global source ID. This field indicates the internal platform global source ID that the error transaction originates. This field only applies to non PEX_CONFIG_ADDR/PEX_CONFIG_DATA transactions. TBD
31	ECV	Error capture valid. This bit indicates that the capture registers 0-3 contain valid info. This bit when set indicates that the captured registers contain valid capturing information. No new capturing is done unless this bit is cleared by writing a 1 to it.

20.3.7.5 PCI Express Error Capture Register 0 (PEX_ERR_CAP_R0)

Together with the other PCI Express error capture registers, PEX_ERR_CAP_R0 allows vital error information to be captured when an error occurs. Different error information is reported depending on whether the error source is from an outbound transaction from an internal source or from an inbound transaction from an external source; the source of the captured error is reflected in PEX_ERR_CAP_STAT[GSID]. Note that after the initial error is captured, no further capturing is performed until the PEX_ERR_CAP_STAT[ECV] bit is clear.

20.3.7.5.1 PEX_ERR_CAP_R0—Outbound Case

PEX_ERR_CAP_R0 for the case when the error is caused by an outbound transaction from an internal source and the error is due to timeout condition or PEX_CONFIG_ADDR/PEX_CONFIG_DATA access is shown in Figure 20-56.

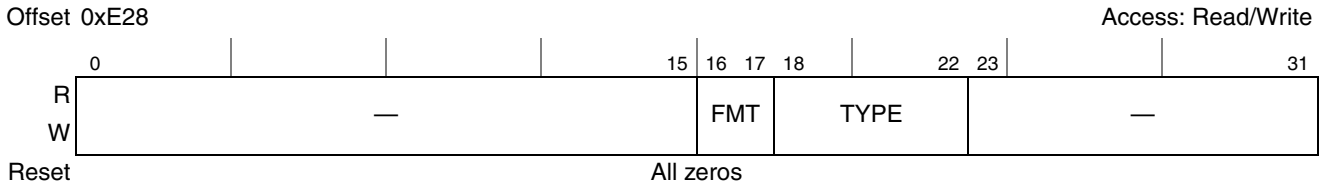


Figure 20-56. PCI Express Error Capture Register 0 (PEX_ERR_CAP_R0) Internal Source, Outbound Transaction

Table 20-52 describes the fields of the PCI Express error capture register 0 for the case when the error is caused by an outbound transaction from an internal source.

Table 20-52. PCI Express Error Capture Register 0 Field Descriptions Internal Source, Outbound Transaction

Bits	Name	Description
0–15	—	Reserved
16-17	FMT	PCI Express format. This field indicates the PCI Express packet format. See the PCI Express base specification for more information on 3 or 4 DW (4-byte) header format.
18–22	TYPE	PCI Express type. This field indicates the PCI express packet type. See the PCI Express base specification for more information on 3 or 4 DW (4-byte) header format.
23–31	—	Reserved

20.3.7.5.2 PEX_ERR_CAP_R0—Inbound Case

PEX_ERR_CAP_R0 for the case when the error is caused by an inbound transaction from an external source, is shown in Figure 20-57.

Inbound transactions that result in any af the following PEX_ERR_DR bits being set will result in PEX_ERR_CAP_STAT[GSID] indicating the controller itself was the source of the error:

- PEX_ERR_DR[PNM]
- PEX_ERR_DR[PCAC]
- PEX_ERR_DR[CDNSC]
- PEX_ERR_DR[CRSNC]
- PEX_ERR_DR[LME]
- PEX_ERR_DR[UREP]

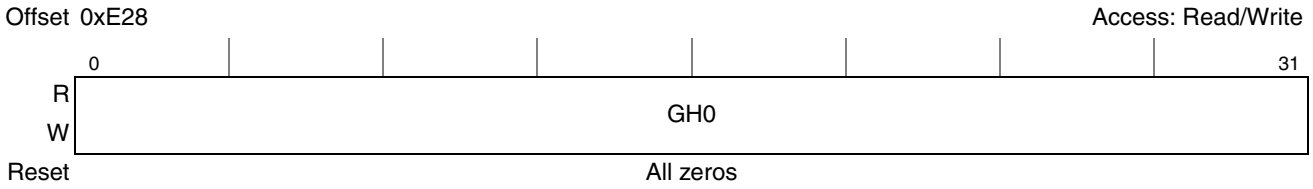


Figure 20-57. PCI Express Error Capture Register 0 (PEX_ERR_CAP_R0) External Source, Inbound Transaction

Table 20-53 describes the fields of PEX_ERR_CAP_R0 for the case when the error is caused by an inbound transaction from an external source.

Table 20-53. PCI Express Error Capture Register 0 Field Descriptions External Source, Inbound Transaction

Bits	Name	Description
0–31	GH0	PCI Express first DW (4-byte) header. This field contains the first DW (4-byte) of the captured PCI Express packet header. 27–31 TYPE 25–26 FMT 20–24 Reserved 17–19 TC 16 Reserved 14–15 LENGTH[9:8] 12–13 Reserved 10–11 ATTR 9 EP 8 TD 0–7 LENGTH[7:0]

20.3.7.6 PCI Express Error Capture Register 1 (PEX_ERR_CAP_R1)

Together with the other PCI Express error capture registers, PEX_ERR_CAP_R1 allows vital error information to be captured when an error occurs. Different error information is reported depending on whether the error source is from an outbound transaction from an internal source or from an inbound

transaction from an external source; the source of the captured error is reflected in PEX_ERR_CAP_STAT[GSID]. Note that after the initial error is captured, no further capturing is performed until the PEX_ERR_CAP_STAT[ECV] bit is clear.

20.3.7.6.1 PEX_ERR_CAP_R1—Outbound Case

PEX_ERR_CAP_R1 for the case when the error is caused by an outbound transaction from an internal source is shown in Figure 20-58.

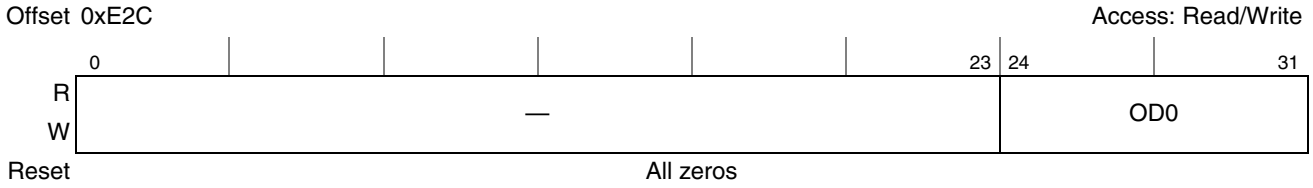


Figure 20-58. PCI Express Error Capture Register 1 (PEX_ERR_CAP_R1) Internal Source, Outbound Transaction

Table 20-54 describes the fields of PEX_ERR_CAP_R1 for the case when the error is caused by an outbound transaction from an internal source.

Table 20-54. PCI Express Error Capture Register 1 Field Descriptions Internal Source, Outbound Transaction

Bits	Name	Description
0–23	—	Reserved
24–31	OD0	Internal platform transaction information. Reserved for factory debug.

20.3.7.6.2 PEX_ERR_CAP_R1—Inbound Case

PEX_ERR_CAP_R1 for the case when the error is caused by an inbound transaction from an external source, is shown in Figure 20-59.

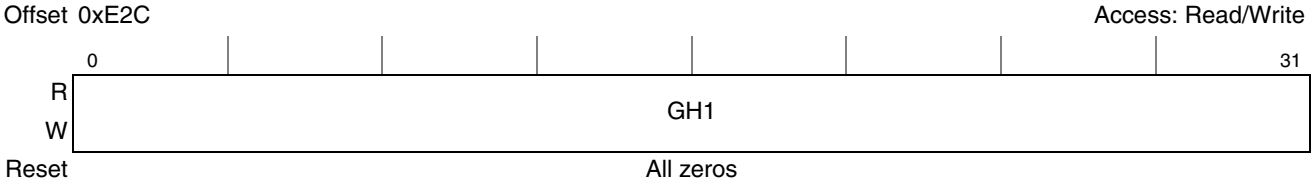


Figure 20-59. PCI Express Error Capture Register 1 (PEX_ERR_CAP_R1) External Source, Inbound Transaction

Table 20-55 describes the fields of PEX_ERR_CAP_R1 for the case when the FMT and TYPE subfields in PEX_ERR_CAP_R0 (see Table 20-53) indicate the error was caused by an inbound completion transaction.

Table 20-55. PCI Express Error Capture Register 1 Field Descriptions External Source, Inbound Completion Transaction

Bits	Name	Description
0–31	GH1	PEX second DW (4-byte) header. This field contains the second DW (4-byte) of the captured PCI Express packet header. 24–31 Comp ID[15:8] 16–23 Comp ID[7:0] 12–15 Byte Count[11:8] 11 BCM 8–10 Comp Status 0–7 Byte Count[7:0]

Table 20-56 describes the fields of PEX_ERR_CAP_R1 for the case when the FMT and TYPE subfields in PEX_ERR_CAP_R0 (see Table 20-53) indicate the error was caused by an inbound memory request transaction.

Table 20-56. PCI Express Error Capture Register 1 Field Descriptions External Source, Inbound Memory Request Transaction

Bits	Name	Description
0–31	GH1	PEX second DW (4-byte) header. This field contains the second DW (4-byte) of the captured PCI Express packet header. 24–31 Requester ID[15:8] 16–23 Requester ID[7:0] 8–15 Tag[7:0] 4–7 First DW BE[3:0] 0–3 Last DW BE[3:0]

20.3.7.7 PCI Express Error Capture Register 2 (PEX_ERR_CAP_R2)

Together with the other PCI Express error capture registers, PEX_ERR_CAP_R2 allows vital error information to be captured when an error occurs. Different error information is reported depending on whether the error source is from an outbound transaction from an internal source or from an inbound transaction from an external source; the source of the captured error is reflected in PEX_ERR_CAP_STAT[GSID]. Note that after the initial error is captured, no further capturing is performed until the PEX_ERR_CAP_STAT[ECV] bit is clear.

20.3.7.7.1 PEX_ERR_CAP_R2—Outbound Case

PEX_ERR_CAP_R2 for the case when the error is caused by an outbound transaction from an internal source is shown in Figure 20-60.

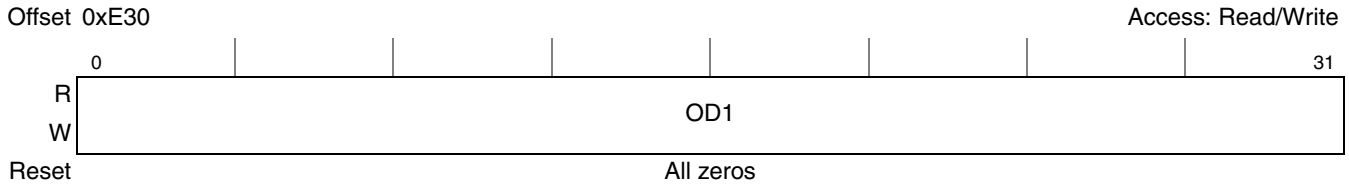


Figure 20-60. PCI Express Error Capture Register 2 (PEX_ERR_CAP_R2) Internal Source, Outbound Transaction

Table 20-57 describes the fields of PEX_ERR_CAP_R2 for the case when the error is caused by an outbound transaction from an internal source.

Table 20-57. PCI Express Error Capture Register 2 Field Descriptions Internal Source, Outbound Transaction

Bit	Name	Description
0–31	OD1	Internal platform transaction information. Reserved for factory debug.

20.3.7.7.2 PEX_ERR_CAP_R2—Inbound Case

PEX_ERR_CAP_R2 for the case when the error is caused by an inbound transaction from an external source is shown in Figure 20-61.

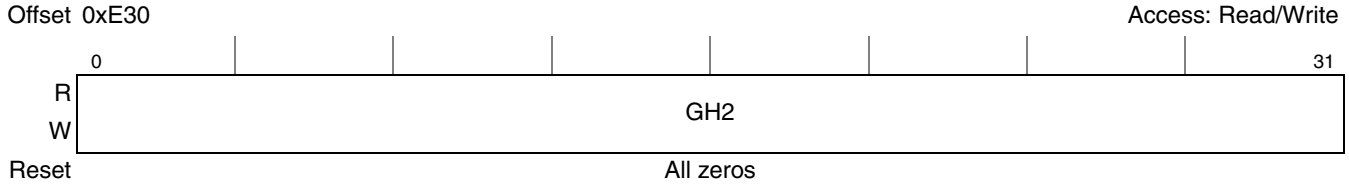


Figure 20-61. PCI Express Error Capture Register 2 (PEX_ERR_CAP_R2) External Source, Inbound Transaction

Table 20-58 describes the fields of PEX_ERR_CAP_R2 for the case when the FMT and TYPE subfields in PEX_ERR_CAP_R0 (see Table 20-53) indicate the error was caused by an inbound completion transaction.

Table 20-58. PCI Express Error Capture Register 2 Field Descriptions External Source, Inbound Completion Transaction

Bits	Name	Description
0–31	GH2	PEX third DW (4-byte) header. This field contains the third DW (4-byte) of the captured PCI Express packet header. 24–31 Req ID[15:8] 16–23 Req ID[7:0] 8–15 Tag[7:0] 1–7 Lower Address[6:0] 0 Reserved

Table 20-59 describes the fields of PEX_ERR_CAP_R2 for the case when the FMT and TYPE subfields in PEX_ERR_CAP_R0 (see Table 20-53) indicate the error was caused by an inbound memory request transaction. Note that PEX_ERR_CAP_R2 captures the 32-bit address for a 3 DW memory request header or the upper half of the 64-bit address for a 4 DW memory request header; the lower half of the 64-bit address for a 4 DW memory request header is captured in PEX_ERR_CAP_R3.

Table 20-59. PCI Express Error Capture Register 2 Field Descriptions
External Source, Inbound Memory Request Transaction

Bits	Name	Description	
		3 DW Header	4 DW Header
0–31	GH2	PEX third DW (4-byte) header. This field contains the third DW (4-byte) of the captured PCI Express packet header.	
		24–31 Address[31:24] 16–23 Address[23:16] 8–15 Address[15:8] 6–7 Reserved 0-5 Address[7:2]	24–31 Address[63:56] 16–23 Address[55:48] 8–15 Address[47:40] 0-7 Address[39:32]

20.3.7.8 PCI Express Error Capture Register 3 (PEX_ERR_CAP_R3)

Together with the other PCI Express error capture registers, PEX_ERR_CAP_R3 allows vital error information to be captured when an error occurs. Different error information is reported depending on whether the error source is from an outbound transaction from an internal source or from an inbound transaction from an external source; the source of the captured error is reflected in PEX_ERR_CAP_STAT[GSID]. Note that after the initial error is captured, no further capturing is performed until the PEX_ERR_CAP_STAT[ECV] bit is clear.

20.3.7.8.1 PEX_ERR_CAP_R3—Outbound Case

PEX_ERR_CAP_R3 for the case when the error is caused by an outbound transaction from an internal source is shown in Figure 20-62.

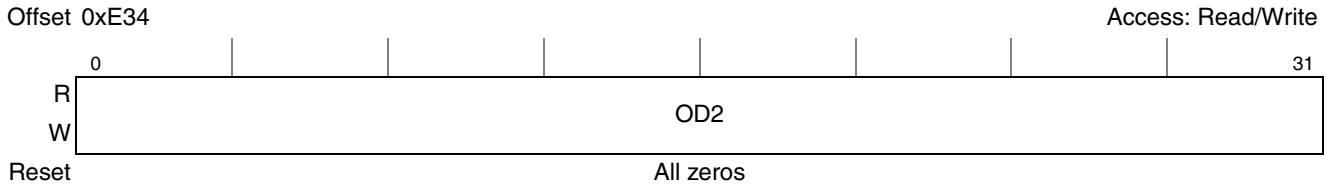


Figure 20-62. PCI Express Error Capture Register 3 (PEX_ERR_CAP_R3)
Internal Source, Outbound Transaction

Table 20-60 describes the fields of PEX_ERR_CAP_R3 for the case when the error is caused by an outbound transaction from an internal source.

Table 20-60. PCI Express Error Capture Register 3 Field Descriptions Internal Source, Outbound Transaction

Bits	Name	Description
0–31	OD2	Internal platform transaction information. Reserved for factory debug.

20.3.7.8.2 PEX_ERR_CAP_R3—Inbound Case

PEX_ERR_CAP_R3 for the case when the error is caused by an inbound transaction from an external source is shown in Figure 20-63.

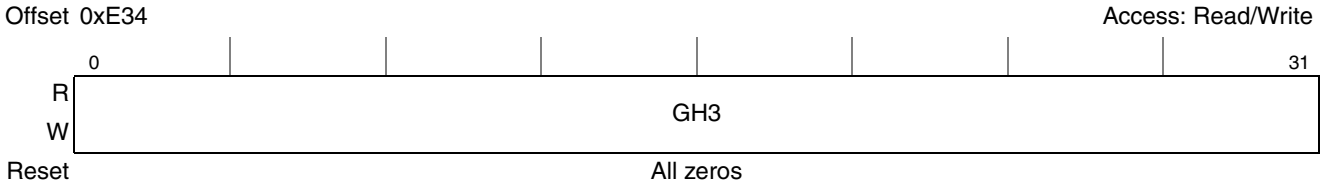


Figure 20-63. PCI Express Error Capture Register 3 (PEX_ERR_CAP_R3) External Source, Inbound Transaction

Table 20-61 describes the fields of PEX_ERR_CAP_R3 for the case when the FMT and TYPE subfields in PEX_ERR_CAP_R0 (see Table 20-53) indicate the error was caused by an inbound memory request transaction. Note that PEX_ERR_CAP_R3 captures the lower half of the 64-bit address for a 4 DW memory request header; the upper half of the 64-bit address for a 4 DW memory request header or the 32-bit address for a 3 DW memory request header is captured in PEX_ERR_CAP_R2.

Table 20-61. PEX Error Capture Register 3 Field Descriptions External Source, Inbound Memory Request Transaction

Bits	Name	Description
0–31	GH3	PEX fourth DW (4-byte) header. This field contains the fourth DW (4-byte) of the captured PCI Express packet header. 24–31 Address[31:24] 16–23 Address[23:16] 8–15 Address[15:8] 6–7 Reserved 0-5 Address[7:2]

20.3.7.9 PCI Express Control/Status Register 0 (PEX_CSR0)

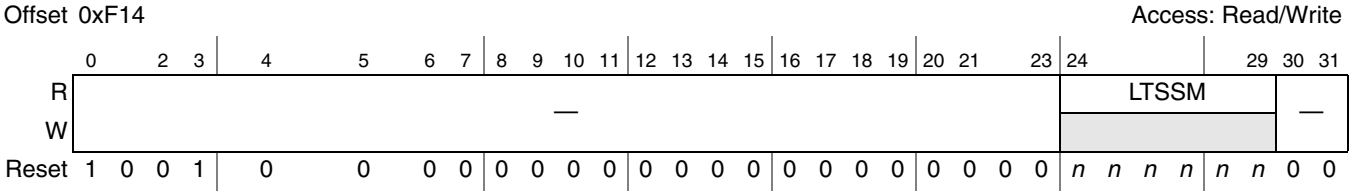


Figure 20-64. PCI Express Control Status Register 0 (PEX_CSR0)

Figure 20-62 describes the fields of the PCI Express control/status register 0.

Table 20-62. PEX_CSR0 Field Descriptions

Bit	Name	Description
0-23	—	Reserved. Do not write to reserved fields.
24-29	LTSSM_SC	Link training and status state machine status code. 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_SC multiple times to ensure the value has stabilized. 00h DETECT_QUIET 01h DETECT_ACTIVE 02h POLL_ACTIVE 03h POLL_COMPLIANCE 04h POLL_CONFIG 05h PRE_DETECT_QUIET 06h DETECT_WAIT 07h CFG_LINKWD_START 08h CFG_LINKWD_ACEPT 09h CFG_LANENUM_WAIT 0Ah CFG_LANENUM_ACEPT 0Bh CFG_COMPLETE 0Ch CFG_IDLE 0Dh RCVRY_LOCK 0Eh RCVRY_SPEED 0Fh RCVRY_RCVRCFG 10h RCVRY_IDLE 11h L0 12h L0S 13h L123_SEND_EIDLE 14h L1_IDLE 15h L2_IDLE 16h L2_WAKE 17h DISABLED_ENTRY 18h DISABLED_IDLE 19h DISABLED 1Ah LPBK_ENTRY 1Bh LPBK_ACTIVE 1Ch LPBK_EXIT 1Dh LPBK_EXIT_TIMEOUT 1Eh HOT_RESET_ENTRY 1Fh HOT_RESET 20h RCVRY_EQ0 21h RCVRY_EQ1 22h RCVRY_EQ3
30-31	—	Reserved

20.3.7.10 PCI Express Control/Status 1 (PEX_CSR1)

The PCI Express control/status 1, contains interrupt mask for the controller.

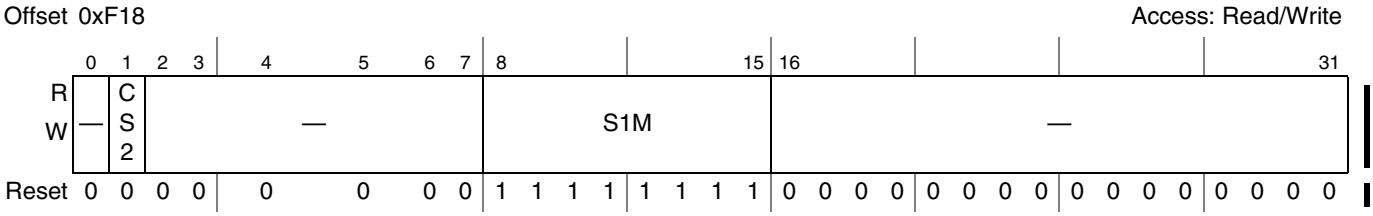


Figure 20-65. PCI Express Control/Status Register 1 (PEX_CSR1)

Figure 20-63 describes the fields of the PCI Express control/status register 1.

Table 20-63. PEX_CSR1 Field Descriptions

Bit	Name	Description
0		Reserved
1	CS2	DBI CS2 enable. When enable will allow software to alter the mask bits of the bar registers.
0		Reserved
2-7		Reserved
8-15	S1M	Status 1 Mask. When set, will disable the corresponding interrupts from the PCI Express Secondary Status register.
16-31		Reserved

20.3.8 PCI Express Configuration Space Access

There are two methods of accessing the PCI Express configuration header:

- PCI Express outbound ATMU window
- PCI Express configuration access registers (PEX_CONFIG_ADDR/PEX_CONFIG_DATA)

20.3.8.1 RC Configuration Register Access

To access internal configuration space, software must rely on the PCI Express configuration access register (PEX_CONFIG_ADDR/ PEX_CONFIG_DATA) mechanism. To access external configuration space, software can either use configuration access registers or the outbound ATMU mechanism. For the configuration access register method, a value must be written to the PEX_CONFIG_ADDR register that specifies the targeted PCI Express bus, the targeted device on that bus, the targeted function within the device, and the configuration register in that device that should be accessed. The PCI Express controller's bus number is obtained from the PCI Express configuration header (type 1). Then either a write or a read to the PEX_CONFIG_DATA register triggers the actual write or read cycle to the configuration space. Note that accesses to the little-endian PCI Express configuration space must be properly formatted. See Section 20.4.1.2.3, "Byte Order for Configuration Transactions," for more information.

Note that external configuration transactions should not be attempted until the link has successfully trained. Software can poll the LTSSM state status register (PEX_CSR0) to check the status of link training before issuing external configuration requests.

20.3.8.1.1 PCI Express Configuration Access Register Mechanism

There are two types of configuration transactions (Type 0 and Type 1) needed to support hierarchical bridges.

- If the targeted bus number, and targeted device number equal to the PCI Express controller's bus number and device number, and the targeted function number is zero, then an internal PCI Express configuration cycle access is performed.
- If the targeted bus number does not equal the PCI Express controller's bus number, but does equal the secondary bus number (from the type 1 header) and the targeted device number is 0, then a Type 0 configuration transaction is sent to the PCI Express link.
- If the targeted bus number does not equal the PCI Express controller's bus number, and does not equal the secondary bus number (from the type 1 header), and the targeted bus number is less than or equal to the subordinate bus number (from the type 1 header), then a Type 1 configuration transaction is sent to the PCI Express link.
- If none of the above conditions occur, then the PCI Express controller returns all 1s for reads and ignores writes. In this case, PEX_ERR_DR[ICCA] is set.

20.3.8.1.2 Outbound ATMU Configuration Mechanism (RC-Only)

Software can also program one of the outbound ATMU windows to perform a configuration access. This is accomplished by programming the ReadTType or WriteTType field of the desired PEXOWAR to 0x2. Software must only issue 4-byte or less access to the ATMU configuration window and the access cannot cross a 4-byte boundary. The targeted bus number, targeted device number, targeted function number, register, and targeted extended register number sent are decoded from the outbound translated PCI Express address.

- targeted bus number[7:0] = PCI Express address[27:20]
- targeted device number[4:0] = PCI Express address[19:15]
- targeted function number[2:0] = PCI Express address[14:12]
- targeted extended register number[3:0] = PCI Express address[11:8]
- targeted register number[5:0] = PCI Express address[7:2]

A Type 0 configuration cycle is sent to the link if the targeted bus number equals the secondary bus number (from the type 1 header) and targeted device number is 0. A Type 1 configuration cycle is sent to the link if targeted bus number does not equal primary bus and secondary bus numbers and it is less than or equal to the subordinate bus number (from the type 1 header). For all other cases, the PCI Express controller squashes the write and read results in a response with error returned. In this case, PEX_ERR_DR[IACA] is set.

Note that the PCI Express controller does not support access to its internal configuration registers using the outbound ATMU mechanism. That is, the outbound ATMU mechanism must not be used to program the internal registers.

20.3.8.2 EP Configuration Register Access

When the PCI Express controller is configured as an EP device it responds to remote host generated configuration cycles. This is indicated by decoding the configuration command along with type 0 access in the packet. A remote host can access all of the PCI Express configuration area except the PCI Express Controller Internal CSR registers in the extended PCI Express configuration space at offsets 0x400–0x6FF. The PCI Express Controller Internal CSR registers are not accessible by inbound PCI Express configuration transactions. Attempts to access these registers return all zeros.

While in EP mode, the PCI Express controller does not support generating configuration accesses as a master. All accesses to PEX_CONFIG_ADDR/PEX_CONFIG_DATA cause the device to access the internal configuration registers regardless of the targeted bus number or targeted device number programmed in the PEX_CONFIG_ADDR register. There is no configuration mechanism supported in EP mode using the ATMU window. If the outbound ATMU window is configured to issue a configuration transaction, all posted transactions hitting this window are ignored and all non-posted transactions get a response with an error.

20.3.9 PCI Compatible Configuration Headers

The first 64 bytes of the 256-byte PCI compatible configuration space consists of a predefined header that every PCI device must support. The first 16 bytes of the predefined header are defined the same for all PCI Express devices. These common registers are shown in Figure 20-66.

Reserved

				Address Offset (Hex)
Device ID		Vendor ID		00
Status		Command		04
Class Code			Revision ID	08
BIST	Header Type	Latency Timer	Cache Line Size	0C

Figure 20-66. PCI Express PCI-Compatible Configuration Header Common Registers

The remaining 48 bytes of the header may have differing layouts depending on the function of the device. There are two header types applicable to PCI Express. Type 0 headers are typically used by endpoints; Type 1 headers are used by root complexes and switches/bridges.

20.3.9.1 Common PCI Compatible Configuration Header Registers

This section details the registers that are common to both type 0 and type 1 configuration headers.

20.3.9.1.1 PCI Express Vendor ID Register—Offset 0x00

The vendor ID register, shown in Figure 20-67, is used to identify the manufacturer of the device.

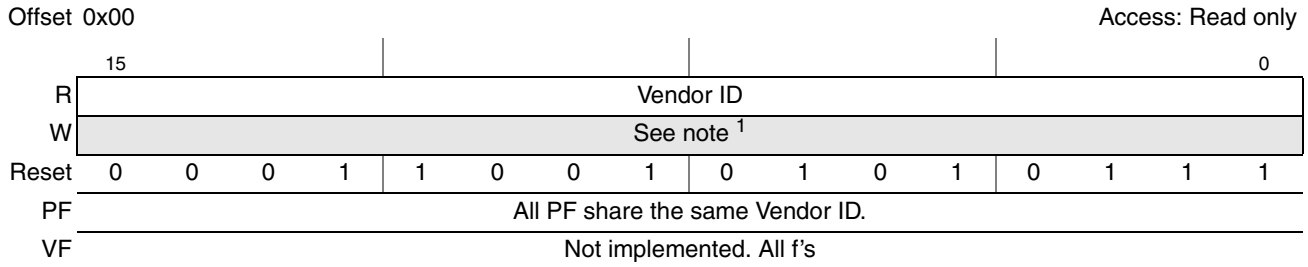


Figure 20-67. PCI Express Vendor ID Register

1 This register is writeable using internal PEX_CONFIG_ADDR/PEX_CONFIG_DATA accesses, but is read-only from inbound configuration accesses by an external host.

Table 20-64 describes the vendor ID register fields.

Table 20-64. PCI Express Vendor ID Register Field Description

Bits	Name	Description
15–0	Vendor ID	0x1957 (Freescale)

20.3.9.1.2 PCI Express Device ID Register—Offset 0x02

The device ID register, shown in Figure 20-68, is used to identify the device.

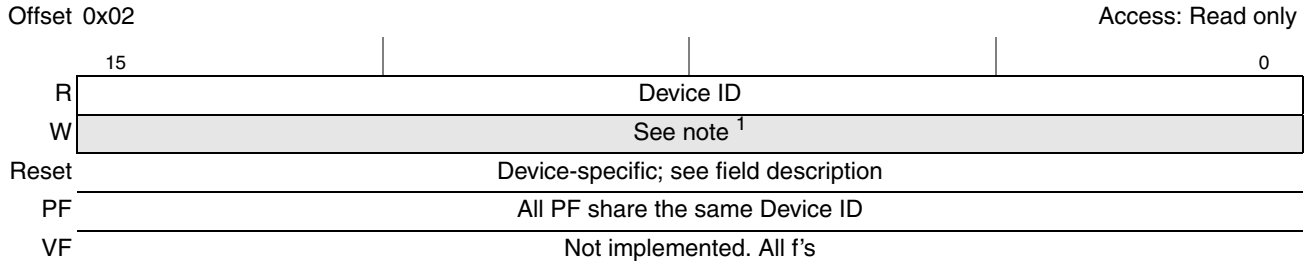


Figure 20-68. PCI Express Device ID Register

1 This register is writeable using internal PEX_CONFIG_ADDR/PEX_CONFIG_DATA accesses, but is read-only from inbound configuration accesses by an external host.

Table 20-65 describes the device ID register fields.

Table 20-65. PCI Express Device ID Register Field Description

Bits	Name	Description
15–0	Device ID	0x0830 T2080 with security 0x0831 T2080 without security 0x0838 T2081 with security 0x0839 T2081 without security

20.3.9.1.3 PCI Express Command Register—Offset 0x04

The command register, shown in Figure 20-69, provides control over the ability to generate and respond to PCI Express cycles.

Offset 0x04 Access: Mixed

	15–11	10	9	8	7	6	5–3	2	1	0
R W	—	Interrupt Disable	—	SERR	—	Parity error response	—	Bus master	Memory space	I/O space
Reset	All zeros									
PF	Implemented per PF as specified in base specification									
VF	0 0 0 0	0 Not implemented	0	From PF	0	From PF	0 0	0 Base	Not implemented	

Figure 20-69. PCI Express Command Register

Table 20-66 describes the bits of the command register.

Table 20-66. PCI Express Command Register Field Descriptions

Bits	Name	Description
15–11	—	Reserved
10	Interrupt Disable	Controls the ability to generate INTx interrupt messages. 0 Enables INTx interrupt messages 1 Disables INTx interrupt messages Any INTx emulation interrupts already asserted by this device must be deasserted when this bit is set.
9	—	Reserved
8	SERR	Controls the reporting of fatal and non-fatal errors detected by the device to the root complex. 0 Disables reporting 1 Enables reporting **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 Section 20.3.10.13, “PCI Express Device Control Register—0x78,” and the advance error reporting capability structure described in sections 20.3.11.1 through 20.3.11.12.
7	—	Reserved
6	Parity error response	Controls whether this PCI Express controller responds to parity errors. 0 Parity errors are ignored and normal operation continues. 1 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. **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 Section 20.3.10.13, “PCI Express Device Control Register—0x78,” and the advance error reporting capability structure described in sections 20.3.11.1 through 20.3.11.12.
5–3	—	Reserved
2	Bus master	Indicates whether this PCI Express device is configured as a master. 0 Disables the ability to generate PCI Express accesses 1 Enables this PCI Express controller to behave as a PCI Express bus 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.

Table 20-66. PCI Express Command Register Field Descriptions (continued)

Bits	Name	Description
1	Memory space	Controls whether this PCI Express device (as a target) responds to memory accesses. 0 This PCI Express device does not respond to PCI Express memory space accesses. 1 This PCI Express device responds to PCI Express memory space 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
0	I/O space	I/O space. 0 This PCI Express device (as a target) does not respond to PCI Express I/O space accesses. 1 This PCI Express device (as a target) does respond to PCI Express I/O space accesses. 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.

20.3.9.1.4 PCI Express Status Register—Offset 0x06

The status register, shown in Figure 20-70, is used to record status information for PCI Express related events.

Offset 0x06 Access: Mixed

	15	14	13	12	11	10	9	8
R	Detected parity error	Signaled system error	Received master-abort	Received target-abort	Signaled target-abort	—		Master data parity error
W	w1c	w1c	w1c	w1c	w1c			w1c
Reset	All zeros							
PF	Implemented per PF as specified in base specification							
VF	Not implemented. All zeros.							

	7	5	4	3	2	0
R	—		Capabilities list	Interrupt Status	—	
W						
Reset	0	0	0	1	0	0	0	0
PF	Implemented per PF as specified in base specification							
VF	Not implemented. All zeros.							

Figure 20-70. PCI Express Status Register

The definition of each bit is given in Table 20-67.

Table 20-67. PCI Express Status Register Field Descriptions

Bits	Name	Description
15	Detected parity error[1]	Set whenever a device receives a poisoned TLP regardless of the state of bit 6 in the command register.
14	Signaled system error[1]	Set whenever a device sends a ERR_FATAL or ERR_NONFATAL message and the SERR enable bit in the command register is set.
13	Received master-abort[1]	Set whenever a requestor receives a completion with unsupported request completion status.

Table 20-67. PCI Express Status Register Field Descriptions (continued)

Bits	Name	Description
12	Received target-abort[1]	Set whenever a device receives a completion with completer abort completion status.
11	Signaled target-abort[1]	Set whenever a device completes a request using completer abort completion status.
10–9	—	Reserved
8	Master data parity error detected[1]	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.
7–5	—	Reserved
4	Capabilities List	All PCI Express devices are required to implement the PCI Express capability structure.
3	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 Section 20.3.10.13, "PCI Express Device Control Register—0x78," and the advance error reporting capability structure described in sections 20.3.11.1 through 20.3.11.12.

20.3.9.1.5 PCI Express Revision ID Register—Offset 0x08

The revision ID register, shown in Figure 20-71, is used to identify the revision of the device.

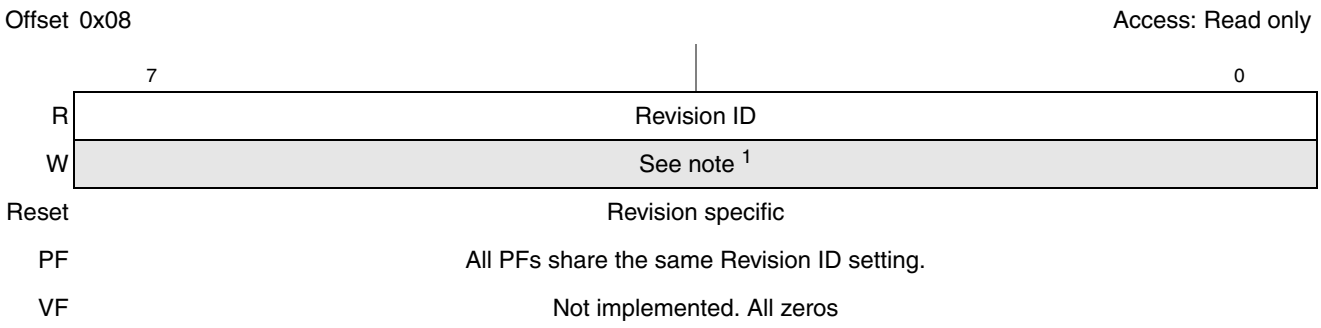


Figure 20-71. PCI Express Revision ID Register

[1] This register is writeable using internal PEX_CONFIG_ADDR/PEX_CONFIG_DATA accesses, but is read-only from inbound configuration accesses by an external host.

Table 20-68 describes the revision ID register fields.

Table 20-68. PCI Express Revision ID Register Field Descriptions

Bits	Name	Description
7–0	Revision ID	Revision specific.

20.3.9.1.6 PCI Express Class Code Register—Offset 0x09

The class code register, shown in Figure 20-72, 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.

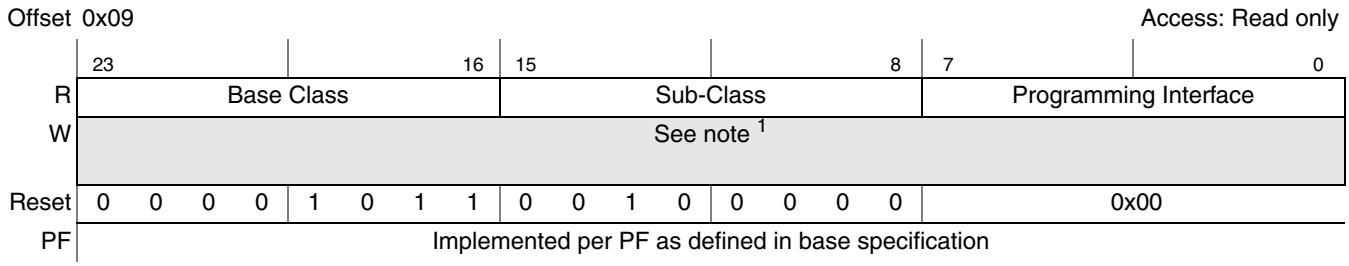


Figure 20-72. PCI Express Class Code Register

1 This register is writeable using internal PEX_CONFIG_ADDR/PEX_CONFIG_DATA accesses, but is read-only from inbound configuration accesses by an external host.

Table 20-69 describes the class code register fields.

Table 20-69. PCI Express Class Code Register Field Descriptions

Bits	Name	Description
23–16	Base Class	0x0B—Processor
15–8	Sub-Class	0x20—PowerPC
7–0	Programming Interface	0x00

20.3.9.1.7 PCI Express Cache Line Size Register—Offset 0x0C

The cache line size register, shown in Figure 20-73, is provided for legacy compatibility purposes (PCI 2.3); it is not used for PCI Express device functionality.

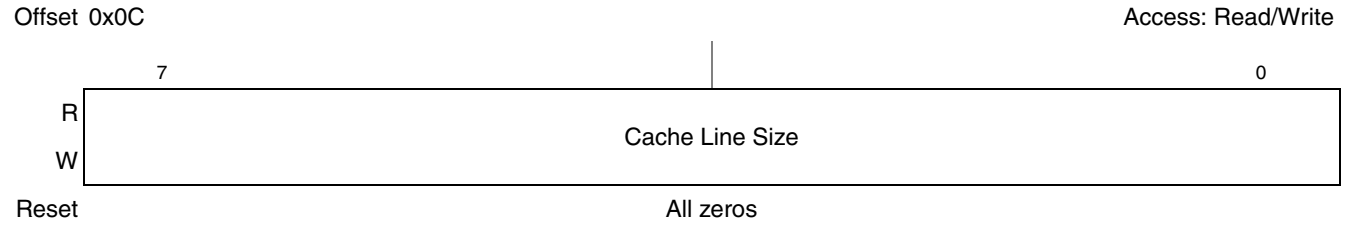


Figure 20-73. PCI Express Bus Cache Line Size Register

Table 20-70 describes the cache line size register.

Table 20-70. PCI Express Bus Cache Line Size Register Field Descriptions

Bits	Name	Description
7–0	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.

20.3.9.1.8 PCI Express Latency Timer Register—0x0D

The latency timer register, shown in Figure 20-74, is provided for legacy compatibility purposes (PCI 2.3); it is not used for PCI Express device functionality.

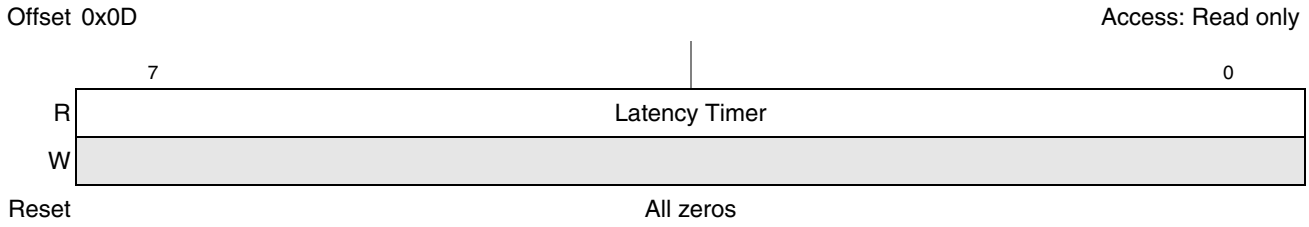


Figure 20-74. PCI Express Bus Latency Timer Register

Table 20-71 describes the PCI Express latency timer register (PLTR).

Table 20-71. PCI Express Bus Latency Timer Register Field Descriptions

Bits	Name	Description
7–0	Latency Timer	Note that for PCI Express operation this register is ignored.

20.3.9.1.9 PCI Express Header Type Register—0x0E

The PCI Express header type register, shown in Figure 20-75, is used to identify the layout of the PCI compatible header.

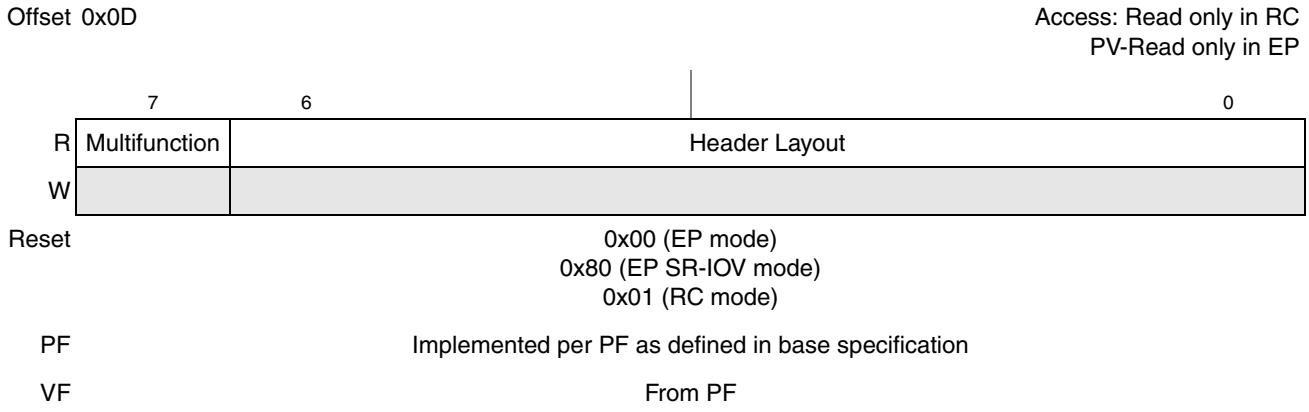


Figure 20-75. PCI Express Header Type Register

Table 20-72 describes the PCI Express header type register.

Table 20-72. PCI Express Header Type Register Field Descriptions

Bits	Name	Description
7	Multifunction	Identifies whether a device supports multiple functions 0 Single function device 1 Multiple function device
6–0	Header Layout	0x00 Endpoint. See Figure 20-76 for type 0 layout. 0x01 Root Complex. See Figure 20-90 for type 1 layout. All other encodings reserved.

20.3.9.1.10 PCI Express BIST Register—0x0F

The BIST register is optional and reserved on the PCI Express controller.

20.3.9.2 Type 0 Configuration Header

The type 0 header is shown in Figure 20-76.

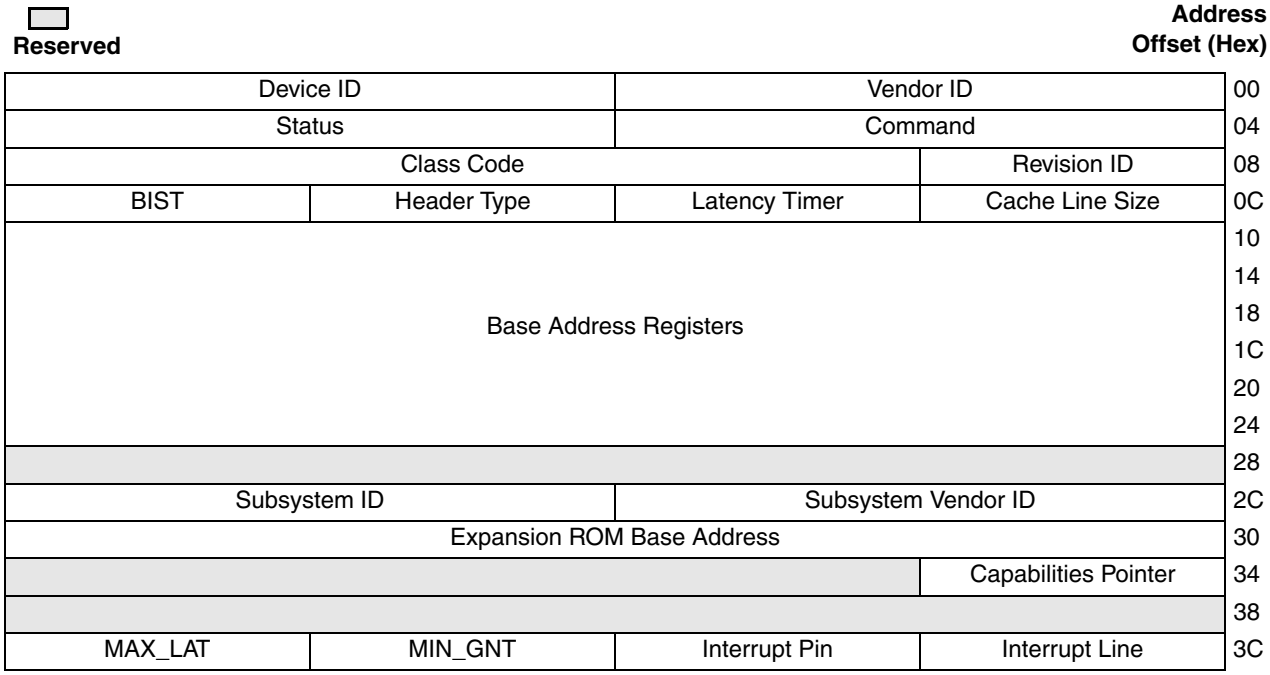


Figure 20-76. PCI Express PCI-Compatible Configuration Header—Type 0

Section 20.3.9.1, "Common PCI Compatible Configuration Header Registers," describes the registers in the first 16 bytes of the header. This section describes the registers that are unique to the type 0 header beginning at offset 0x10.

20.3.9.2.1 PCI Express Base Address Registers—0x10–0x27

The PCI Express base address registers (BARs) point to the beginning of distinct address ranges which the device should claim. In EP mode, the device supports a configuration space BAR, a 32-bit memory space

BAR, and two 64-bit memory space BARs. In RC mode, the device only supports the configuration space BAR in the header; the other memory spaces are defined by the inbound ATMUs. Refer to Section 20.3.5.4, "PCI Express Inbound ATMU Registers," for more information.

Base address register 0 at offset 0x10 is a special fixed 16M or general purpose window that is used for inbound configuration or memory accesses. This window is called the PCI Express configuration and status register base address register (PEXCSRBAR) or window 0. Note that PEXCSRBAR cannot be updated through the inbound ATMU registers. The PEXCSRBAR is shown in Figure 20-77.

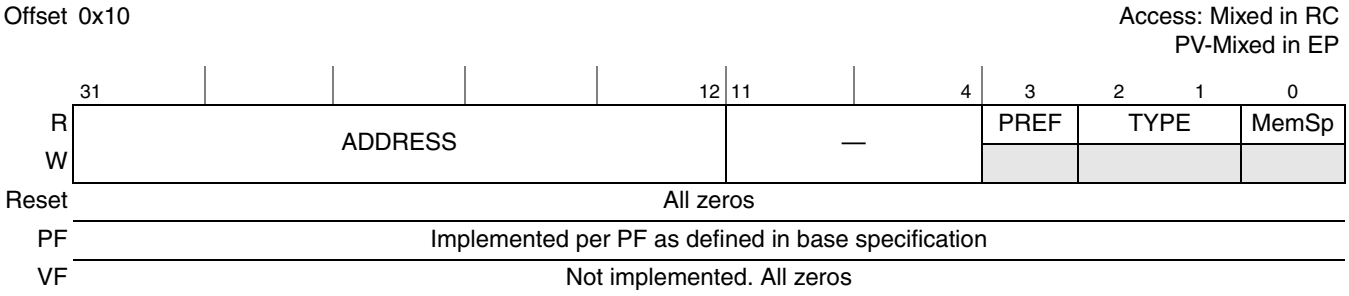


Figure 20-77. PCI Express Base Address Register 0 (PEXCSRBAR)

Table 20-73 describes the PCI Express configuration and status register base address register.

Table 20-73. PEXCSRBAR Field Descriptions

Bits	Name	Description
31–12	ADDRESS	Indicates the base address of the inbound configuration window or the general purpose inbound memory window. If used as an inbound configuration window, the base address must be aligned to the 16-Mbyte CCSR space and the the size of the window in the inbound window attributes register (PEXIWAR0) must be 16-Mbytes. If the window is used as a general purpose inbound memory window, then the number of upper bits that the device allows to be writable is selected through the inbound window size in the inbound window attributes register (PEXIWAR0).
11–4	—	Reserved
3	PREF	Prefetchable
2–1	TYPE	Type. 00 Locate anywhere in 32-bit address space.
0	MemSp	Memory space indicator

Base address register 1 at offset 0x14 is used to define the inbound memory window in the 32-bit memory space. The 32-bit memory BAR is shown in Figure 20-78.

Offset 0x14 (EP-mode only)

Access: Mixed in RC
PV-Mixed in EP

	31											12	11				4	3	2	1	0
R	ADDRESS												—					PREF	TYPE		MemSp
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	1	0	0	0
PF	Implemented per PF as defined in base specification																				
VF	Not implemented. All zeros.																				

Figure 20-78. 32-Bit Memory Base Address Register (BAR1)

Table 20-74 describes the PCI Express 32-bit memory BAR fields.

Table 20-74. 32-Bit Memory Base Address Register (BAR1) Field Descriptions

Bits	Name	Description
31–12	ADDRESS	Indicates the base address where the inbound memory window begins. The number of upper bits that the device allows to be writable is selected through the inbound window size in the inbound window attributes register (PEXIWAR1).
11–4	—	Reserved. The device allows a 4 Kbyte window minimum.
3	PREF	Prefetchable. This bit is determined by PEXIWAR1[PF].
2–1	TYPE	Type. 00 Locate anywhere in 32-bit address space.
0	MemSp	Memory space indicator.

Base address register 2 at offset 0x18 and base address register 4 at offset 0x20 are used to define the lower portion of the 64-bit inbound memory windows. The 64-bit low memory BARs are shown in Figure 20-79.

Offset 0x18 (EP-mode only)
0x20 (EP-mode only)

Access: Mixed in RC
PV-Mixed in EP

	31											12	11				4	3	2	1	0
R	ADDRESS												—					PREF	TYPE		MemSp
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	1	1	0	0
PF	Implemented per PF as defined in base specification																				
VF	Not implemented. All zeros																				

Figure 20-79. 64-Bit Low Memory Base Address Register

Table 20-75 describes the PCI Express 64-bit low memory BAR fields.

Table 20-75. 64-Bit Low Memory Base Address Register Field Descriptions

Bits	Name	Description
31–12	ADDRESS	Indicates the lower portion of the base address where the inbound memory window begins. The number of bits that the device allows to be writable is selected through the inbound window size in the inbound window attributes registers (PEXIWAR2 for offset 0x18 and PEXIWAR3 for offset 0x20).
11–4	—	Reserved. The device allows a 4 Kbyte window minimum.
3	PREF	Prefetchable. This bit is determined by PEXIWAR*n*[2].

Table 20-75. 64-Bit Low Memory Base Address Register Field Descriptions (continued)

Bits	Name	Description
2–1	TYPE	Type. 0b10 Locate anywhere in 64-bit address space.
0	MemSp	Memory space indicator

Base address register 3 at offset 0x1C and base address register 5 at offset 0x24 are used to define the upper portion of the 64-bit inbound memory windows. The 64-bit high memory BARs are shown in Figure 20-80.

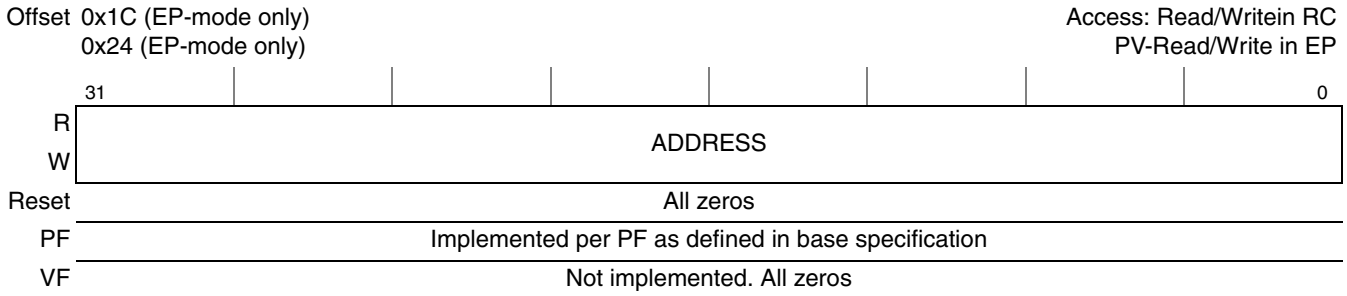


Figure 20-80. 64-Bit High Memory Base Address Register

Table 20-76 describes the PCI Express 64-bit low memory BAR fields.

Table 20-76. Bit Setting for 64-Bit High Memory Base Address Register

Bits	Name	Description
31–0	ADDRESS	Indicates the upper portion of the base address where the inbound memory window begins. The number of bits that the device allows to be writable is selected through the inbound window size in the inbound window attributes registers (PEXIWAR2 for offset 0x1C and PEXIWAR3 for offset 0x24). If no access to local memory is to be permitted by external requestors, then all bits are programmed.

20.3.9.2.2 PCI Express Subsystem Vendor ID Register (EP-Mode Only)—0x2C

The PCI Express subsystem vendor ID register is used to identify the subsystem.

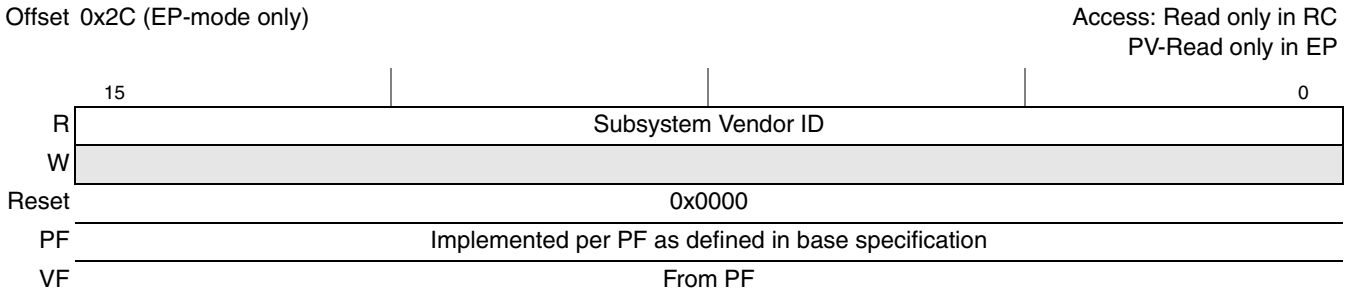


Figure 20-81. PCI Express Subsystem Vendor ID Register

Table 20-77. PCI Express Subsystem Vendor ID Register Field Description

Bits	Name	Description
15–0	Subsystem Vendor ID	

20.3.9.2.3 PCI Express Subsystem ID Register (EP-Mode Only)—0x2E

The PCI Express subsystem ID register is used to identify the subsystem.

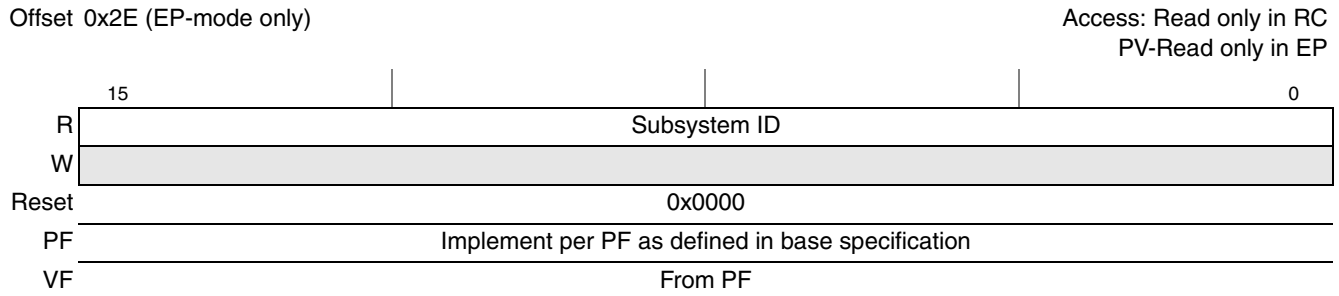


Figure 20-82. PCI Express Subsystem ID Register

Table 20-78. PCI Express Subsystem ID Register Field Description

Bits	Name	Description
15–0	Subsystem ID	

20.3.9.2.4 PCI Express Expansion ROM Base Address Register—0x30

The register at offset 0x30 has a different function depending on the value in PEX_CSR1[CS2]. When PEX_CSR1[CS2] = 0, the register functions as the PCI Express expansion ROM base address register and has the format as shown in Figure 20-83.

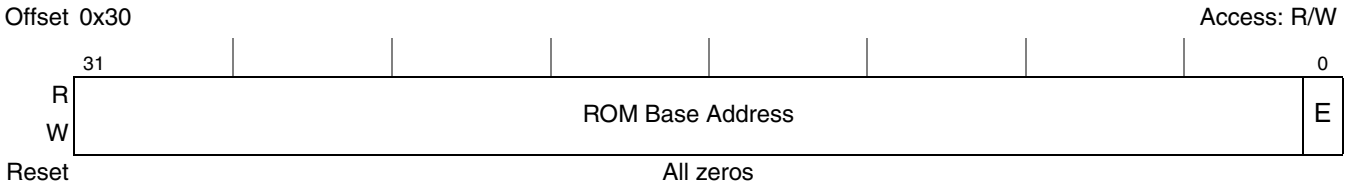


Figure 20-83. PCI Express Expansion ROM Base Address Register when PEX_CSR[CS2]=0

Table 20-79. PCI Express Expansion ROM Base Address Register Field Description when PEX_CSR[CS2]=0

Bits	Name	Description
31–4	ROM Base Address	Specifies bits 31:20 of the non-prefetchable expansion ROM space start address. Typically used for specifying memory-mapped I/O space. The default size is 16M.
3–1	—	Reserved
0	Address decode enable	Address decode enable

20.3.9.2.5 PCI Express Expansion ROM BAR Mask Register—0x30

The register at offset 0x30 has a different function depending on the value in PEX_CSR1[CS2]. When PEX_CSR1[CS2] = 1, the register functions as the PCI Express expansion ROM BAR mask register and has the format as shown in Figure 20-84.

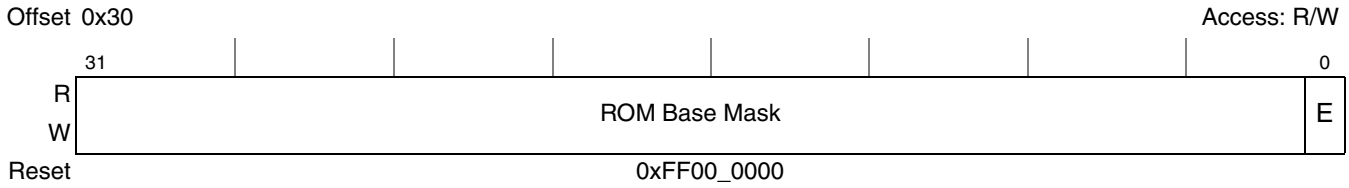


Figure 20-84. PCI Express Expansion ROM BAR Mask Register when PEX_CSR1[CS2]=1

Table 20-80. PCI Express Expansion ROM BAR Mask Register Field Description when PEX_CSR[CS2]=1

Bits	Name	Description
31–4	ROM Mask	The default mask is 0xFF00000 which is 16 M.
3–1	—	Reserved
0	Address decode enable	Address decode enable

20.3.9.2.6 Capabilities Pointer Register—0x34

The capabilities pointer identifies additional functionality supported by the device.

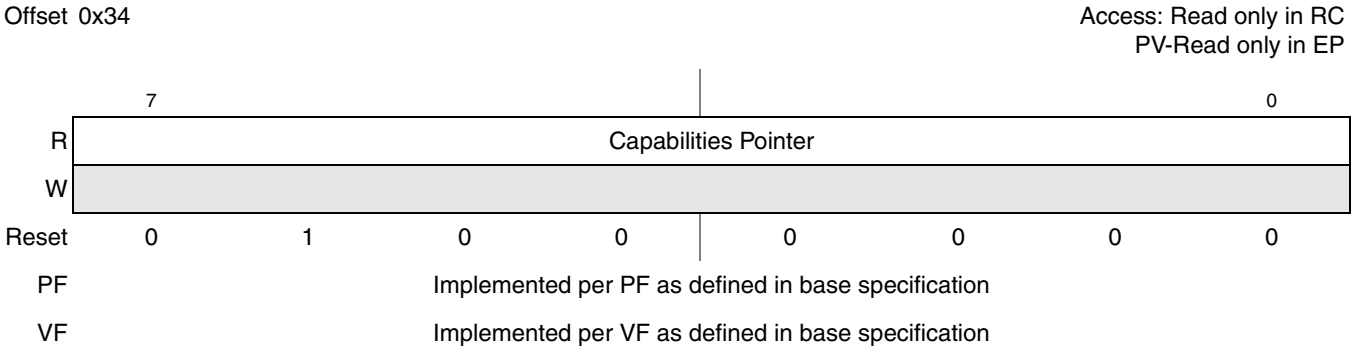


Figure 20-85. Capabilities Pointer Register

Table 20-81. Capabilities Pointer Register Field Description

Bits	Name	Description
7–0	Capabilities Pointer	The capabilities pointer provides the offset for additional PCI-compatible registers above the common 64-byte header. Refer to Section 20.3.10, "PCI Compatible Device-Specific Configuration Space," for more information.

20.3.9.2.7 PCI Express Interrupt Line Register (EP-Mode Only)—0x3C

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.

Offset 0x3C (EP-mode only)

Access: Read/Write in RC
PV-Read/Write in EP

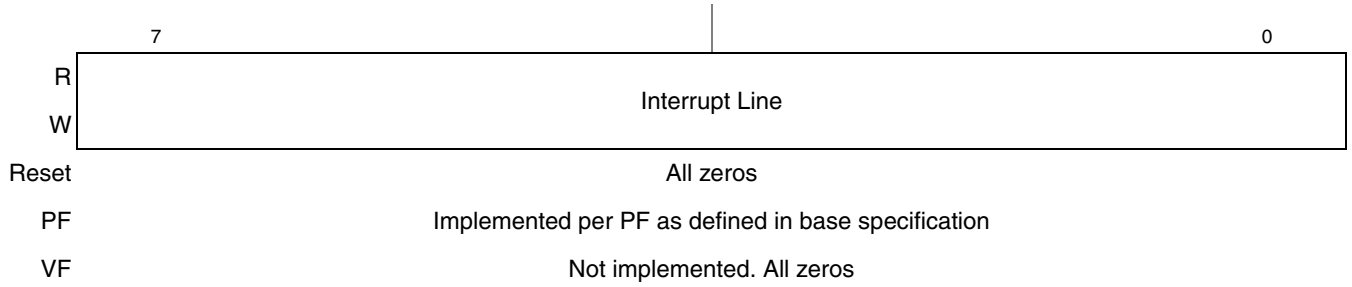


Figure 20-86. PCI Express Interrupt Line Register

Table 20-82. PCI Express Interrupt Line Register Field Description

Bits	Name	Description
7–0	Interrupt Line	Used to communicate interrupt line routing information.

20.3.9.2.8 PCI Express Interrupt Pin Register—0x3D

The interrupt pin register identifies the legacy interrupt (INTx) messages the device (or function) uses.

Offset 0x3D

Access: Read only in RC
PV-Read only in EP

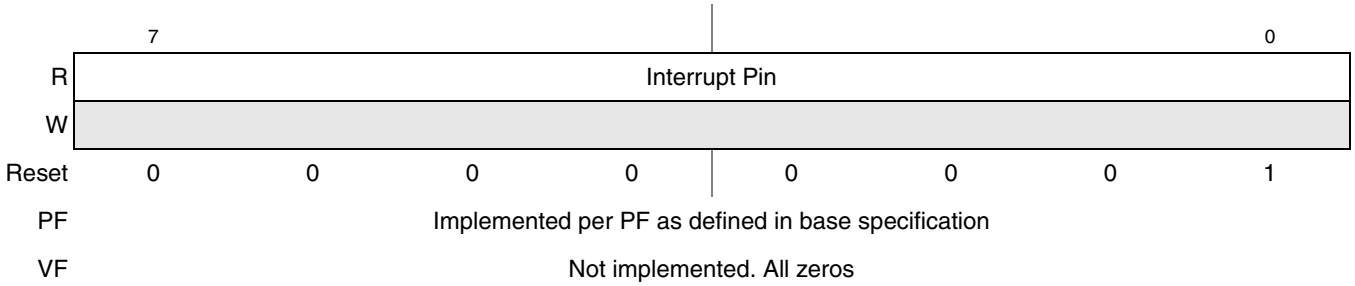


Figure 20-87. PCI Express Interrupt Pin Register

Table 20-83. PCI Express Interrupt Pin Register Field Description

Bits	Name	Description
7–0	Interrupt pin	Legacy INTx message used by this device. 0x00 This device does not use legacy interrupt (INTx) messages. 0x01 INTA 0x02 INTB 0x03 INTC 0x04 INTD all others Reserved.

20.3.9.2.9 PCI Express Minimum Grant Register (EP-Mode Only)—0x3E

This register does not apply to PCI Express. It is present for legacy purposes.

Offset 0x3E (EP-mode only) Access: Read only in RC
PV-Read only in EP

	7	0
R	MIN_GNT	
W		
Reset	All zeros	
PF	Implemented per PF as defined in base specification	
VF	Not implemented. All zeros	

Figure 20-88. PCI Express Maximum Grant Register (MAX_GNT)

Table 20-84. PCI Express Maximum Grant Register Field Description

Bits	Name	Description
7–0	MIN_GNT	Does not apply for PCI Express.

20.3.9.2.10 PCI Express Maximum Latency Register (EP-Mode Only)—0x3F

This register does not apply to PCI Express. It is present for legacy purposes.

Offset 0x3F (EP-mode only) Access: Read only in RC
PV-Read only in EP

	7	0
R	MAX_LAT	
W		
Reset	All zeros	
PF	Implemented per PF as defined in base specification	
VF	Not implemented. All zeros	

Figure 20-89. PCI Express Maximum Latency Register (MAX_LAT)

Table 20-85. PCI Express Maximum Latency Register Field Description

Bits	Name	Description
7–0	MAX_LAT	Does not apply for PCI Express.

20.3.9.3 Type 1 Configuration Header

The type 1 header is shown in Figure 20-90.

Reserved

				Address Offset (Hex)
Device ID		Vendor ID		00
Status		Command		04
Class Code			Revision ID	08
BIST	Header Type	Latency Timer	Cache Line Size	0C
Base Address Register 0				10
				14
Secondary Latency Timer	Subordinate Bus Number	Secondary Bus Number	Primary Bus Number	18
Secondary Status		I/O Limit	I/O Base	1C
Memory Limit		Memory Base		20
Prefetchable Memory Limit		Prefetchable Memory Base		24
Prefetchable Base Upper 32 Bits				28
Prefetchable Limit Upper 32 Bits				2C
I/O Limit Upper 16 Bits		I/O Base Upper 16 Bits		30
			Capabilities Pointer	34
Expansion ROM Base Address				38
Bridge Control		Interrupt Pin	Interrupt Line	3C

Figure 20-90. PCI Express PCI-Compatible Configuration Header—Type 1

Section 20.3.9.1, "Common PCI Compatible Configuration Header Registers," describes the registers in the first 16 bytes of the header. This section describes the registers that are unique to the type 1 header beginning at offset 0x10.

20.3.9.3.1 PCI Express Base Address Register 0—0x10

Base address register 0 at offset 0x10 is a special fixed 16-Mbyte window that is used for inbound configuration accesses. This window is called the PCI Express configuration and status register base address register (PEXCSRBAR). Note that PEXCSRBAR cannot be updated through the inbound ATMU registers. The PEXCSRBAR is shown in Figure 20-91.

Offset 0x10 Access: Mixed

	31	20	19	12	11	4	3	2	1	0
R	ADDRESS				—		PREF	TYPE		MemSp
W										
Reset	All zeros									

Figure 20-91. PCI Express Base Address Register 0 (PEXCSRBAR)

Table 20-86 describes the PCI Express configuration and status register base address register.

Table 20-86. PEXCSRBAR Field Descriptions

Bits	Name	Description
31–12	ADDRESS	Indicates the base address that the inbound configuration window occupies. This window is fixed at 16 Mbyte.
11–4	—	Reserved
3	PREF	Prefetchable
2–1	TYPE	Type. 00 Locate anywhere in 32-bit address space.
0	MemSp	Memory space indicator

20.3.9.3.2 PCI Express Primary Bus Number Register—Offset 0x18

The primary bus number register is shown in Figure 20-92.

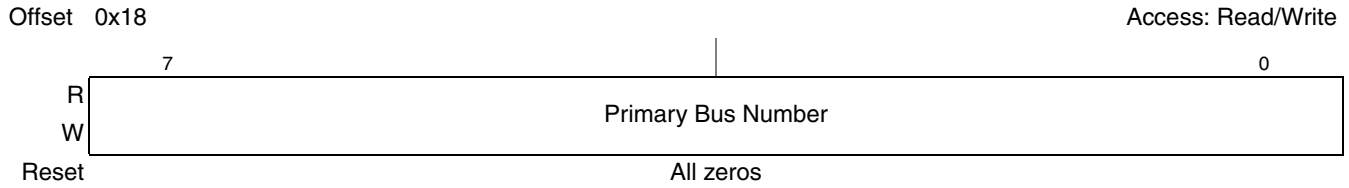


Figure 20-92. PCI Express Primary Bus Number Register

Table 20-87 describes the primary bus number register fields.

Table 20-87. PCI Express Primary Bus Number Register Field Description

Bits	Name	Description
7–0	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.

20.3.9.3.3 PCI Express Secondary Bus Number Register—Offset 0x19

The secondary bus number register is shown in Figure 20-93.

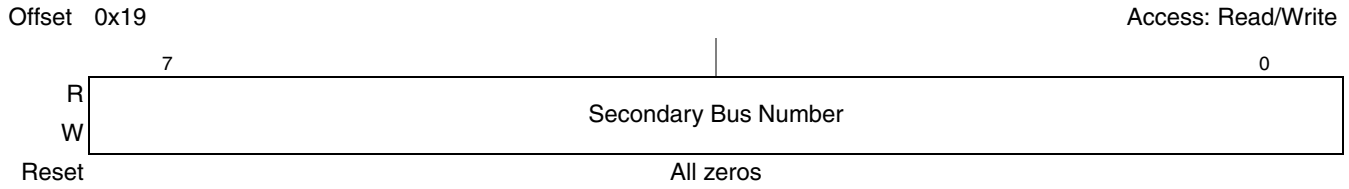


Figure 20-93. PCI Express Secondary Bus Number Register

Table 20-88 describes the secondary bus number register fields.

Table 20-88. PCI Express Secondary Bus Number Register Field Description

Bits	Name	Description
7–0	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.

20.3.9.3.4 PCI Express Subordinate Bus Number Register—Offset 0x1A

The subordinate bus number register is shown in Figure 20-94.

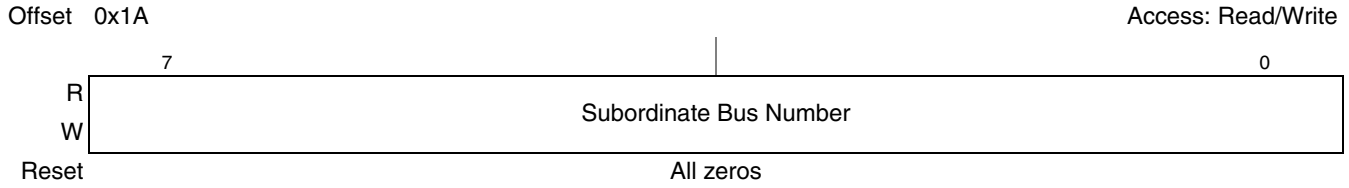


Figure 20-94. PCI Express Subordinate Bus Number Register

Table 20-89 describes the subordinate bus number register fields.

Table 20-89. PCI Express Subordinate Bus Number Register Field Description

Bits	Name	Description
7–0	Subordinate Bus Number	Highest bus number that is on the downstream interface.

20.3.9.3.5 PCI Express Secondary Latency Timer Register—0x1B

The secondary latency timer register does not apply to PCI Express. It must be read-only and return all zeros when read.

20.3.9.3.6 PCI Express I/O Base Register—0x1C

Note that this device does not support inbound I/O transactions. The I/O base register is shown in Figure 20-95.

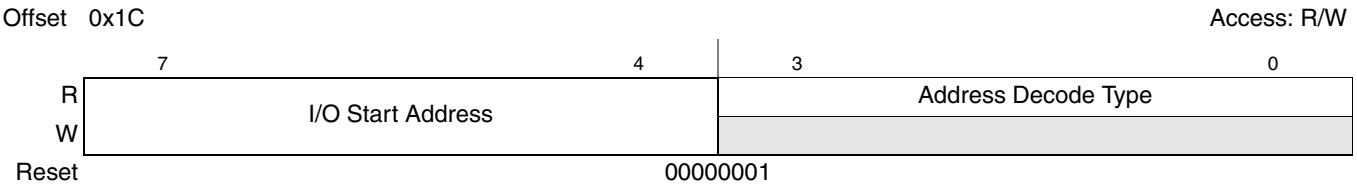


Figure 20-95. PCI Express I/O Base Register

Table 20-90 describes the I/O base register fields.

Table 20-90. PCI Express I/O Base Register Field Description

Bits	Name	Description
7–4	I/O Start Address	Specifies bits 15:12 of the I/O space start address
3–0	Address Decode Type	Specifies the number of I/O address bits. 0x00 16-bit I/O address decode 0x01 32-bit I/O address decode All other settings reserved.

20.3.9.3.7 PCI Express I/O Limit Register—0x1D

Note that this device does not support inbound I/O transactions. The I/O limit register is shown in Figure 20-96.

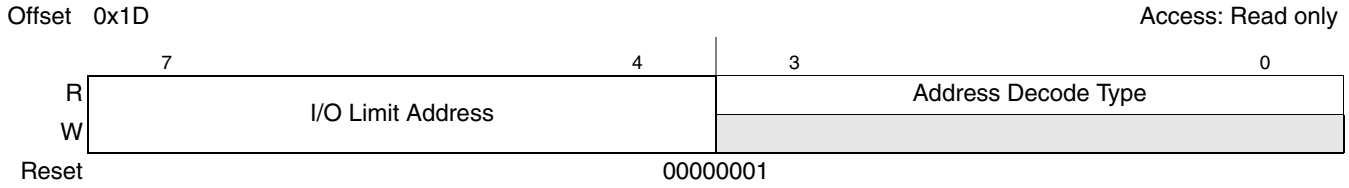


Figure 20-96. PCI Express I/O Limit Register

Table 20-91 describes the I/O limit register fields.

Table 20-91. PCI Express I/O Limit Register Field Description

Bits	Name	Description
7–4	I/O Limit Address	Specifies bits 15:12 of the I/O space ending address
3–0	Address Decode Type	Specifies the number of I/O address bits. 0x00 16-bit I/O address decode 0x01 32-bit I/O address decode All other settings reserved.

20.3.9.3.8 PCI Express Secondary Status Register—0x1E

The PCI Express secondary status register is shown in Figure 20-97. Note that the errors in this register may be masked by corresponding bits in the secondary status interrupt mask register (PEX_SS_INTR_MASK) and that by default all of the errors are masked.

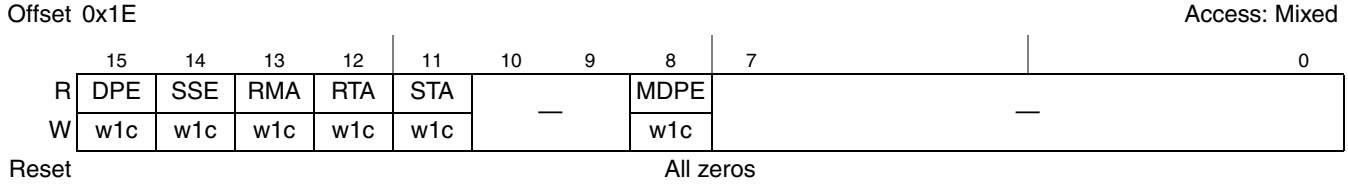


Figure 20-97. PCI Express Secondary Status Register

Table 20-92 describes the PCI Express secondary status register fields.

Table 20-92. PCI Express Secondary Status Register Field Description

Bits	Name	Description
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
8	MDPE	Master data parity error. This bit is set when the parity error response bit is set and the secondary side requestor 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

20.3.9.3.9 PCI Express Memory Base Register—0x20

The memory base register is shown in Figure 20-98.

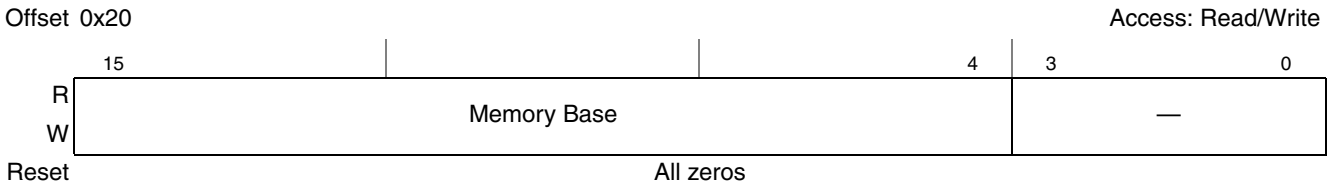


Figure 20-98. PCI Express Memory Base Register

Table 20-93 describes the memory base register fields.

Table 20-93. PCI Express Memory Base Register Field Description

Bits	Name	Description
15–4	Memory Base	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

20.3.9.3.10 PCI Express Memory Limit Register—0x22

The memory limit register is shown in Figure 20-99.

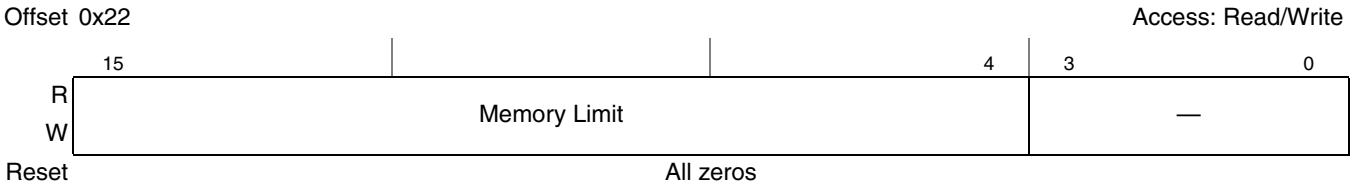


Figure 20-99. PCI Express Memory Limit Register

Table 20-94 describes the memory base register fields.

Table 20-94. PCI Express Memory Limit Register Field Description

Bits	Name	Description
15–4	Memory Limit	Specifies bits 31:20 of the non-prefetchable memory space ending 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 unsupported request response.
3–0	—	Reserved

20.3.9.3.11 PCI Express Prefetchable Memory Base Register—0x24

The prefetchable memory base register is shown in Figure 20-100.

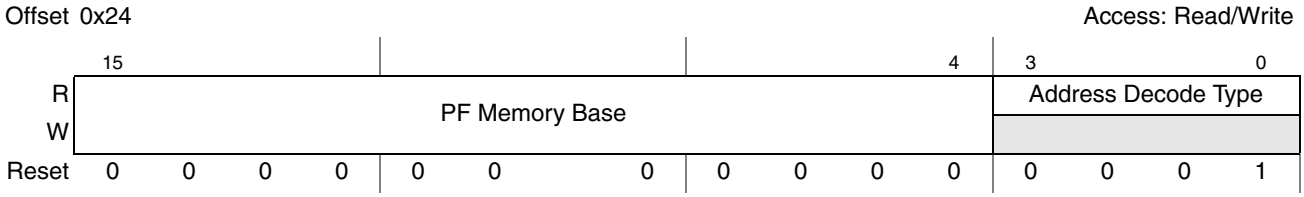


Figure 20-100. PCI Express Prefetchable Memory Base Register

Table 20-95 describes the prefetchable memory base register fields.

Table 20-95. PCI Express Prefetchable Memory Base Register Field Description

Bits	Name	Description
15–4	PF Memory Base	Specifies bits 31:20 of the prefetchable memory space start address.
3–0	Address Decode Type	Specifies the number of prefetchable memory address bits. 0x00 32-bit memory address decode 0x01 64-bit memory address decode All other settings reserved.

20.3.9.3.12 PCI Express Prefetchable Memory Limit Register—0x26

The prefetchable memory limit register is shown in Figure 20-101.

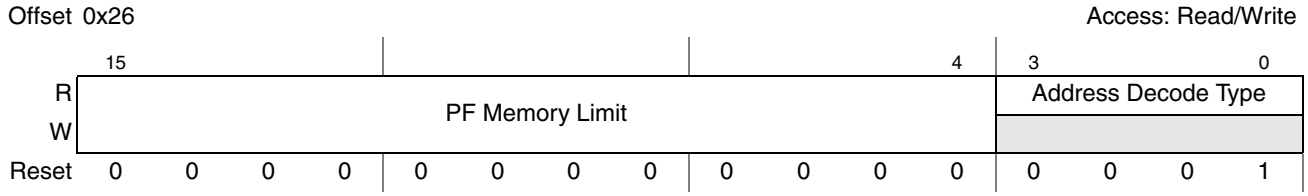


Figure 20-101. PCI Express Prefetchable Memory Limit Register

Table 20-96 describes the prefetchable memory limit register fields.

Table 20-96. PCI Express Prefetchable Memory Limit Register Field Description

Bits	Name	Description
15–4	PF Memory Limit	Specifies bits 31:20 of the prefetchable memory space ending address.
3–0	Address Decode Type	Specifies the number of prefetchable memory address bits. 0x00 32-bit memory address decode 0x01 64-bit memory address decode All other settings reserved.

20.3.9.3.13 PCI Express Prefetchable Base Upper 32 Bits Register—0x28

The PCI Express prefetchable memory base upper 32 bits register is shown in Figure 20-102.

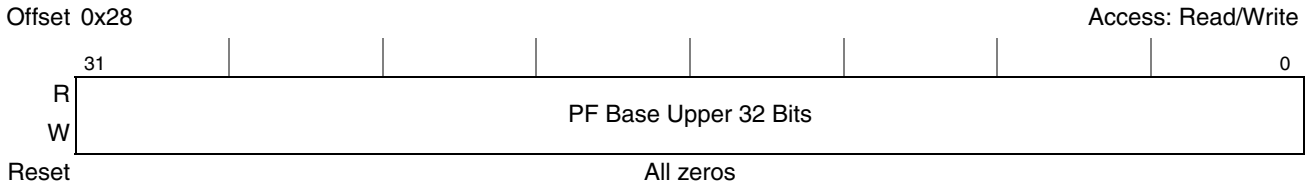


Figure 20-102. PCI Express Prefetchable Base Upper 32 Bits Register

Table 20-97 describes the PCI Express prefetchable memory base upper 32 bits register fields.

Table 20-97. PCI Express Prefetchable Base Upper 32 Bits Register

Bits	Name	Description
31–0	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.

20.3.9.3.14 PCI Express Prefetchable Limit Upper 32 Bits Register—0x2C

The PCI Express prefetchable memory base upper 32 bits register is shown in Figure 20-103.

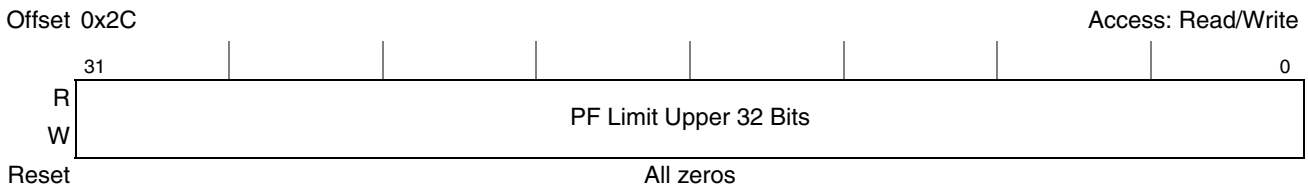


Figure 20-103. PCI Express Prefetchable Limit Upper 32 Bits Register

Table 20-98 describes the PCI Express prefetchable memory limit upper 32 bits register fields.

Table 20-98. PCI Express Prefetchable Limit Upper 32 Bits Register

Bits	Name	Description
31–0	PF Limit Upper 32 Bits	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.

20.3.9.3.15 PCI Express I/O Base Upper 16 Bits Register—0x30

Note that this device does not support inbound I/O transactions. The I/O base upper 16 bits register is shown in Figure 20-104.

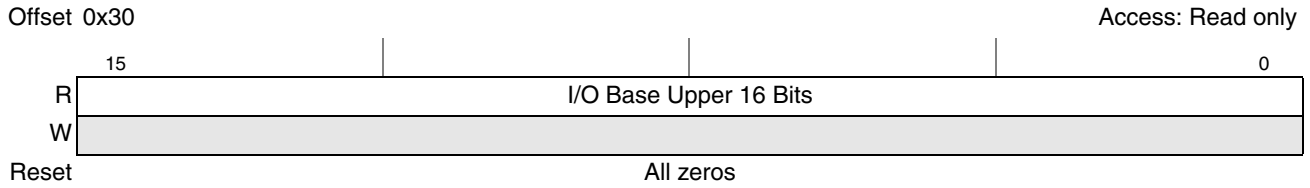


Figure 20-104. PCI Express I/O Base Upper 16 Bits Register

Table 20-99 describes the I/O base upper 16 bits register fields.

Table 20-99. PCI Express I/O Base Upper 16 Bits Register Field Description

Bits	Name	Description
15–0	I/O Base Upper 16 Bits	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.

20.3.9.3.16 PCI Express I/O Limit Upper 16 Bits Register—0x32

Note that this device does not support inbound I/O transactions. The I/O limit upper 16 bits register is shown in Figure 20-105.

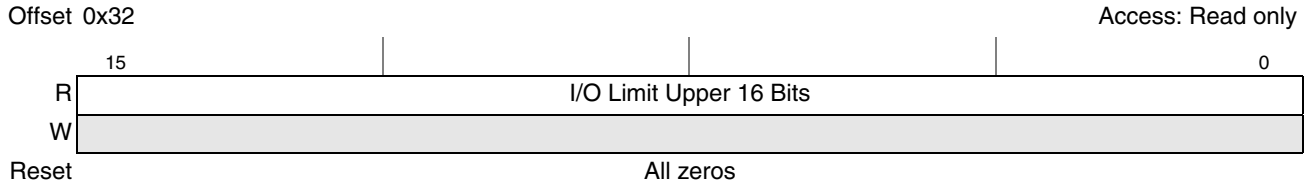


Figure 20-105. PCI Express I/O Limit Upper 16 Bits Register

Table 20-100 describes the I/O limit upper 16 bits register fields.

Table 20-100. PCI Express I/O Limit Upper 16 Bits Register Field Description

Bits	Name	Description
15–0	I/O Limit Upper 16 Bits	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.

20.3.9.3.17 Capabilities Pointer Register—0x34

The capabilities pointer identifies additional functionality supported by the device.

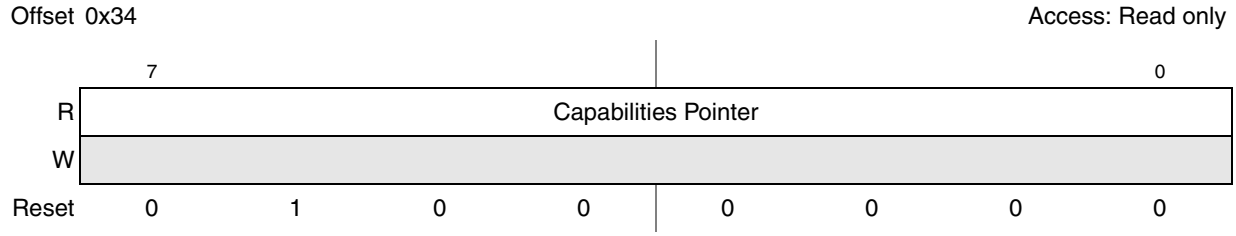


Figure 20-106. Capabilities Pointer Register

Table 20-101. Capabilities Pointer Register Field Description

Bits	Name	Description
7–0	Capabilities Pointer	The capabilities pointer provides the offset for additional PCI-compatible registers above the common 64-byte header. Refer to Section 20.3.10, "PCI Compatible Device-Specific Configuration Space," for more information.

20.3.9.3.18 PCI Express Expansion ROM Base Address Register—0x38

The expansion ROM base address register is shown in Figure 20-107.

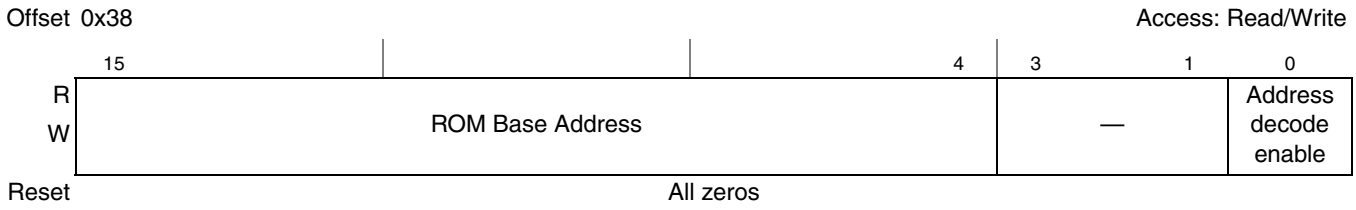


Figure 20-107. PCI Express Expansion ROM Base Address Register

Table 20-102 describes the expansion ROM base address register fields.

Table 20-102. PCI Express Expansion ROM Base Address Register Field Description

Bits	Name	Description
15–4	ROM Base Address	Specifies bits 31:20 of the non-prefetchable expansion ROM space start address. Typically used for specifying memory-mapped I/O space. The default size is 16M.
3–0	—	Reserved
0	Address decode enable	Address Decode enable

20.3.9.3.19 PCI Express Interrupt Line Register—0x3C

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.

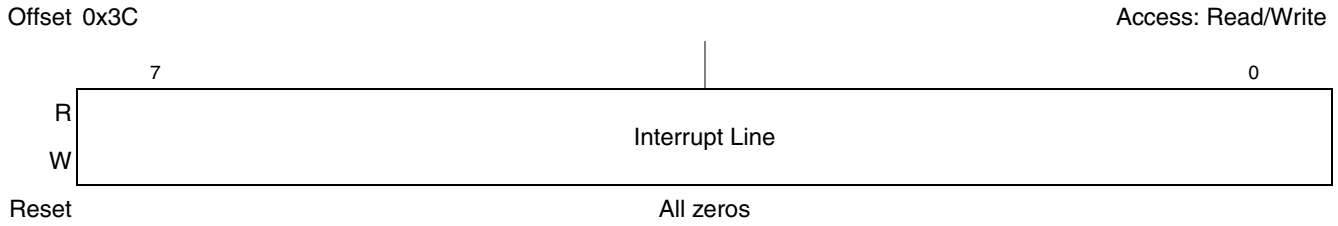


Figure 20-108. PCI Express Interrupt Line Register

Table 20-103. PCI Express Interrupt Line Register Field Description

Bits	Name	Description
7–0	Interrupt Line	Used to communicate interrupt line routing information.

20.3.9.3.20 PCI Express Interrupt Pin Register—0x3D

The interrupt pin register identifies the legacy interrupt (INTx) messages the device (or function) uses.

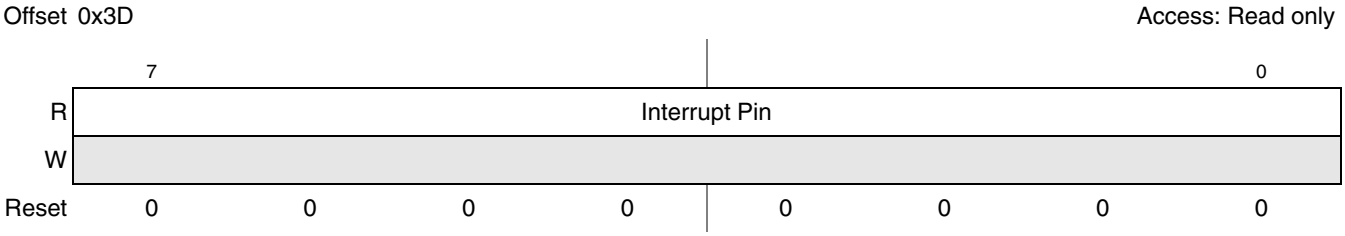


Figure 20-109. PCI Express Interrupt Pin Register

Table 20-104. PCI Express Interrupt Pin Register Field Description

Bits	Name	Description
7–0	Interrupt pin	Legacy INTx message used by this device. 0x00 This device does not use legacy interrupt (INTx) messages. 0x01 INTA 0x02 INTB 0x03 INTC 0x04 INTD all others Reserved.

20.3.9.3.21 PCI Express Bridge Control Register—0x3E

The PCI Express bridge control register is shown in Figure 20-110.

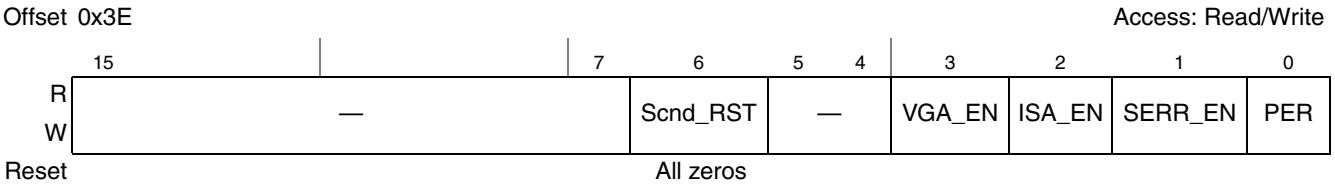


Figure 20-110. PCI Express Bridge Control Register

Table 20-105 describes the PCI Express bridge control register fields.

Table 20-105. PCI Express Bridge Control Register Field Description

Bits	Name	Description
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.

20.3.10 PCI Compatible Device-Specific Configuration Space

The PCI compatible device-specific configuration space is a PCI compatible configuration space from 0x40 to 0xFF (just after the 64-byte PCI-compatible configuration header).

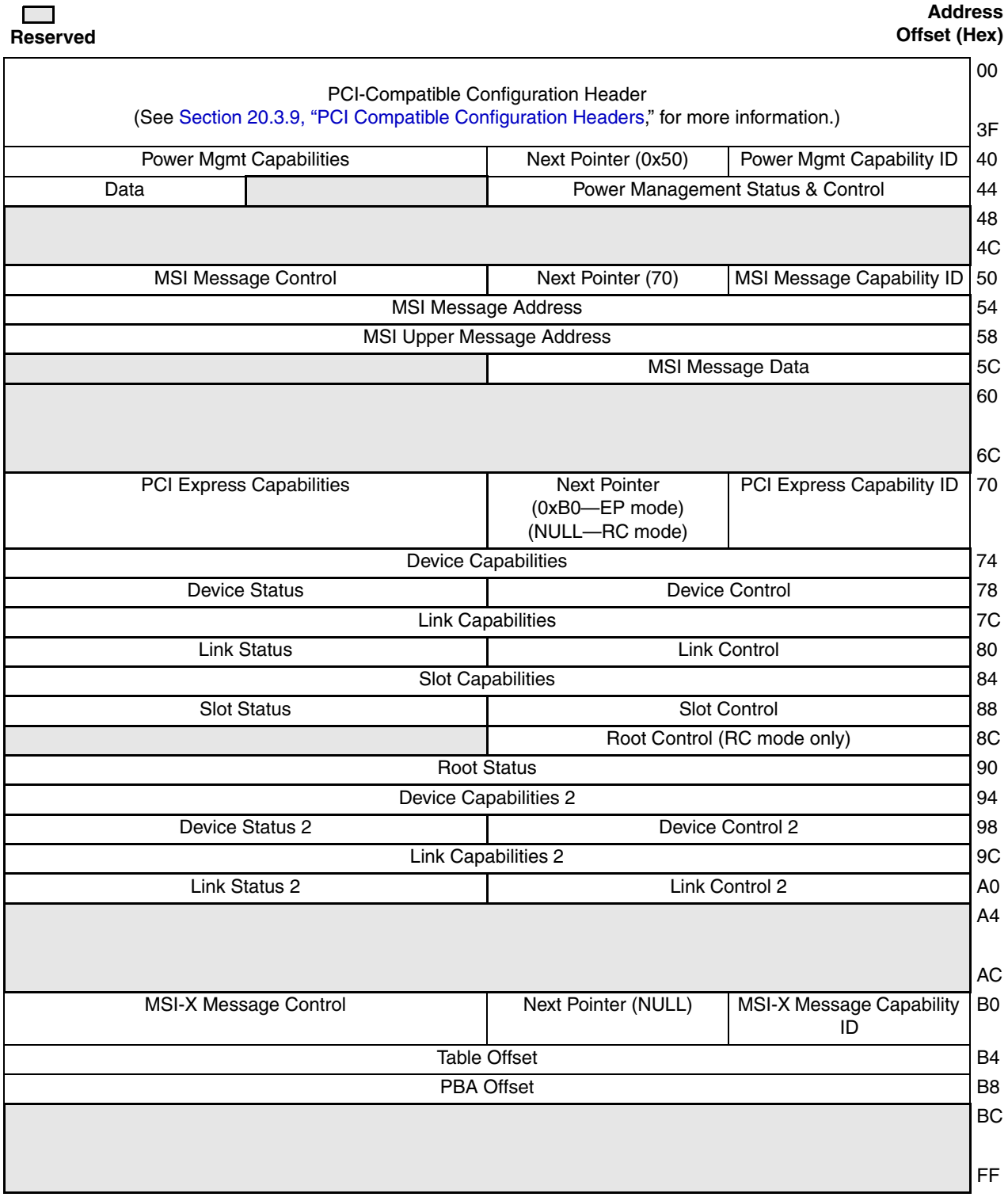


Figure 20-111. PCI Compatible Device-Specific Configuration Space (SR-IOV)

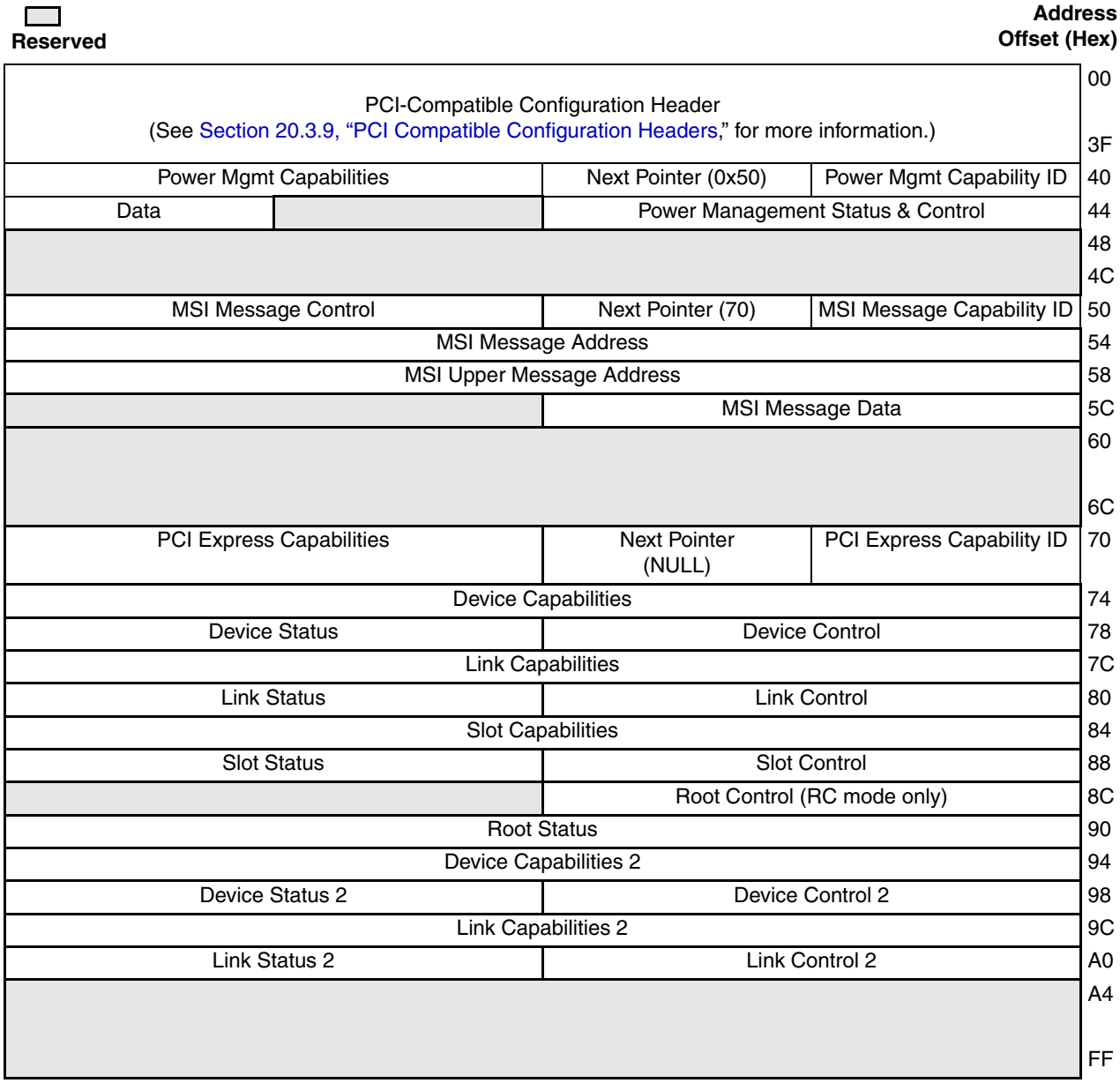


Figure 20-112. PCI Compatible Device-Specific Configuration Space (non-SR-IOV)

20.3.10.1 PCI Express Power Management Capability ID Register—0x40

The PCI Express power management capability ID register is shown in Figure 20-113.

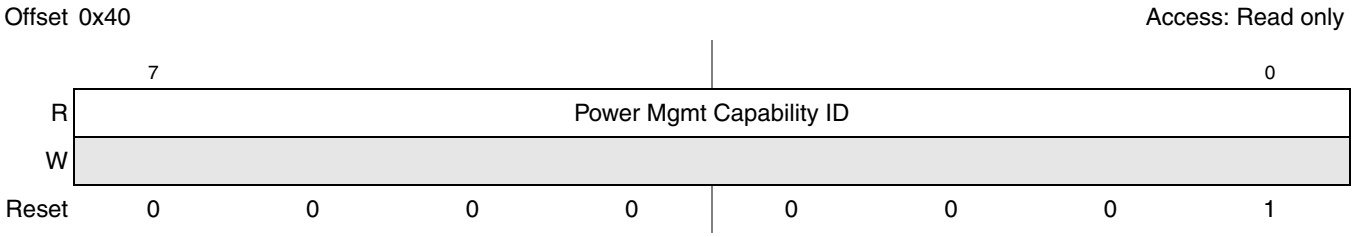


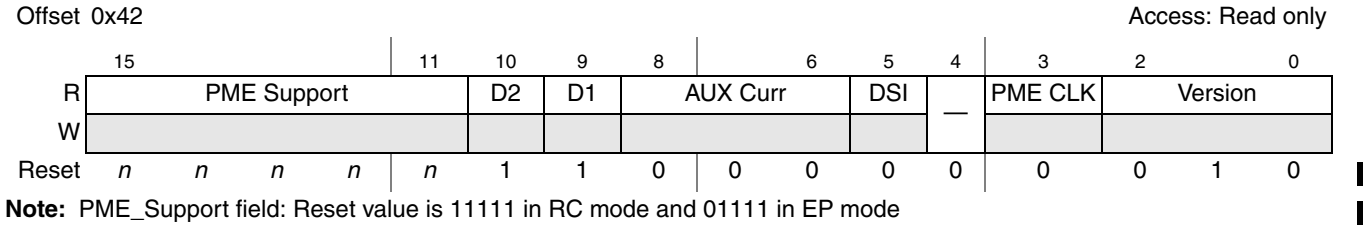
Figure 20-113. PCI Express Power Management Capability ID Register

Table 20-106. PCI Express Power Management Capability ID Register Field Description

Bits	Name	Description
7–0	Power Mgmt Capability ID	Power Management = 0x01

20.3.10.2 PCI Express Power Management Capabilities Register—0x42

The PCI Express power management capabilities register is shown in Figure 20-114.



Note: PME_Support field: Reset value is 11111 in RC mode and 01111 in EP mode

Figure 20-114. PCI Express Power Management Capabilities Register

Table 20-107. PCI Express Power Management Capabilities Register Field Description

Bits	Name	Description
15–11	PME Support	Indicates the power states that this device supports
10	D2	D2 Support
9	D1	D1 Support
8–6	AUX Curr	AUX Current
5	DSI	Device Specific Initialization
4	—	Reserved
3	PME CLK	Does not apply to PCI Express.
2–0	Version	Set to 0x2 for this version of the specification.

20.3.10.3 PCI Express Power Management Status and Control Register—0x44

The PCI Express power management status and control register is shown in Figure 20-115.

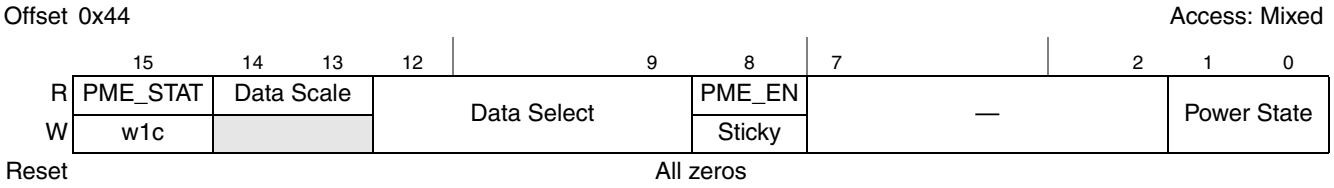


Figure 20-115. PCI Express Power Management Status and Control Register

Table 20-108. PCI Express Status and Control Register Field Description

Bits	Name	Description
15	PME_STAT	PME Status
14–13	Data Scale	Obtained directly from the PCI Express base specification.
12–9	Data Select	Obtained directly from the PCI Express base specification.
8	PME_EN	PME Enable
7–2	—	Reserved
1–0	Power State	Power state. Indicates the current power state of the function. 0x00 D0 0x01 D1 0x02 D2 0x03 D3

20.3.10.4 PCI Express Power Management Data Register—0x47

The PCI Express power management data register is shown in Figure 20-116.

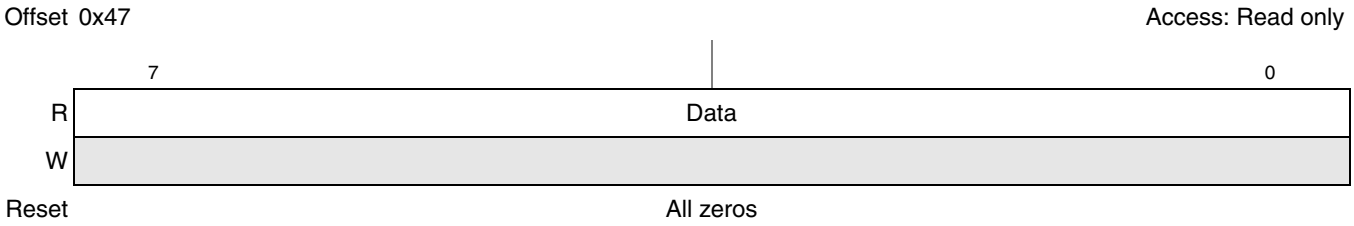


Figure 20-116. PCI Express Power Management Data Register

Table 20-109. PCI Express Power Management Data Register Field Description

Bits	Name	Description
7–0	Data	Obtained from the PCI Express base specification.

20.3.10.5 PCI Express MSI Message Capability ID Register (EP Mode Only)—0x50

The PCI Express MSI message capability ID register is shown in Figure 20-117.

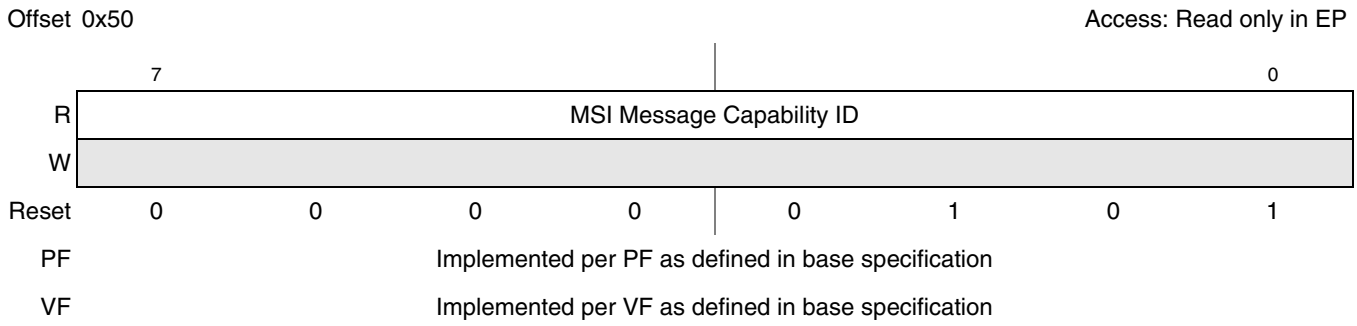


Figure 20-117. PCI Express MSI Message Capability ID Register

Table 20-110. PCI Express MSI Message Capability ID Register Field Description

Bits	Name	Description
7–0	MSI Message Capability ID	MSI Message = 0x05

20.3.10.6 PCI Express MSI Message Control Register (EP Mode Only)—0x52

The PCI Express MSI message control register is shown in Figure 20-118.

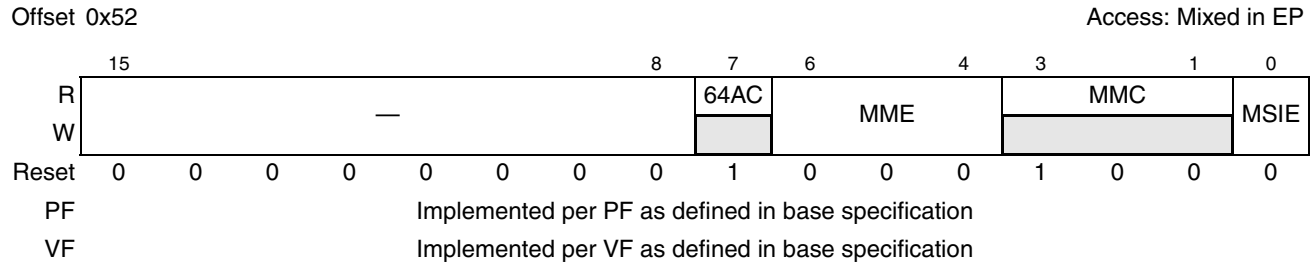


Figure 20-118. PCI Express MSI Message Control Register

Table 20-111. PCI Express MSI Message Control Register Field Description

Bits	Name	Description
15–8	—	Reserved
7	64AC	64-bit address capable.
6–4	MME	Multiple message enable.
3–1	MMC	Multiple message capable.
0	MSIE	MSI enable.

20.3.10.7 PCI Express MSI Message Address Register (EP Mode Only)—0x54

The PCI Express MSI message address register is shown in Figure 20-119.

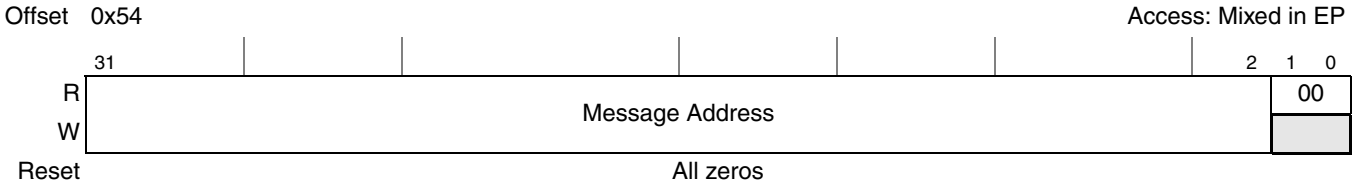


Figure 20-119. PCI Express MSI Message Address Register

Table 20-112. PCI Express MSI Message Address Register Field Description

Bits	Name	Description
31–2	Message Address	System-specified message address
1–0	00	Always returns 00 on reads; write operations have no effect.

20.3.10.8 PCI Express MSI Message Upper Address Register (EP Mode Only)—0x58

The PCI Express MSI message upper address register is shown in Figure 20-120.

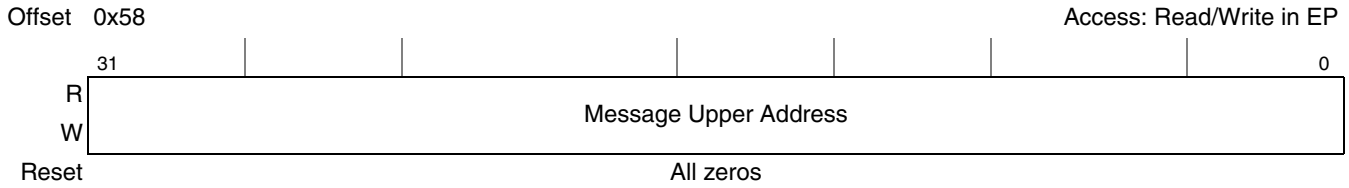


Figure 20-120. PCI Express MSI Message Upper Address Register

Table 20-113. PCI Express MSI Message Upper Address Register Field Description

Bits	Name	Description
31–0	Message Upper Address	System-specified message upper address

20.3.10.9 PCI Express MSI Message Data Register (EP Mode Only)—0x5C

The PCI Express MSI message data register is shown in Figure 20-121.

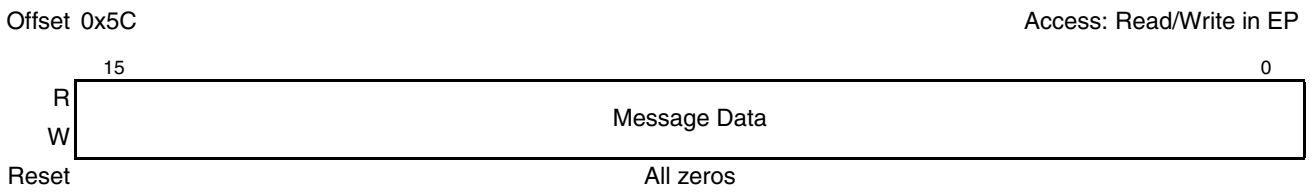


Figure 20-121. PCI Express MSI Message Data Register

Table 20-114. PCI Express MSI Message Data Register Field Description

Bits	Name	Description
15–0	Message Data	System-specified message.

20.3.10.10 PCI Express Capability ID Register—0x70

The PCI Express capability ID register is shown in Figure 20-122.

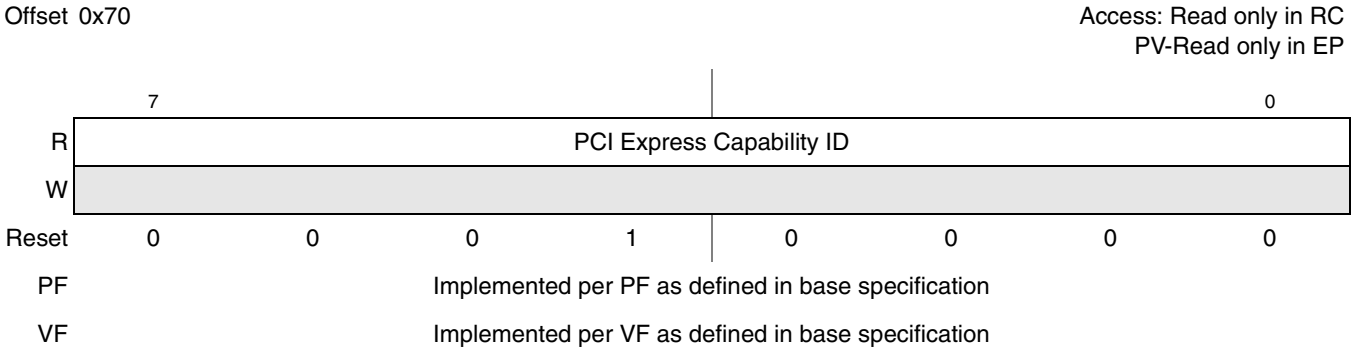


Figure 20-122. PCI Express Capability ID Register

Table 20-115. PCI Express Capability ID Register Field Description

Bits	Name	Description
7–0	PCI Express Capability ID	PCI Express = 0x10

20.3.10.11 PCI Express Capabilities Register—0x72

The PCI Express capabilities register is shown in Figure 20-123.

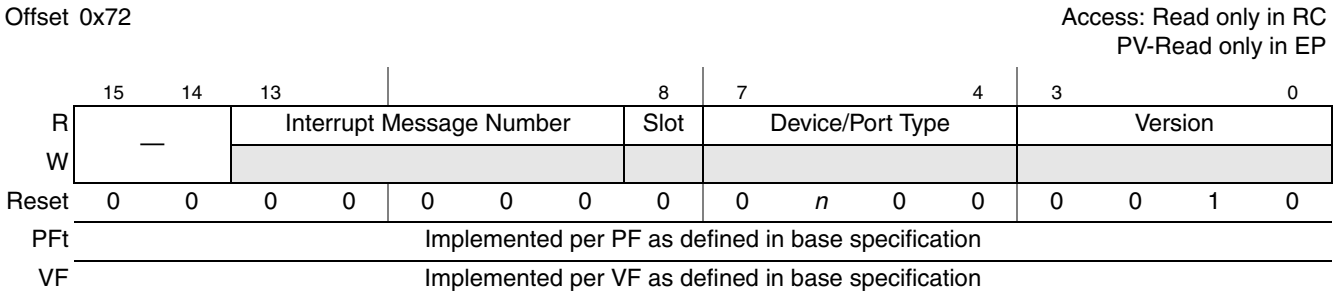


Figure 20-123. PCI Express Capabilities Register

Table 20-116. PCI Express Capabilities Register Field Description

Bits	Name	Description
15–14	—	Reserved
13–9	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	0100 (RC mode) 0000 (EP mode)
3–0	Capability Version	Indicates the defined PCI Express capability structure version number.

20.3.10.12 PCI Express Device Capabilities Register—0x74

The PCI Express device capabilities register is shown in Figure 20-124.

Offset 0x74 Access: Read only in RC; PV-Read only in EP

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	—			FLRC	CSPLS		CSPLV								—	
W																
Reset	0	0	0	1	0	0	1	1	1	1	1	1	1	1	0	0
PF	Implemented per PF as defined in base specification															
VF	0	0	0	1	Not implemented. Undefined											

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	—	PIP	AIP	ABP	EP_L1_LAT			EP_L0s_LAT			ET	PHAN_FCT		MAX_PL_SIZE_SUP		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
PF	Implemented per PF as defined in base specification															
VF	Implemented per VF as defined in base specification															

Figure 20-124. PCI Express Device Capabilities Register

Table 20-117. PCI Express Device Capabilities Register Field Description

Bits	Name	Description
31–29	—	Reserved
28	FLRC	Functiona level reset capability
27–26	CSPLS	Captured Slot Power Limit Scale
25–18	CSPLV	Captured Slot Power Limit Value
17–15	—	Reserved
14	PIP	Power Indicator Present
13	AIP	Attention Indicator Present
12	ABP	Attention Button Present
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	Maximum payload size supported. 001 = 256-bytes

20.3.10.13 PCI Express Device Control Register—0x78

The PCI Express device control register is shown in Figure 20-125.

Offset 0x78 Access: Read/write in RC
PV-Read only in EP

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	IFLR	MAX_READ_SIZE			NSE	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
PF	Implemented per PF as defined in base specification															
VF	0	From PF														

Figure 20-125. PCI Express Device Control Register

Table 20-118. PCI Express Device Control Register Field Description

Bits	Name	Description
15	IFLR	Initiate functional level reset.
14–12	MAX_READ_SIZE	Maximum read request size
11	NSE	No snoop enable
10	APE	AUX power PM enable
9	PFE	Phantom functions enable
8	ETE	Extended tag field enable
7–5	MAX_PAYLOAD_SIZE	Maximum payload size
4	RO	Relaxed ordering
3	URR	Unsupported request reporting
2	FER	Fatal error reporting
1	NFER	Non-fatal error reporting
0	CER	Correctable error reporting

20.3.10.14 PCI Express Device Status Register—0x7A

The PCI Express device status register is shown in Figure 20-126.

Offset 0x7A Access: Mixed in RC
PV-Mixed in EP

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	—										TP	APD	URD w1c	FED w1c	NFED w1c	CED w1c
Reset	All zeros															
PF	Implemented per PF as defined in base specification															
VF	Implemented per VF											0	Implemented per VF			

Figure 20-126. PCI Express Device Status Register

Table 20-119. PCI Express Device Status Register Field Description

Bits	Name	Description
15–6	—	Reserved
5	TP	Transactions pending
4	APD	AUX power detected. Note that this bit is always zero for VF.
3	URD	Unsupported request detected
2	FED	Fatal error detected
1	NFED	Non-fatal error detected
0	CED	Correctable error detected

20.3.10.15 PCI Express Link Capabilities Register—0x7C

The PCI Express link capabilities register is shown in Figure 20-127.

Offset 0x7C

Access: Read only in RC
PV-Read only in EP

	31–24	23	22	21	20	19	18	17–15	14–12	11–10	9–4	3–0
R	Port Number	—	AOC	LBWN	DLLARE	—	CPM	L1_EX_LAT	L0s_EX_LAT	ASPM	MAX_LINK_W	MAX_LINK_SP
W												
Reset	0 0 0 0 0 0 0 0	0	1	n	n	0	0	1 1 1	1 1 1	0 1	0 0 1 0 0 0	0 0 1 1
PF	Implemented per PF as defined in base specification											
VF	Implemented per VF as defined in base specification											

Figure 20-127. PCI Express Link Capabilities Register

Table 20-120. PCI Express Link Capabilities Register Field Description

Bits	Name	Description
31–24	Port Number	—
23	—	Reserved
22	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 capable. In EP-mode, this bit is hardwired to 0. In RC-mode it is hardwired to1.
20	Data link layer active reporting capable	Set to 1 when in RC. Set to 0 when in EP.
19	—	Reserved
18	Clock power management	
17–15	L1_EX_LAT	L1 exit latency
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 0011 8.0 GT/s, 5.0 GT/s, and 2.5 GT/s link

20.3.10.16 PCI Express Link Control Register—0x80

The PCI Express link control register is shown in Figure 20-128.

Offset 0x80 Access: Mixedin RC
PV-Mixed only in EP

	15–12	11	10	9	8	7	6	5	4	3	2	1	0
R W	—	LABIE	LBMIE	—	ECPM	EXT_SYNC	CCC	RL	LD	RCB	—	ASPM_CTL	
Reset	All zeros												
PF	Implemented per PF as defined in base specification												
VF	0 0 0 0 0 0	From PF						0	0	From PF			

Figure 20-128. PCI Express Link Control Register

Table 20-121. PCI Express Link Control Register Field Description

Bits	Name	Description
15–12	—	Reserved
11	LABIE	Link autonomous bandwidth interrupt enable
10	LBMIE	Link bandwidth management interrupt enable
9	—	Reserved
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
2	—	Reserved
1–0	ASPM_CTL	Active state power management (ASPM) control

20.3.10.17 PCI Express Link Status Register—0x82

The PCI Express link status register is shown in Figure 20-129.

Offset 0x82 Access: Read only in RC
PV-Read only in EP

	15	14	13	12	11	10	9–4	3–0
R	LABS	LBMS	—	SCC	LT	—	NEG_LINK_W	LINK_SP
W-EP								
W-RC	w1c	w1c						
Reset	0	0	0	1	0	0	0 0 n n n n	0 0 n n
PF	Implemented per PF as defined in base specification							
VF	From PF							

Figure 20-129. PCI Express Link Status Register

Table 20-122. PCI Express Link Status Register Field Description

Bits	Name	Description
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
3–0	LINK_SP	Negotiated link speed. 0001 2.5 GT/s 0010 5.0 GT/s 0011 8.0 GT/s

20.3.10.18 PCI Express Slot Capabilities Register (RC Mode Only)—0x84

The PCI Express slot capabilities register is shown in Figure 20-130.

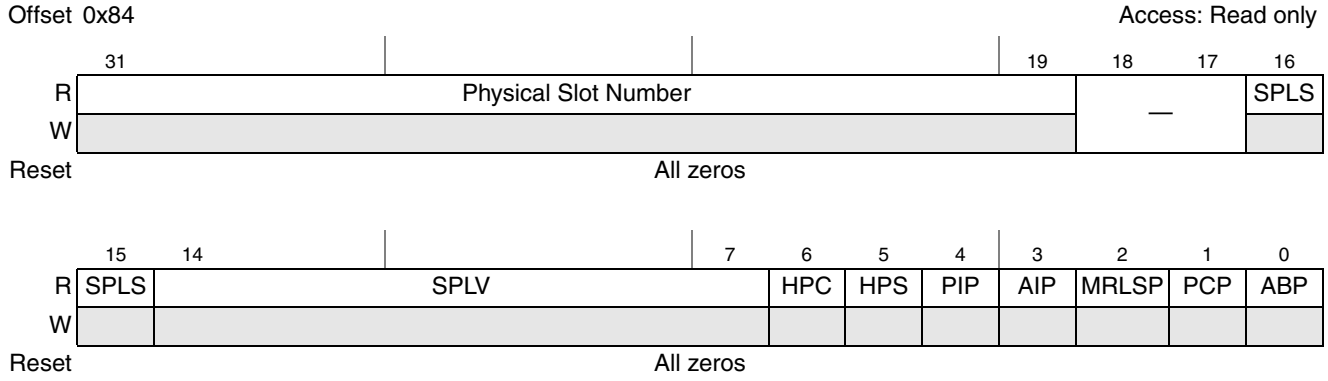


Figure 20-130. PCI Express Slot Capabilities Register

Table 20-123. PCI Express Slot Capabilities Register Field Description

Bits	Name	Description
31–19	Physical Slot Number	This hardware initialized field indicates the physical slot number attached to this Port. This field must be hardware initialized to a value that assigns a slot number that is globally unique. These registers should be initialized to 0 for Ports connected to devices that are either integrated on the system board or integrated within the same silicon as the Switch device or Root Port.
18–17	—	Reserved
16–15	SPLS	Slot power limit scale.
14–7	SPLV	Slot power limit value.

Table 20-123. PCI Express Slot Capabilities Register Field Description (continued)

Bits	Name	Description
6	HPD	Hot plug capable.
5	HPS	Hot plug surprise.
4	PIP	Power indicator present.
3	AIP	Attention indicator present.
2	MRLSP	MRL sensor present.
1	PCP	Power controller present.
0	ABP	Attention button present.

20.3.10.19 PCI Express Slot Control Register (RC Mode Only)—0x88

The PCI Express slot control register is shown in Figure 20-131.

Offset 0x88 Access: Read/Write

	15–11	10	9–8	7–6	5	4	3	2	1	0
R W	—	PCC	PIC	AIC	HPIE	CCIE	PDCE	MRLSCE	PFDE	ABPE
Reset	0000001111000000									

Figure 20-131. PCI Express Slot Control Register

Table 20-124. PCI Express Slot Control Register Field Description

Bits	Name	Description
15–11	—	Reserved
10	PCC	Power controller control.
9–8	PIC	Power indicator control. If a power indicator is implemented, set the power indicator to the written state. Reads of this field must reflect the value from the latest write, even if the corresponding hot-plug command is not complete, unless software issues a write without waiting for the previous command to complete in which case the read value is undefined. Defined encodings are shown below. **Note:** The default value of this field must be one of the non-reserved values. If the power indicator present bit in the slot capabilities register is 0, this bit is permitted to be read-only with a value of 00. Defined encodings are: 00 Reserved 01 On 10 Blink 11 Off

Table 20-124. PCI Express Slot Control Register Field Description (continued)

Bits	Name	Description
7–6	AIC	Attention indicator control. If an attention indicator is implemented, writes to this field set the attention indicator to the written state. Reads of this field must reflect the value from the latest write, even if the corresponding hot-plug command is not complete, unless software issues a write without waiting for the previous command to complete in which case the read value is undefined. **Note:** The default value of this field must be one of the non-reserved values. if the attention indicator present bit in the slot capabilities register is 0, this bit is permitted to be readonly with a value of 00. Defined encodings are: 00 Reserved 01 On 10 Blink 11 Off
5	HPIE	Hot plug interrupt enable.
4	CCIE	Command completed interrupt enable.
3	PDCE	Presence detect changed enable.
2	MRLSCE	MRL sensor changed enable.
1	PFDE	Power fault detected enable.
0	ABPE	Attention button pressed enable.

20.3.10.20 PCI Express Slot Status Register (RC Mode Only)—0x8A

The PCI Express slot status register is shown in Figure 20-132.

Offset 0x8A Access: Mixed

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	—									PDS	MRLSS	CC	PDC	MRLSC	PFD	ABP
W												w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0

Figure 20-132. PCI Express Slot Status Register

Table 20-125. PCI Express Slot Status Register Field Descriptions

Bits	Name	Description
15–7	—	Reserved
6	PDS	This bit indicates the presence of an adapter in the slot, reflected by the logical âORâ 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. Defined encodings are: 0 Slot Empty 1 Card Present in slot 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.
5	MRLSS	MRL sensor state. 0 MRL closed 1 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.

20.3.10.21 PCI Express Root Control Register (RC Mode Only)—0x8C

The PCI Express root control register is shown in Figure 20-133.

Offset 0x8C Access: Read/Write

	15–4	3	2	1	0
R W	—	PMEIE	SEFEE	SENFEE	SECEE
Reset	All zeros				

Figure 20-133. PCI Express Root Control Register

Table 20-126. PCI Express Root Control Register Field Description

Bits	Name	Description
15–4	—	Reserved
3	PMEIE	PME interrupt enable.
2	SEFEE	System error on fatal error enable.

Table 20-126. PCI Express Root Control Register Field Description (continued)

Bits	Name	Description
1	SENFEE	System error on non-fatal error enable.
0	SECEE	System error on correctable error enable.

20.3.10.22 PCI Express Root Status Register (RC Mode Only)—0x90

The PCI Express root status register is shown in Figure 20-134.

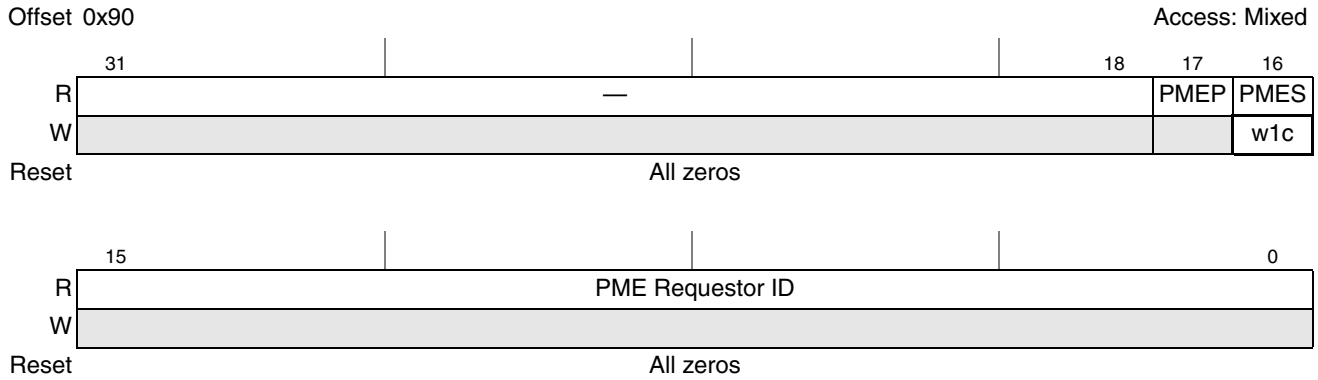


Figure 20-134. PCI Express Root Status Register

Table 20-127. PCI Express Root Status Register Field Description

Bits	Name	Description
31–18	—	Reserved
17	PMEP	PME pending.
16	PMES	PME status.
15–0	PME Requestor ID	PME requestor ID.

20.3.10.23 PCI Express Device Capabilities 2 Register—0x94

The PCI Express device capabilities 2 register is shown in Figure 20-135.

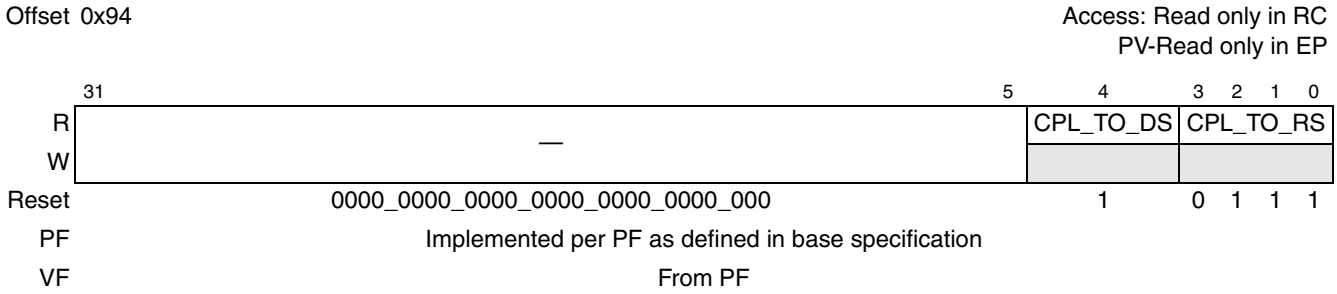


Figure 20-135. PCI Express Device Capabilities 2 Register

Table 20-128. PCI Express Device Capabilities 2 Register Field Description

Bits	Name	Description
31–5	—	Reserved
4	CPL_TO_DS	Completion timeout disable supported
3–0	CPL_TO_RS	Completion timeout ranges supported

20.3.10.24 PCI Express Device Control 2 Register—0x98

The PCI Express device control 2 register is shown in Figure 20-136.

Offset 0x98 Access: Read/Write in RC
PV-Read/Write in EP

	15–6	5	4	3–0
R W	—	ARIFE	CPL_TOD	CPL_TO_VAL
Reset	All zeros			
PF	Implemented per PF as defined in base specification			
VF	Copy from PF			

Figure 20-136. PCI Express Device Control 2 Register

Table 20-129. PCI Express Device Control 2 Register Field Description

Bits	Name	Description
15–6	—	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

20.3.10.25 PCI Express Link Capabilities 2 Register—0x9C

The PCI Express link capability 2 register is shown in Figure 20-137.

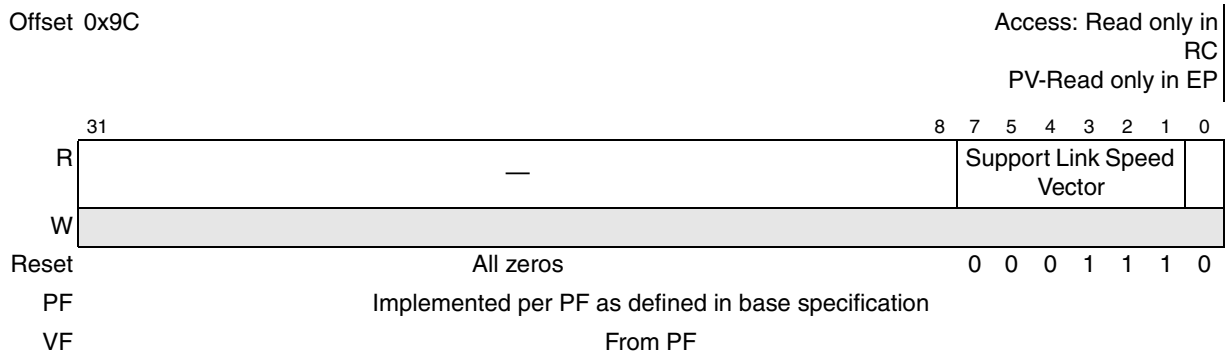


Figure 20-137. PCI Express Link Capabilities 2 Register

Table 20-130. PCI Express Link Capabilites 2 Field Description

Bits	Name	Description
7-1	Support Link Speed 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 RsvP
0	—	Reserved

20.3.10.26 PCI Express Link Control 2 Register—0xA0

The PCI Express link control 2 register is shown in Figure 20-138.

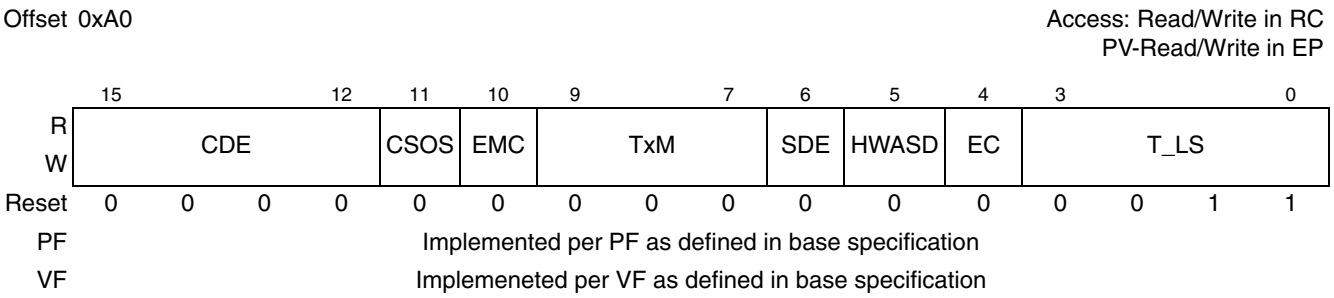


Figure 20-138. PCI Express Link Control 2 Register

Table 20-131. PCI Express Link Control 2 Register Field Description

Bits	Name	Description
15–12	CDE	Compliance 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

20.3.10.27 PCI Express Link Status 2 Register—0xA2

The PCI Express link status 2 register is shown in Figure 20-139.

Offset 0xA2 Access: Read only in RC
PV-Read only in EP

	15–6	5	4	3	2	1	0
R	—	LER	EP3S	EP2S	EP1S	EC	DE_LVL
W							
Reset	All zeros						
PF	Implemented per PF as defined in base specification						
VF	From PF						

Figure 20-139. PCI Express Link Status 2 Register

Table 20-132. PCI Express Link Status 2 Register Field Description

Bits	Name	Description
15–6	—	Reserved
5	LER	Link equalization request
4	EP3S	Equalization phase 3 complete
3	EP2S	Equalization phase 2 complete
2	EP1S	Equalization phase 1 complete
1	EC	Equalization complete
0	DE_LVL	Current de-emphasis level

20.3.10.28 PCI Express MSI-X Message Capability ID Register (EP Mode Only)—0xB0

The PCI Express MSI-X message capability ID register is shown in Figure 20-140.

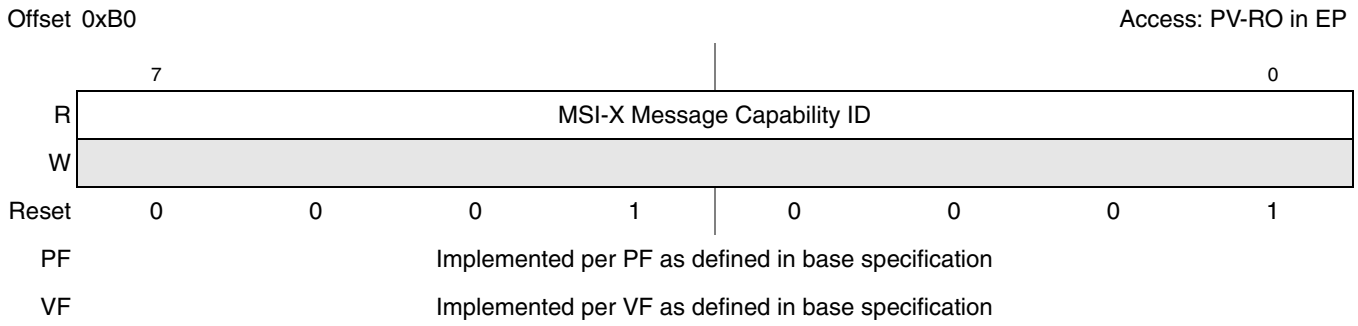


Figure 20-140. PCI Express MSI-X Capability ID Register

Table 20-133. PCI Express MSI-X Capability ID Register Field Description

Bits	Name	Description
7–0	MSI-X Message Capability ID	MSI Message = 0x11

20.3.10.29 PCI Express MSI-X Message Control Register (EP Mode Only)—0xB2

The PCI Express MSI-X message control register is shown in Figure 20-141.

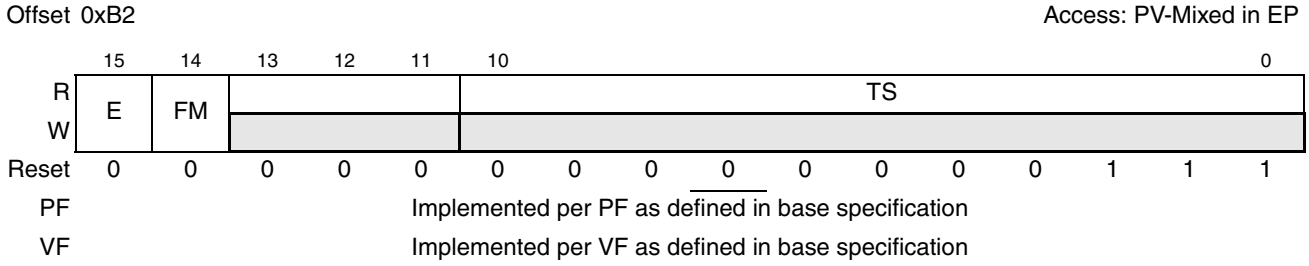


Figure 20-141. PCI Express MSI-X Message Control Register

Table 20-134. PCI Express MSI-X Message Control Register Field Description

Bits	Name	Description
15	E	Enable. If 1 and the MSI Enable bit in the MSI Message Control register is 0, the function is permitted to use MSI-X to request service.
14	FM	Function Mask. If 1, all of the vectors associated with the function are masked, regardless of their per-vector Mask bit states. If 0, each vector's Mask bit determines whether the vector is masked or not.!
13-11		Reserved
10-0	TS	Table Size. System software reads this field to determine the MSI-X Table Size N, which is encoded as N-1. For example, a returned value of "000_0000_0011" indicates a table size of 4.

20.3.10.30 PCI Express MSI-X Table Offset/BIR Register (EP Mode Only)—0xB4

The PCI Express MSI-X table offset/BIR register is shown in Figure 20-142.

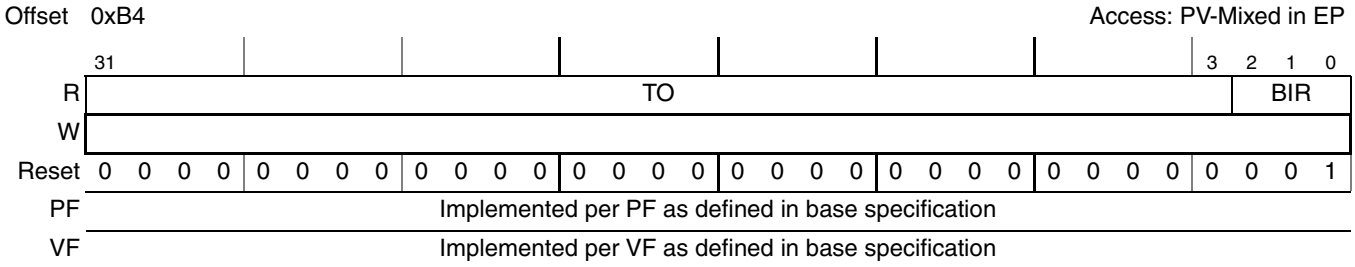


Figure 20-142. PCI Express MSI-X Table Offset/BIR Register

Table 20-135. PCI Express MSI Table Offset/BIR Register Field Description

Bits	Name	Description
31–3	TO	Table Offset. Used as an offset from the address contained by one of the function's Base Address registers to point to the base of the MSI-X Table.
2–0	BIR	Indicates which one of a function's Base Address registers, located beginning at 10h in Configuration Space, is used to map the function's MSI-X Table into Memory Space. BIR Value Base Address register 000 10h 001 14h 010 18h 011 1Ch 100 20h 101 24h 110 Reserved 111 Reserved

20.3.10.31 PCI Express MSI-X PBA Offset/BIR Register (EP Mode Only)—0xB8

The PCI Express MSI-X PBA Offset/BIR register is shown in Figure 20-143.

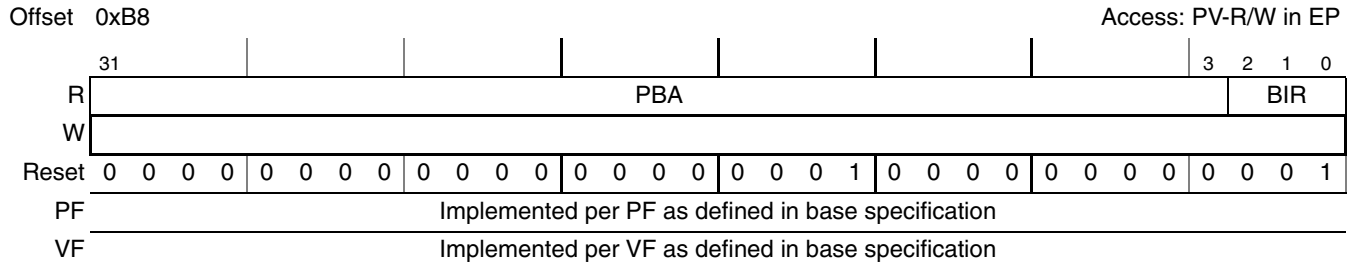


Figure 20-143. PCI Express MSI-X PBA Offset/BIR Register

Table 20-136. PCI Express MSI-X PBA Offset/BIR Field Description

Bits	Name	Description
31–3	PBA	PBA Offset. Used as an offset from the address contained by one of the function's Base Address registers to point to the base of the MSI-X PBA.
2–0	BIR	Indicates which one of a function's Base Address registers, located beginning at 10h in Configuration Space, is used to map the function's MSI-X PBA into Memory Space. BIR Value Base Address register 000 10h 001 14h 010 18h 011 1Ch 100 20h 101 24h 110 Reserved 111 Reserved

20.3.11 PCI Express Extended Configuration Space

Reserved

		Address Offset (Hex)
PCI Compatible Configuration Header (See Section 20.3.9, “PCI Compatible Configuration Headers,” for more information.)		000 03F
PCI-Compatible Device-Specific Configuration Space (See Section 20.3.10, “PCI Compatible Device-Specific Configuration Space,” for more information.)		040 0FF
Next Capability Offset (RC=NULL;EP=0x140)/ Capability Version	Advanced Error Reporting Capability ID	100
Uncorrectable Error Status		104
Uncorrectable Error Mask		108
Uncorrectable Error Severity		10C
Correctable Error Status		110
Correctable Error Mask		114
Advanced Error Capabilities and Control		118
Header Log		11C 120 124 128
Root Error Command		12C
Root Error Status		130
Error Source ID	Correctable Error Source ID	134
		138
		13C
Next Capability Offset (0x150)/Capability Version	ARI Extended Capability ID	140
ARI Control	ARI Capabilites	144
		148
		14C
Next Capability Offset (NULL)/Capability Version	SR-IOV Extended Capability ID	150
SR-IOV Capabilities		154
SR-IOV Status	SR-IOV Control	158
TotalVFs (RO)	InitialVFs (RO)	15C
Function Dependency Link	NumVFs (RW)	160
VFStride (RO)	FirstVF Offset (RO)	164
VF Device ID		168
Supported Page Size (RO)		16C
System Page Size (RW)		170
VF BAR0 (RW)		174
VF BAR1 (RW)		178
VF BAR2 (RW)		17C
VF BAR3 (RW)		180
VF BAR4 (RW)		184
VF BAR5 (RW)		188

Figure 20-144. PCI Express Extended Configuration Space (SR-IOV)

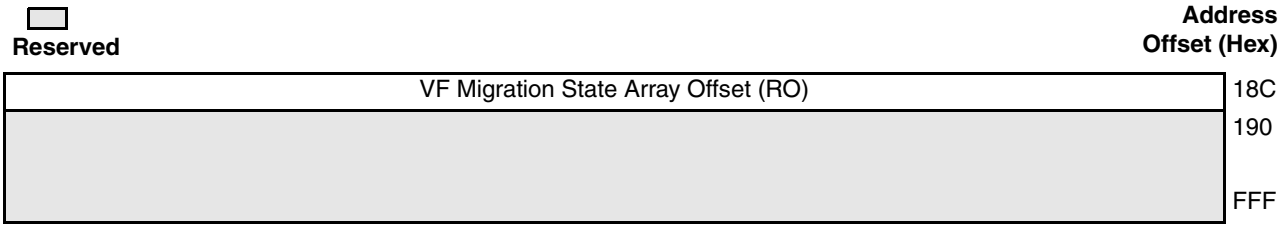


Figure 20-144. PCI Express Extended Configuration Space (SR-IOV)

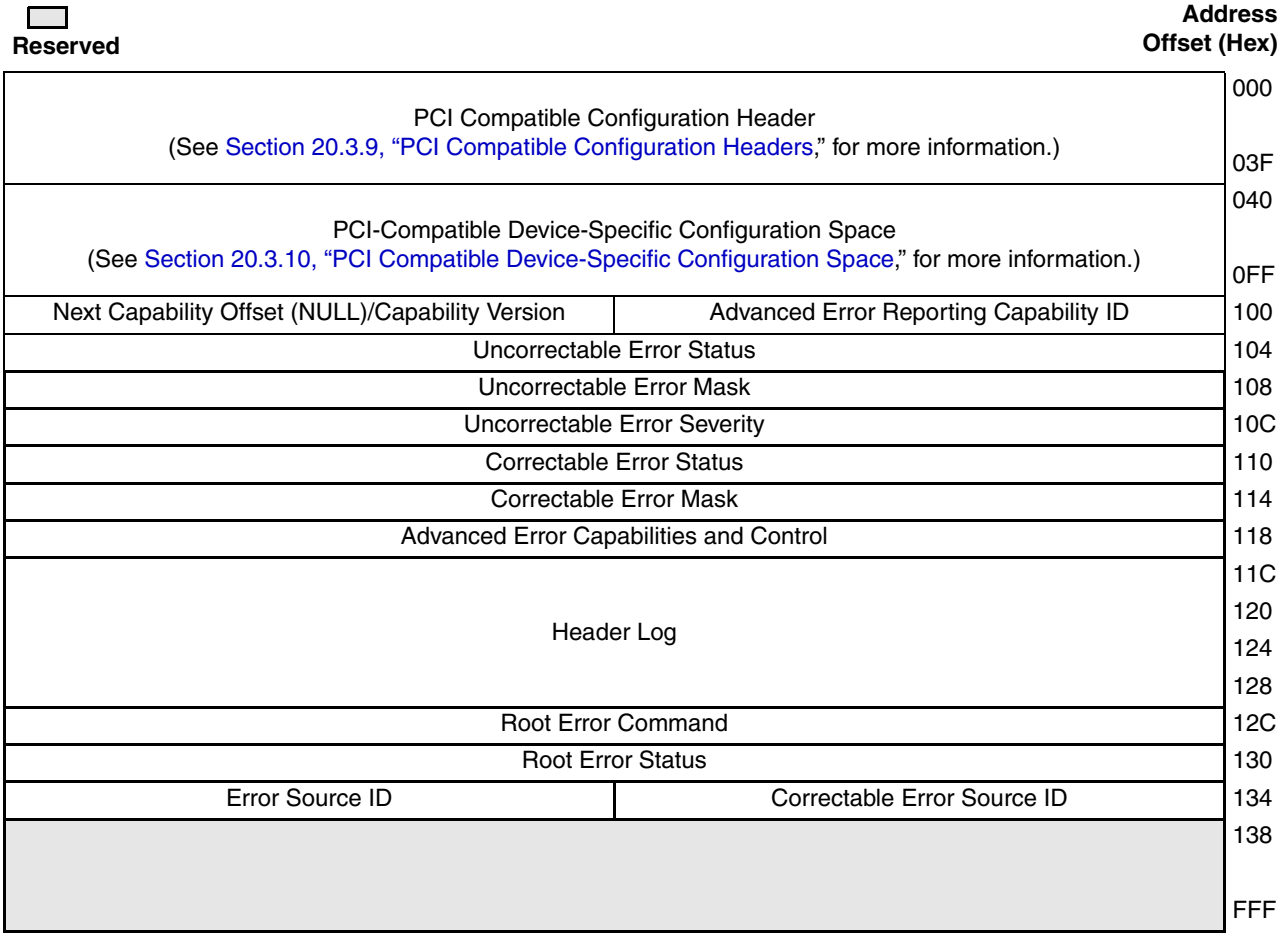


Figure 20-145. PCI Express Extended Configuration Space (non-SR-IOV)

20.3.11.1 PCI Express Advanced Error Reporting Capability ID Register—0x100

The PCI Express advanced error reporting capability ID register is shown in Figure 20-146.

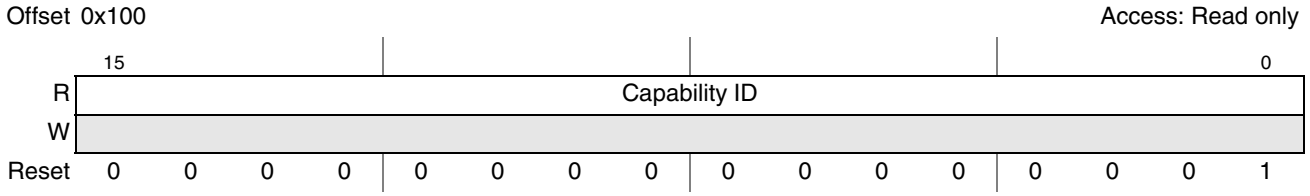


Figure 20-146. PCI Express Advanced Error Reporting Capability ID Register

Table 20-137. PCI Express Advanced Error Reporting Capability ID Register Field Description

Bits	Name	Description
15–0	Capability ID	Advanced error reporting capability = 0x0001

20.3.11.2 PCI Express Uncorrectable Error Status Register—0x104

The PCI Express uncorrectable error status register is shown in Figure 20-147.

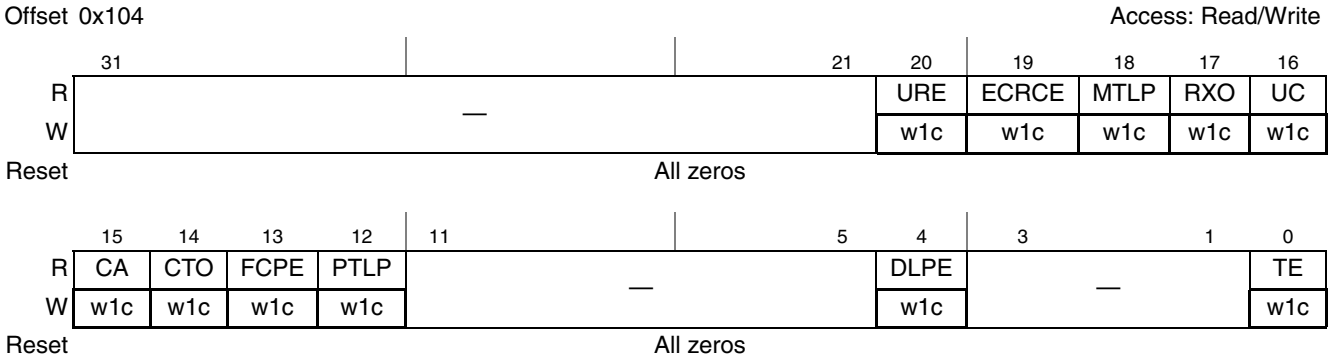


Figure 20-147. PCI Express Uncorrectable Error Status Register

Table 20-138. PCI Express Uncorrectable Error Status Register Field Description

Bits	Name	Description
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.

Table 20-138. PCI Express Uncorrectable Error Status Register Field Description (continued)

Bits	Name	Description
13	FCPE	Flow control protocol error status.
12	PTLP	Poisoned TLP status.
11–5	—	Reserved
4	DLPE	Data link protocol error status.
3–1	—	Reserved
0	TE	Training error status.

20.3.11.3 PCI Express Uncorrectable Error Mask Register—0x108

The PCI Express uncorrectable error mask register is shown in Figure 20-148.

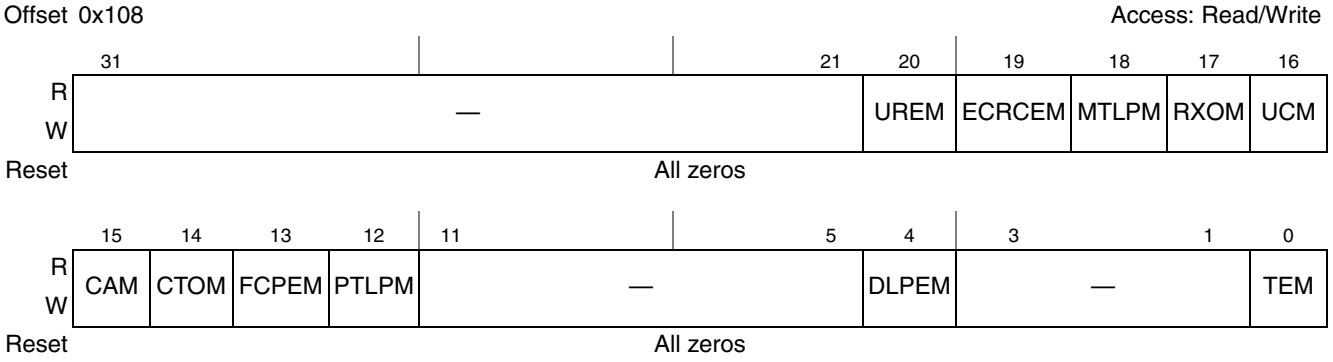


Figure 20-148. PCI Express Uncorrectable Error Mask Register

Table 20-139. PCI Express Uncorrectable Error Mask Register Field Description

Bits	Name	Description
31–21	—	Reserved
20	UREM	Unsupported request error mask.
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.

Table 20-139. PCI Express Uncorrectable Error Mask Register Field Description (continued)

Bits	Name	Description
3–1	—	Reserved
0	TEM	Training error mask.

20.3.11.4 PCI Express Uncorrectable Error Severity Register—0x10C

The PCI Express uncorrectable error severity register is shown in Figure 20-149.

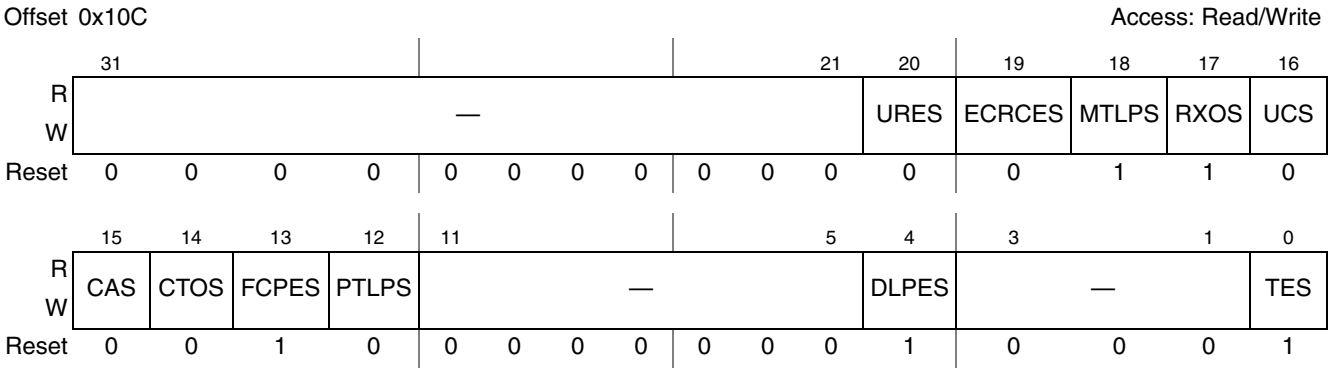


Figure 20-149. PCI Express Uncorrectable Error Severity Register

Table 20-140. PCI Express Uncorrectable Error Severity Register Field Description

Bits	Name	Description
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	DLPES	Data link protocol error severity.
3–1	—	Reserved
0	TES	Training error severity.

20.3.11.5 PCI Express Correctable Error Status Register—0x110

The PCI Express correctable error status register is shown in Figure 20-150.

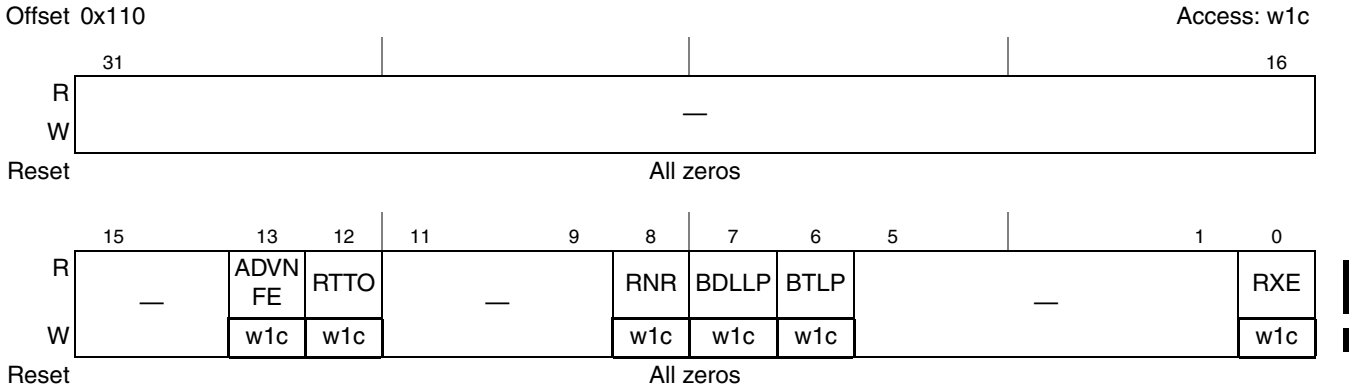


Figure 20-150. PCI Express Correctable Error Status Register

Table 20-141. PCI Express Correctable Error Status Register Field Description

Bits	Name	Description
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 Section 20.3.11.6, "PCI Express Correctable Error Mask Register—0x114 is checked to determine whether to proceed further with logging and signaling
12	RTTO	Replay timer timeout status
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

20.3.11.6 PCI Express Correctable Error Mask Register—0x114

The PCI Express correctable error mask register is shown in Figure 20-151.

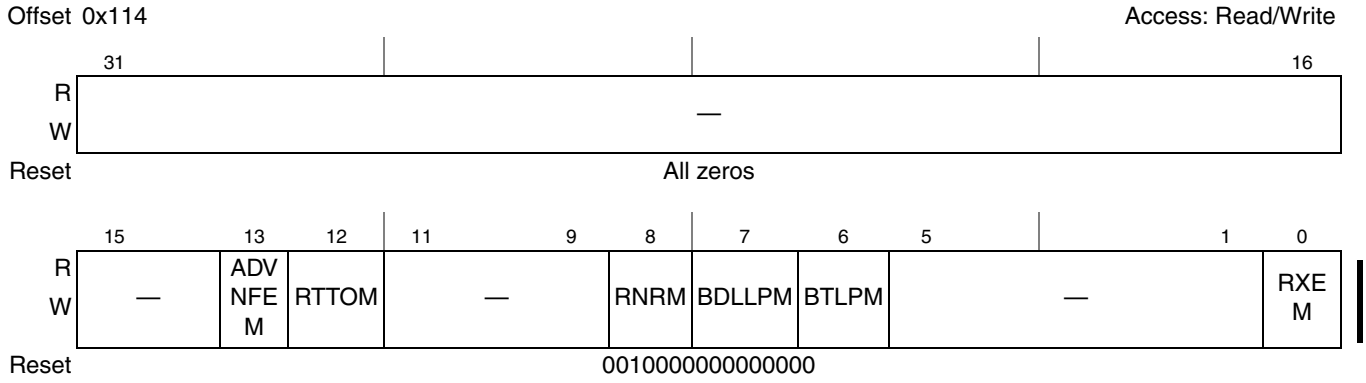


Figure 20-151. PCI Express Correctable Error Mask Register

Table 20-142. PCI Express Correctable Error Mask Register Field Description

Bits	Name	Description
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

20.3.11.7 PCI Express Advanced Error Capabilities and Control Register—0x118

The PCI Express advanced error capabilities and control register is shown in Figure 20-152.

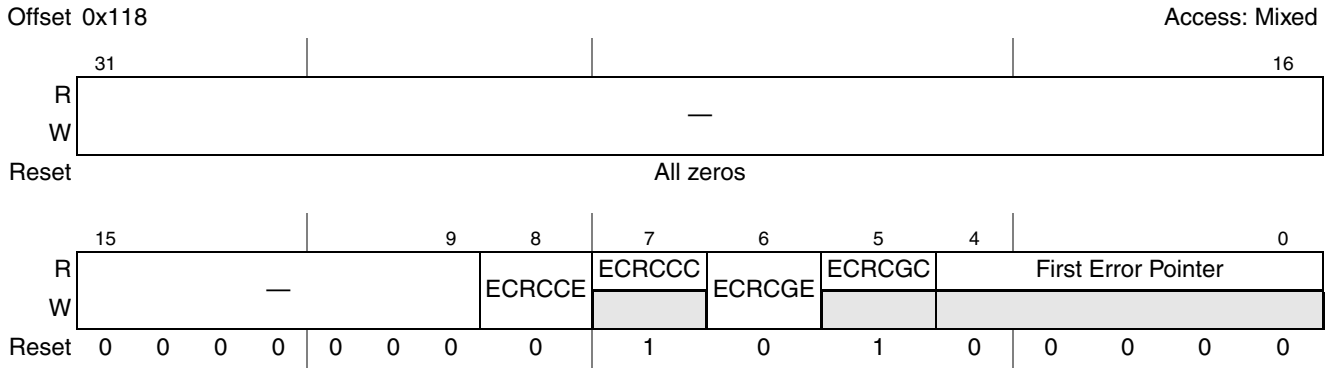


Figure 20-152. PCI Express Advanced Error Capabilities and Control Register

Table 20-143. PCI Express Advanced Error Capabilities and Control Register Field Description

Bits	Name	Description
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.

20.3.11.8 PCI Express Header Log Register—0x11C–0x12B

The PCI Express header log register is shown in Figure 20-153.

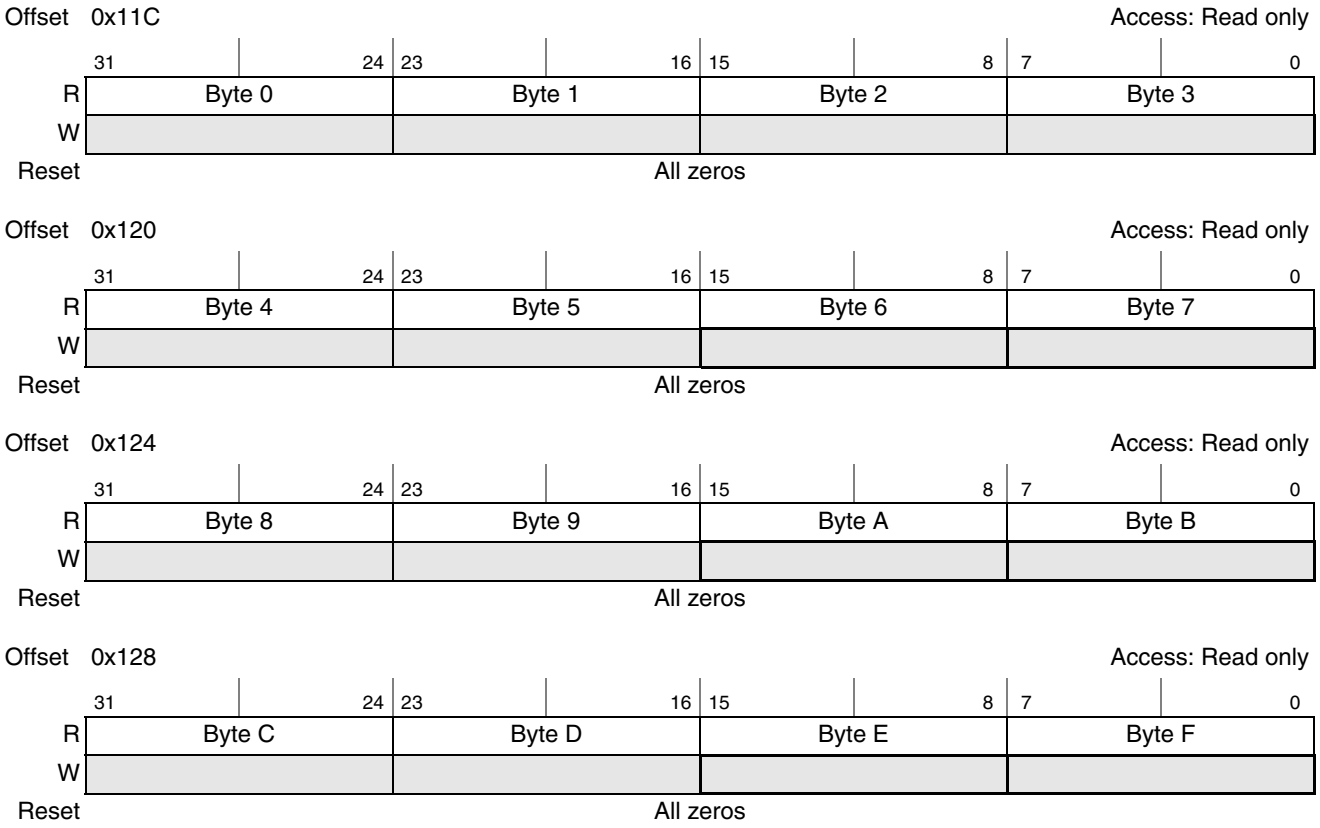


Figure 20-153. PCI Express Header Log Register

Table 20-144. PCI Express Header Log Register Field Description

Bits	Name	Description
127–0	Header Log	Header of TLP associated with error.

20.3.11.9 PCI Express Root Error Command Register (RC Mode Only)—0x12C

The PCI Express root error command register is shown in Figure 20-154.

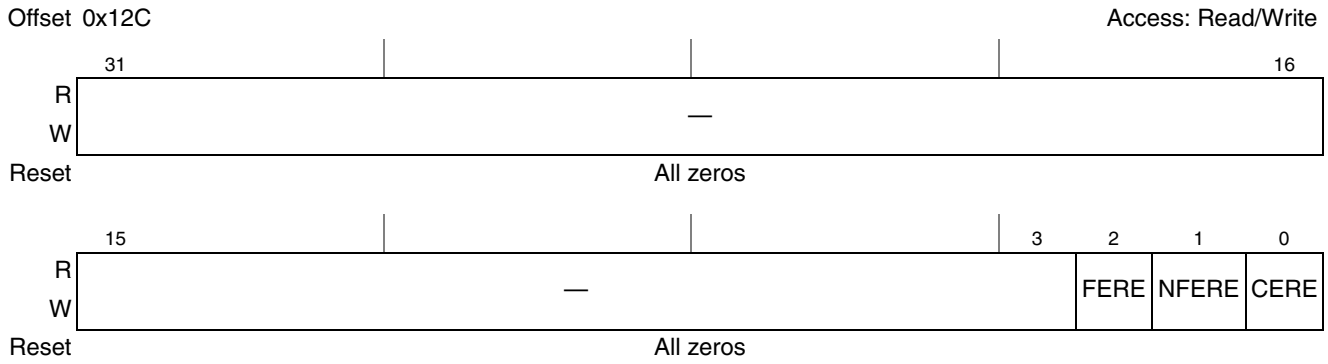


Figure 20-154. PCI Express Root Error Command Register

Table 20-145. PCI Express Root Error Command Register Field Description

Bits	Name	Description
31–3	—	Reserved
2	FERE	Fatal error reporting enable.
1	NFERE	Non-fatal error reporting enable
0	CERE	Correctable error reporting enable

20.3.11.10 PCI Express Root Error Status Register (RC Mode Only)—0x130

The PCI Express root error status register is shown in Figure 20-155.

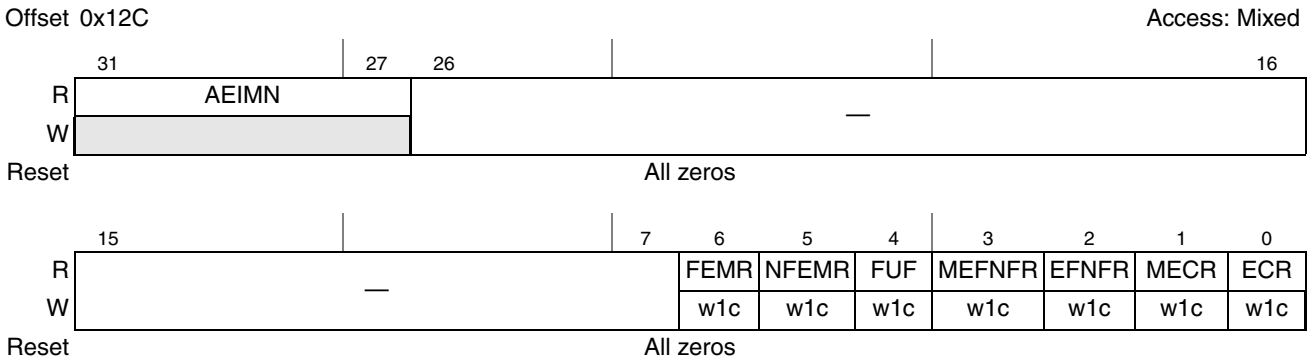


Figure 20-155. PCI Express Root Error Status Register

Table 20-146. PCI Express Root Error Command Status Field Description

Bits	Name	Description
31–27	AEIMN	Advanced error interrupt message number.
26–7	—	Reserved
6	FEMR	Fatal error messages received.

Table 20-146. PCI Express Root Error Command Status Field Description (continued)

Bits	Name	Description
5	NFEMR	Non-fatal error messages received.
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.

20.3.11.11 PCI Express Correctable Error Source ID Register—0x134

The PCI Express correctable error source ID register is shown in Figure 20-156.

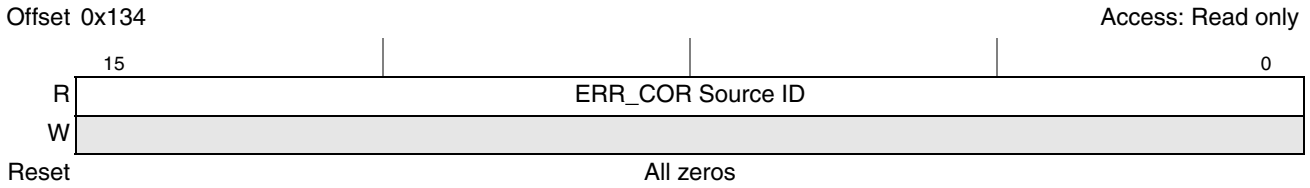


Figure 20-156. PCI Express Correctable Error Source ID Register

Table 20-147. PCI Express Correctable Error Source ID Register Field Description

Bits	Name	Description
15–0	ERR_COR Source ID	Loaded with the Requestor 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.

20.3.11.12 PCI Express Error Source ID Register—0x136

The PCI Express error source ID register is shown in Figure 20-157.

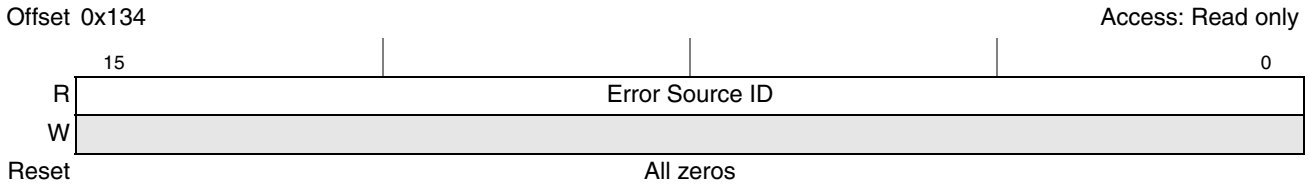


Figure 20-157. PCI Express Correctable Error Source ID Register

Table 20-148. PCI Express Correctable Error Source ID Register Field Description

Bits	Name	Description
15–0	Error Source ID	ERR_FATAL/NONFATAL source ID

20.3.11.13 PCI Express ARI Extended Capability ID Register (SR-IOV-only)—0x140

The PCI Express ARI capability ID register is shown in Figure 20-158.

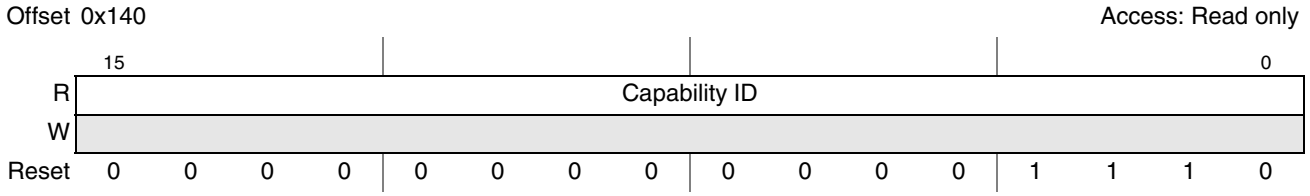


Figure 20-158. PCI Express ARI Extended Capability ID Register

Table 20-149. PCI Express ARI Extended Capability ID Register Field Description

Bits	Name	Description
15–0	Extended Capability ID	ARI extended capability = 0x000E

20.3.11.14 PCI Express ARI Capability Register (SR-IOV-only)—0x144

The PCI Express ARI capability register is shown in Figure 20-159.

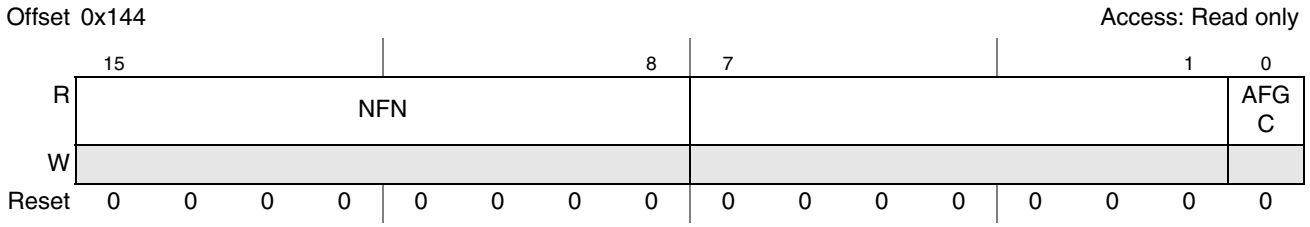


Figure 20-159. PCI Express ARI Capability ID Register

Table 20-150. PCI Express ARI Capability Register Field Description

Bits	Name	Description
15–8	NFN	Next function number. This field indicates the Function Number of the next higher numbered Function in the Device, or 00h if there are no higher numbered Functions. Function 0starts this linked list of Functions.
7–1		Reserved
0	AFGC	.ACS Function Groups Capability. Applicable only for Function 0; must be 0b for all other Functions. If 1b, indicates that the ARI Device supports Function Group level granularity for ACS P2P Egress Control via its ACS Capability structures.

20.3.11.15 PCI Express ARI Control Register (SR-IOV-only)—0x146

The PCI Express ARI control register is shown in Figure 20-160.

Offset 0x146 Access: Read only

	15								7	6			4	3	2	1	0
R										FG						AFGE	MFGE
W																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0

Figure 20-160. PCI Express ARI Control Register

Table 20-151. PCI Express ARI Control Register Field Description

Bits	Name	Description
15–7		Reserved
6–4	FG	Function Group. Assigns a Function Group Number to this Function.
3–2		Reserved
1	AFGE	ACS Function Groups Enable. Applicable only for Function 0; must be hardwired to 0b for all other Functions. When set, each Function in the ARI Device must associate bits within its Egress Control Vector with Function Group Numbers rather than Function Numbers. Default value of this field is 0b. Must be hardwired to 0b if the ACS Function Groups Capability bit is 0b.
0	MFGE	MFVC Function Groups Enable. Applicable only for Function 0; must be hardwired to 0b for all other Functions. When set, the ARI Device must interpret entries in its Function Arbitration Table as Function Group Numbers rather than Function Numbers. Default value of this field is 0b. Must be hardwired to 0b if the MFVC Function Groups Capability bit is 0b.

20.3.11.16 PCI Express SR-IOV Extended Capability ID Register (SR-IOV-only)—0x150

The PCI Express SR-IOV capability ID register is shown in Figure 20-161.

Offset 0x150 Access: Read only

	15																0
R	Capability ID																
W																	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	

Figure 20-161. PCI Express SR-IOV Extended Capability ID Register

Table 20-152. PCI Express SR-IOV Extended Capability ID Register Field Description

Bits	Name	Description
15–0	Extended Capability ID	SR-IOV extended capability = 0x0010

20.3.11.17 PCI Express SR-IOV Capabilities Register (SR-IOV-only)—0x150

The PCI Express SR-IOV capabilities register is shown in Figure 20-162.

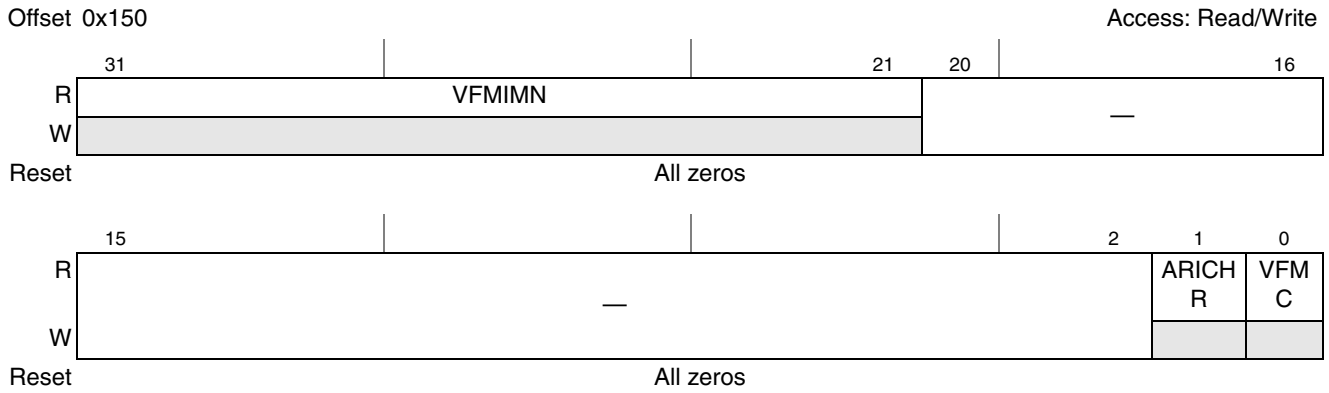


Figure 20-162. PCI Express SR-IOV Capabilities Register

Table 20-153. PCI Express SR-IOV Capabilities Register Field Description

Bits	Name	Description
31–21	VFMIMN	VF Migration Interrupt Message Number. Indicates the MSI/MSI-X vector used for migration interrupts. The value in this field is undefined if VF Migration Capable is clear.
20-2		Reserved
1	ARICHR	ARI Capable Hierarchy Preserved. If set, the ARI Capable Hierarchy bit is preserved across certain power state transitions.
0	VFMC	VF Migration Capable. If set, the PF is Migration Capable and operating under a Migration Capable MR-PCIM.

20.3.11.18 PCI Express SR-IOV Control Register (SR-IOV-only)—0x158

The PCI Express SR-IOV control register is shown in Figure 20-163.

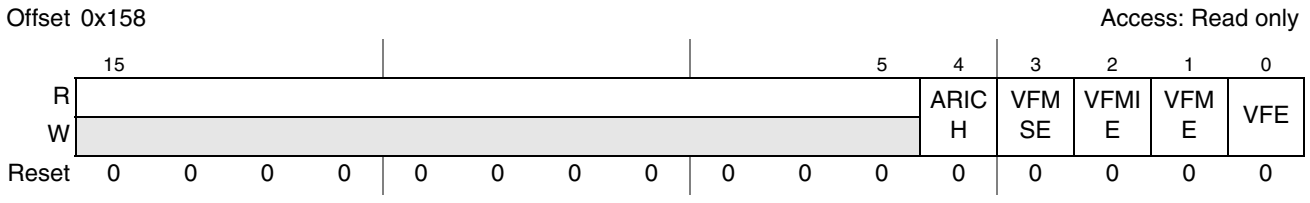


Figure 20-163. PCI Express SR-IOV Control Register

Table 20-154. PCI Express SR-IOV Control Register Field Description

Bits	Name	Description
15–5	0	Reserved
4	ARICH	ARI Capable Hierarchy PCI Express Endpoint: The device is permitted to locate VFs in Function numbers 8 to 255 of the captured Bus Number. Default value is 0. This field is RW in the lowest numbered PF of the Device and is Read Only Zero in all other PFs. Root Complex Integrated Endpoint: Not applicable. Must hardwire the bit to 0
3	VFMSE	VF MSE. Memory Space Enable for Virtual Functions.
2	VFMIE	VF Migration Interrupt Enable. Enables/Disables VF Migration State Change Interrupt. Default value is 0b.
1	VFME	VF Migration Enable. Enables/Disables VF Migration Support. Default value is 0b.
0	VFE	VF Enable. Enables/Disables VFs. Default value is 0.

20.3.11.19 PCI Express SR-IOV Status Register (SR-IOV-only)—0x15A

The PCI Express SR-IOV status register is shown in Figure 20-164.

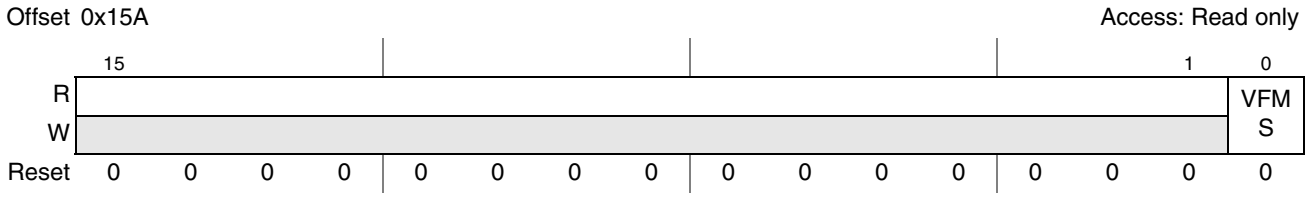


Figure 20-164. PCI Express SR-IOV Status Register

Table 20-155. PCI Express SR-IOV Status Register Field Description

Bits	Name	Description
15–1	0	Reserved
0	VFMS	VF Migration Status. Indicates a VF Migration In or Migration Out Request has been issued by MR-PCIM. To determine the cause of the event, software may scan the VF State Array. Default value is 0b.

20.3.11.20 PCI Express SR-IOV Initial VF Register (SR-IOV-only)—0x160

The PCI Express SR-IOV initial VF register is shown in Figure 20-165.

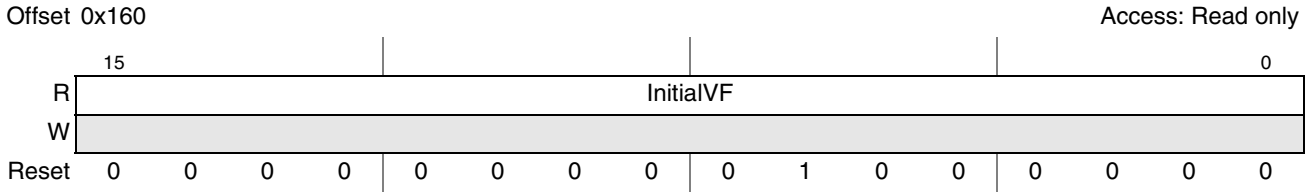


Figure 20-165. PCI Express SR-IOV Initial VF Register

Table 20-156. PCI Express SR-IOV Initial VF Register Field Description

Bits	Name	Description
15–0	InitialVF	Initial VF. Indicates to SR-PCIM the number of VFs that are initial associated with the PF.The minimum value of InitialVFs is 0.

20.3.11.21 PCI Express SR-IOV Total VF Register (SR-IOV-only)—0x162

The PCI Express SR-IOV total VF register is shown in Figure 20-166.

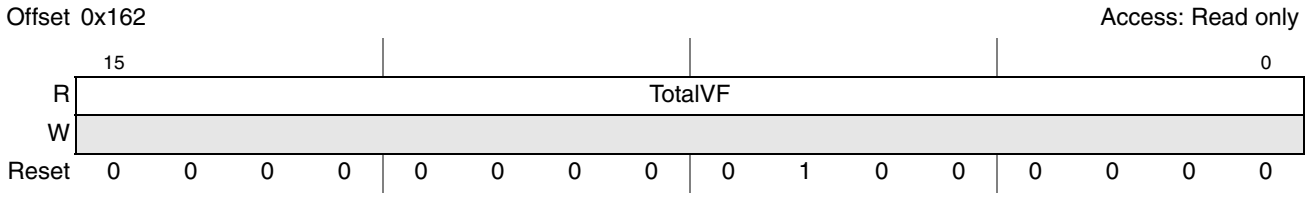


Figure 20-166. PCI Express SR-IOV Total VF Register

Table 20-157. PCI Express SR-IOV Total VF Register Field Description

Bits	Name	Description
15–0	TotalVF	Total VF. indicates the maximum number of VFs that could be associated with the PF.The minimum value of TotalVFs is 0.

20.3.11.22 PCI Express SR-IOV Number VF Register (SR-IOV-only)—0x164

The PCI Express SR-IOV number VF register is shown in Figure 20-167.

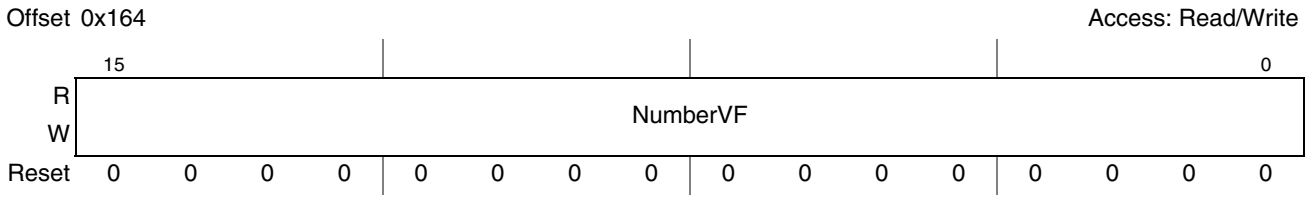


Figure 20-167. PCI Express SR-IOV Number VF Register

Table 20-158. PCI Express SR-IOV Number VF Register Field Description

Bits	Name	Description
15–0	NumberVF	Number of VF. Controls the number of VFs that are visible. Only supports a number that is a power of 2 (that is, 1, 2, 4, 8, 16...).

20.3.11.23 PCI Express SR-IOV Function Dependency Link Register (SR-IOV-only)—0x166

The PCI Express SR-IOV function dependency link register is shown in Figure 20-168.

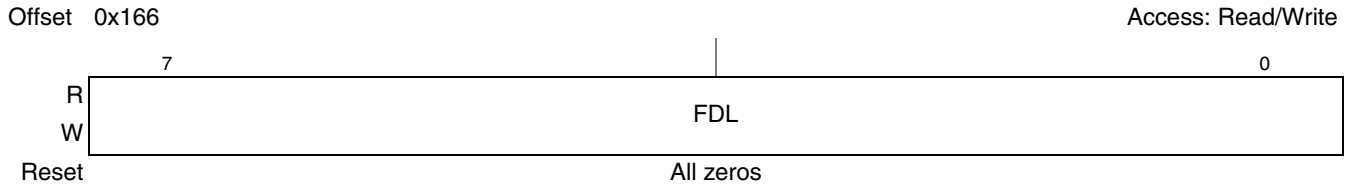


Figure 20-168. PCI Express SR-IOV Function Dependency Link Register

Table 20-159 describes the PCI Express SR-IOV function dependency link register fields.

Table 20-159. PCI Express SR-IOV Function Dependency Link Field Description

Bits	Name	Description
7–0	FDL	Function Dependency Link. This field describes dependencies between PFs.

20.3.11.24 PCI Express SR-IOV First VF Offset Register (SR-IOV-only)—0x168

The PCI Express SR-IOV first VF offset register is shown in Figure 20-169.

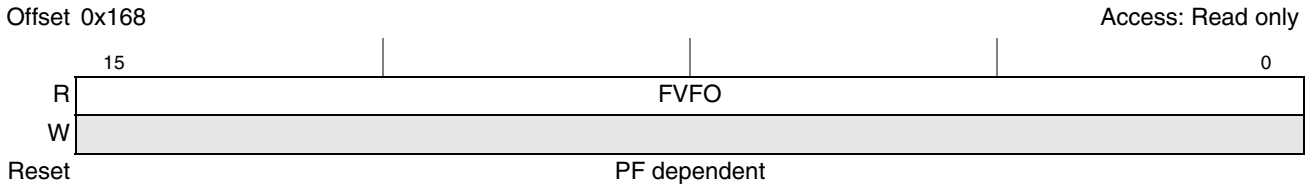


Figure 20-169. PCI Express First VF Offset Register

Table 20-160. PCI Express SR-IOV First VF Offset Register Field Description

Bits	Name	Description
15–0	FVFO	First VF Offset. Defines the Routing ID offset of the first VF that is associated with the PF that contains this Capability structure. For PF0, it is 0x0004. For PF1, it is 0x0100.

20.3.11.25 PCI Express SR-IOV VF Stride Register (SR-IOV-only)—0x16A

The PCI Express SR-IOV VF stride register is shown in Figure 20-170.

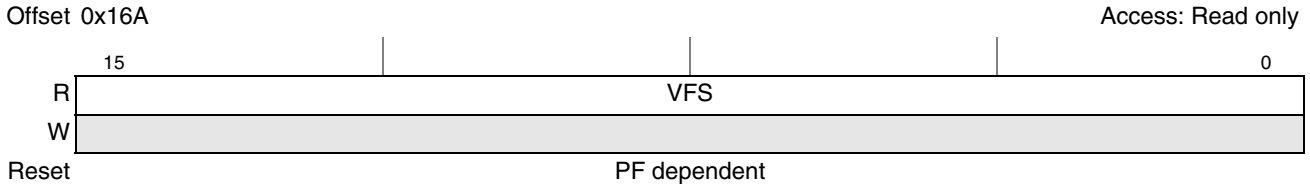


Figure 20-170. PCI Express SR-IOV VF Stride Register

Table 20-161. PCI Express SR-IOV VF Stride Register Field Description

Bits	Name	Description
15–0	VFS	VF Stride. Defines the Routing ID offset from one VF to the next one for all VFs associated with the PF that contains this Capability structure. For PF0, it is 0x0004. For PF1, it is 0x0100.

20.3.11.26 PCI Express SR-IOV Device ID Register (SR-IOV-only)—0x16C

The PCI Express SR-IOV device ID register is shown in Figure 20-171.

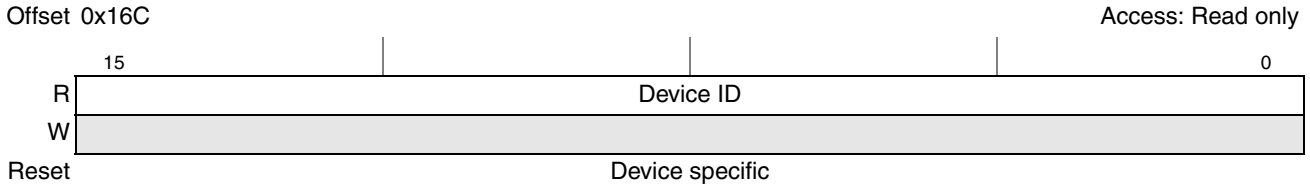


Figure 20-171. PCI Express SR-IOV Device ID Register

Table 20-162. PCI Express SR-IOV Device ID Register Field Description

Bits	Name	Description
15–0	Device ID	Device ID.

20.3.11.27 PCI Express SR-IOV Supported Page Size Register (SR-IOV-only)—016E

The PCI Express SR-IOV supported page size register is shown in Figure 20-172.

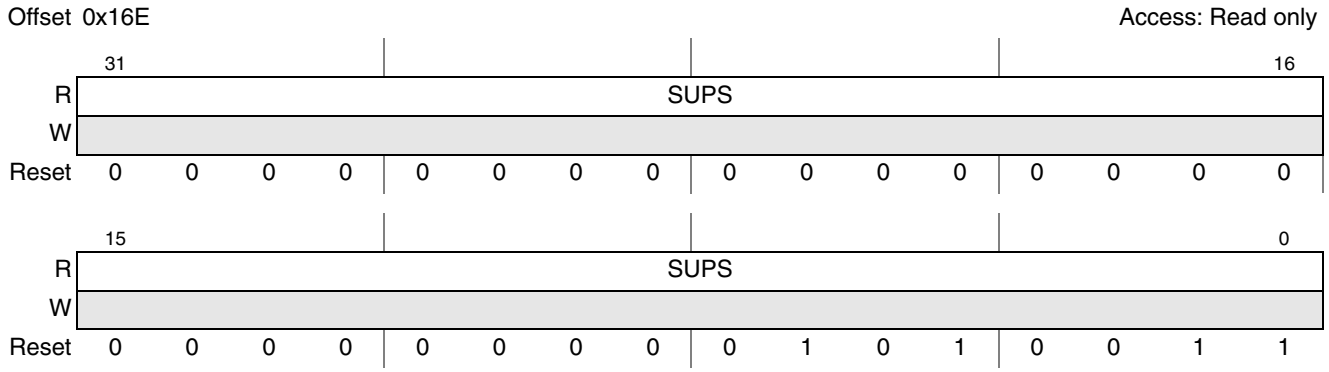


Figure 20-172. PCI Express SR-IOV Supported Page Size Register

Table 20-163. PCI Express SR-IOV Supported Page Size Register Field Description

Bits	Name	Description
31–0	SUPS	Supported Page Size.

20.3.11.28 PCI Express SR-IOV System Page Size Register (SR-IOV-only)—0x170

The PCI Express SR-IOV system page size register is shown in Figure 20-173.

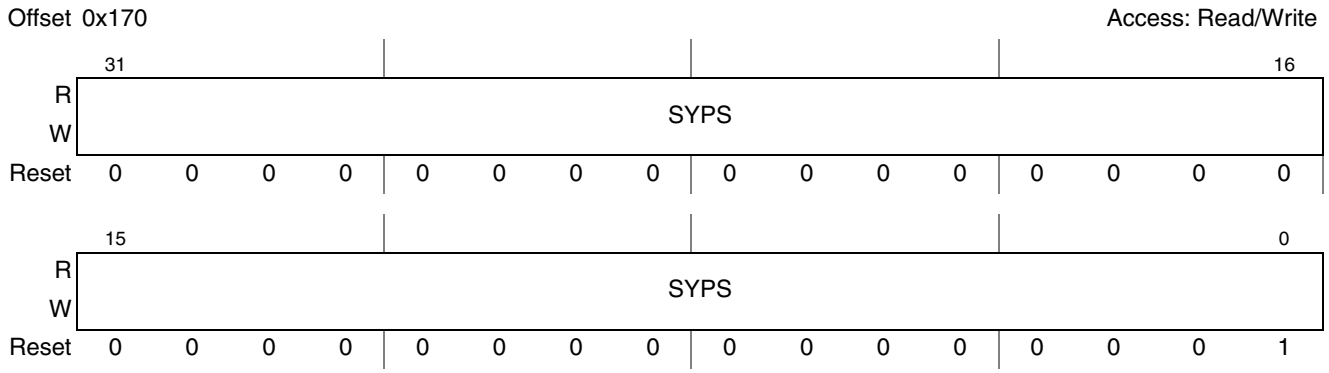


Figure 20-173. PCI Express SR-IOV System Page Size Register

Table 20-164. PCI Express SR-IOV System Page Size Register Field Description

Bits	Name	Description
31–0	SYPS	System Page Size. Defines the page size the system will use to map the VFs memory addresses.

20.3.11.28.1 PCI Express SR-IOV VF Base Address Registers (SR-IOV-only)

The PCI Express VF base address registers (VF BARs) point to the beginning of distinct address ranges which the device should claim. In EP mode, the device supports a two 32-bit memory space BARs, and two 64-bit memory space BARs.

Base address register 0 at offset 0x174 and base address sregister 1 at offset 0x178 are used to define the inbound memory windows in the 32-bit memory space. The 32-bit memory BARs are shown in Figure 20-174.

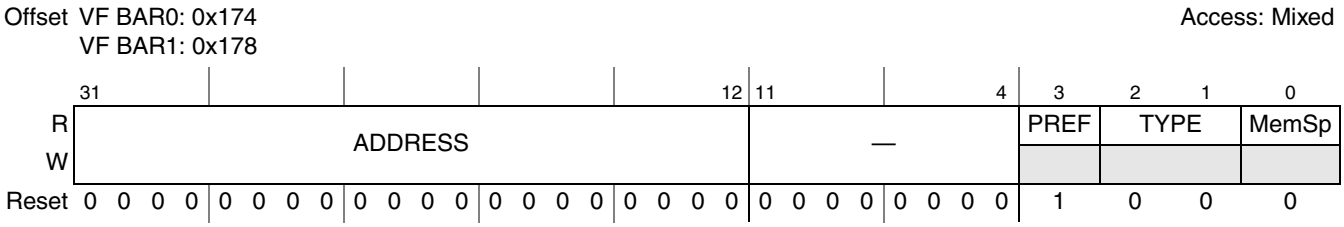


Figure 20-174. PCI Express SR-IOV VF 32-Bit Memory Base Address Register (VF BAR0,1)

Table 20-165 describes the PCI Express SR-IOV VF 32-bit memory BAR fields.

Table 20-165. PCI Express VF 32-Bit Memory Base Address Register (VF BAR0,1) Field Descriptions

Bits	Name	Description
31–12	ADDRESS	Indicates the base address where the inbound memory window begins. The number of upper bits that the device allows to be writable is selected through the inbound window size in the VF inbound window attributes register (PEXVFIWAR0,1).
11–4	—	Reserved. The device allows a 4 Kbyte window minimum.
3	PREF	Prefetchable. This bit is determined by PEXVFIWAR0,1[PF].
2–1	TYPE	Type. 00 Locate anywhere in 32-bit address space.
0	MemSp	Memory space indicator.

Base address register 2 at offset 0x17C and base address register 4 at offset 0x184 are used to define the lower portion of the 64-bit inbound memory windows. The 64-bit low memory VF BARs are shown in Figure 20-175.

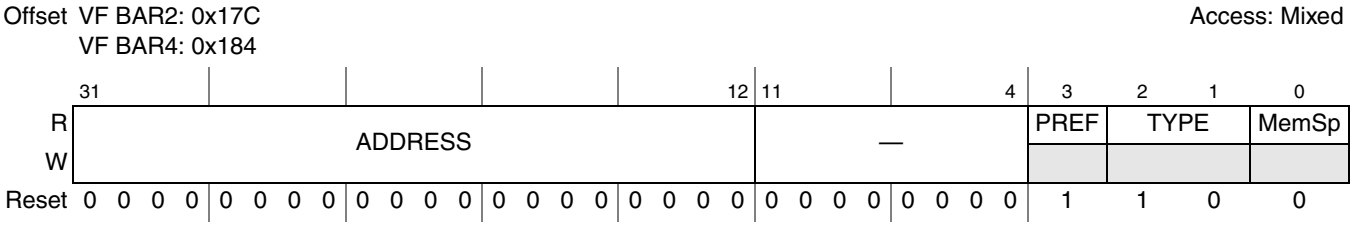


Figure 20-175. PCI Express VF 64-Bit Low Memory Base Address Register

Table 20-166 describes the PCI Express VF 64-bit low memory BAR fields.

Table 20-166. PCI Express VF 64-Bit Low Memory Base Address Register Field Descriptions

Bits	Name	Description
31–12	ADDRESS	Indicates the lower portion of the base address where the inbound memory window begins. The number of bits that the device allows to be writable is selected through the inbound window size in the inbound window attributes registers (PEXVFIWAR2 for VF BAR2 and PEXVFIWAR3 for VF BAR4).
11–4	—	Reserved. The device allows a 4 Kbyte window minimum.
3	PREF	Prefetchable. This bit is determined by PEXVFIWAR*2,3*[2].

Table 20-166. PCI Express VF 64-Bit Low Memory Base Address Register Field Descriptions (continued)

Bits	Name	Description
2–1	TYPE	Type. 0b10 Locate anywhere in 64-bit address space.
0	MemSp	Memory space indicator

Base address register 3 at offset 0x180 and base address register 5 at offset 0x188 are used to define the upper portion of the 64-bit inbound memory windows. The VF 64-bit high memory BARs are shown in Figure 20-176.

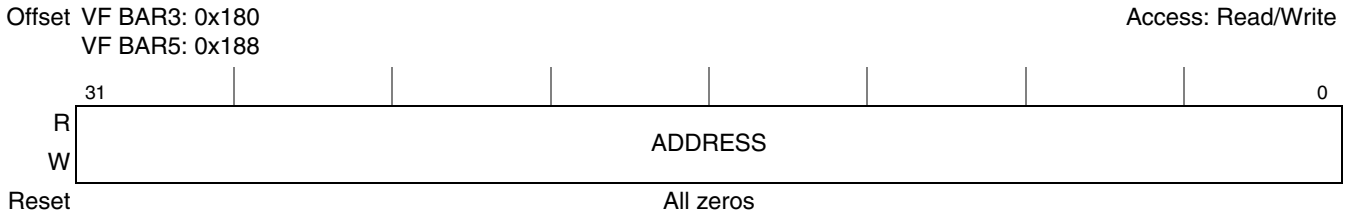


Figure 20-176. VF 64-Bit High Memory Base Address Register

Table 20-167 describes the PCI Express VF 64-bit low memory BAR fields.

Table 20-167. VF 64-Bit High Memory Base Address Register

Bits	Name	Description
31–0	ADDRESS	Indicates the upper portion of the base address where the inbound memory window begins. The number of bits that the device allows to be writable is selected through the inbound window size in the inbound window attributes registers (PEXVFIWAR2 for VF BAR3 and PEXVFIWAR3 for VF BAR5). If no access to local memory is to be permitted by external requestors, then all bits are programmed.

20.3.11.28.2 PCI Express SR-IOV Migration State Array Offset Register (SR-IOV-only)—0x18C

The PCI Express SR-IOV migration state array offset is shown in Figure 20-177.

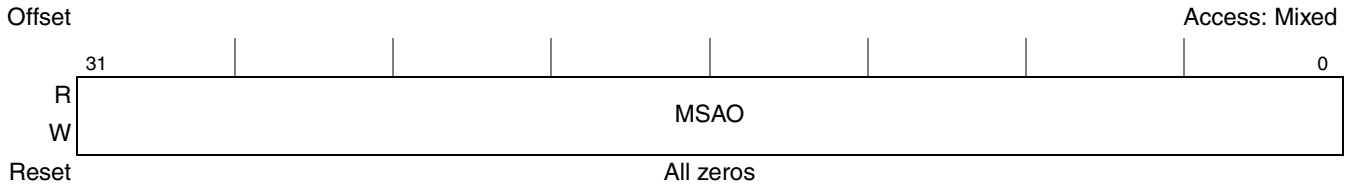


Figure 20-177. PCI Express SR-IOV Migration State Array Offset Register

Table 20-168 describes the PCI Express SR-IOV migration state array offset fields.

Table 20-168. PCI Express SR-IOV Migration State Array Offset Field Descriptions

Bits	Name	Description
31–0	MSAO	Migration state array offset

20.4 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 requestor and to the completer, the intermediate components are transparent.

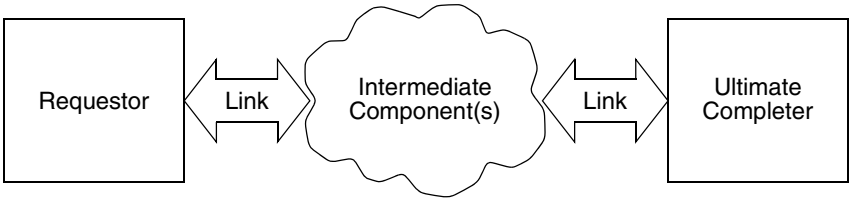


Figure 20-178. Requestor/Completer Relationship

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 Figure 20-179.

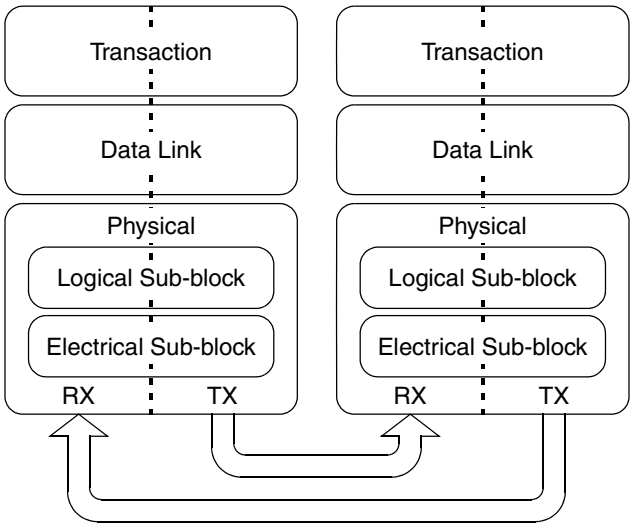


Figure 20-179. PCI Express High-Level Layering

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 Figure 20-180. 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.

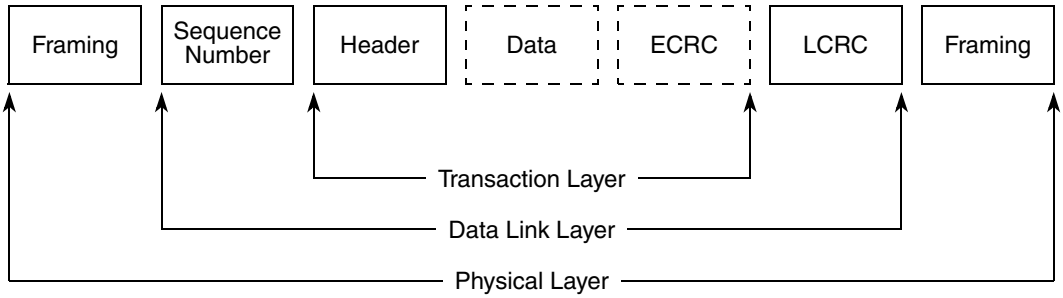


Figure 20-180. PCI Express Packet Flow

20.4.1 Architecture

This section describes implementation details of the PCI Express controller.

20.4.1.1 PCI Express Transactions

Table 20-169 contains the list of transactions that the PCI Express controller supports as an initiator and a target.

Table 20-169. 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	Configuration Read Type 0.
CfgWr0	Yes (RC only)	Yes	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. Note that Set_Slot_Power_Limit is the only message with data that is supported and then only when the controller is an initiator and in RC mode or a target and in EP mode.
Cpl	Yes	Yes	Completion without Data
CplD	Yes	Yes	Completion with Data

Table 20-169. PCI Express Transactions (continued)

PCI Express Transaction	Supported as an Initiator	Supported as a Target	Definition
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

20.4.1.2 Byte Ordering

Whenever data must cross a bridge between two busses, the byte ordering of data on the source and destination buses must be considered. The internal platform bus of this device is inherently big endian and the PCI Express bus interface is inherently little endian.

There are two methods to handle ordering of data as it crosses a bridge—address invariance and data invariance. Address invariance preserves the addressing of bytes within a scalar data element, but not the relative significance of the bytes within that scalar. Conversely, data invariance preserves the relative significance of bytes within a scalar, but not the addressing of the individual bytes that make up a scalar.

This device uses address invariance as its default byte ordering policy, but it also may be configured to use a data invariance policy. The byte ordering policy is controlled by the ATMUs using the data invariance enable parameter in the inbound and outbound window attributes registers (PEXIWAR*n*[DIEN] and PEXOWAR*n*[DIEN]). Note that by using the ATMUs to control the byte ordering policy, it is possible to allow data invariance for certain devices or regions, while mapping other devices or regions using address invariance. However, in such cases, it is important that software keep track of which devices and regions use which byte ordering policy and access them accordingly.

20.4.1.2.1 Address Invariance

As stated above, address invariance preserves the byte address of each byte on an I/O interface as it is placed in memory or moved into a register. This policy can have the effect of reversing the significance order of bytes (most significant to least significant and vice versa), but it has the benefit of preserving the format of general data structures. Provided that software is aware of the endianness and format of the data structure, it can correctly interpret the data on either side of the bridge.

Figure 20-181 shows the transfer of a 4-byte scalar, 0x4142_4344, from a big endian source across an address invariant bridge to a little endian destination.

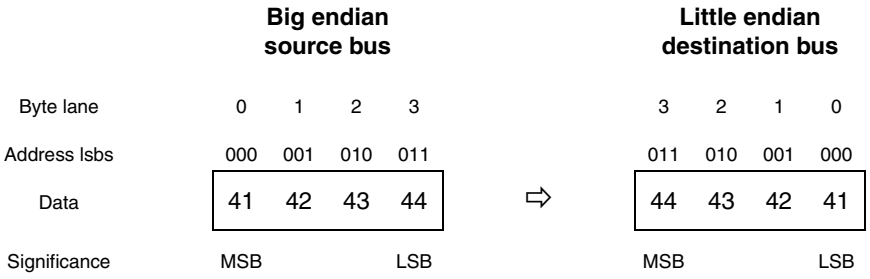


Figure 20-181. Address Invariant Byte Ordering—4 bytes Outbound

Note that although the significance of the bytes within the scalar have changed, the address of the individual bytes that make up the scalar have not changed. As long as software is aware that the source of the data used a big endian format, the data can be interpreted correctly.

Figure 20-183 shows data flowing the other way, from a little endian source to a big endian destination.

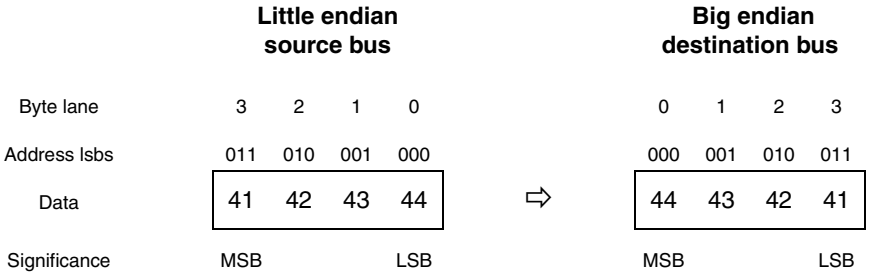


Figure 20-182. Address Invariant Byte Ordering—4 bytes Inbound

Figure 20-183 shows an outbound transfer of an 8-byte scalar, 0x5455_1617_CDCE_2728, using address invariance.

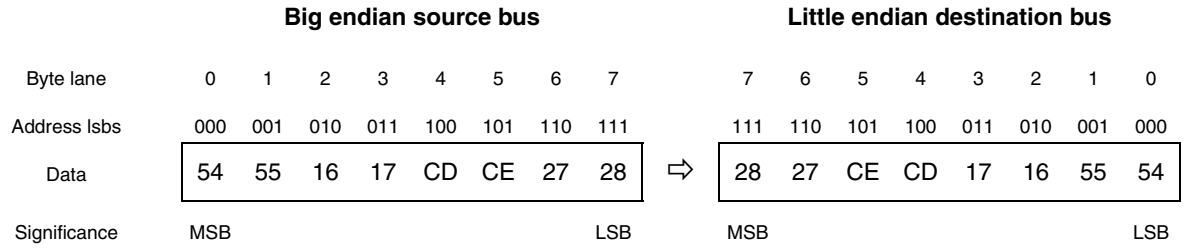


Figure 20-183. Address Invariant Byte Ordering—8 bytes Outbound

Figure 20-184 shows an inbound transfer of a 2-byte scalar, 0x5837, using address invariance.

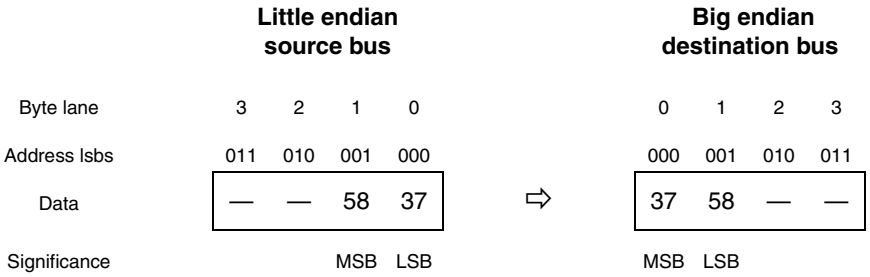


Figure 20-184. Address Invariant Byte Ordering—2 bytes Inbound

Note that in all of these examples, the original addresses of the individual bytes within the scalars (as created by the source) have been preserved.

20.4.1.2.2 Data Invariance

The alternate policy of data invariance preserves the relative significance of data by translating the byte addressing within a given boundary. The problem with data invariance is it can only preserve the significance of one size of data—in this device's case, 4-bytes. For scalars of different sizes or for data

structures employing multiple sized scalars, it has the very undesirable effect of scrambling their addresses and formats. This puts a much greater burden on software to reconstruct non-homogeneous data structures that have passed through a data invariant bridge. It also requires knowledge of all potential paths that the data may have traversed and all the data invariance translations in each of those paths.

Freescale provides the option of using data invariance on some of its devices to allow them to be used in systems employing external devices that require this policy. Note that the data invariance feature is only supported for use with specific interface cards that require this non-default byte ordering policy.

Figure 20-185 shows the transfer of a 4-byte scalar, 0x4142_4344, from a big endian source across a bridge to a little endian destination using a 4-byte data invariance policy.

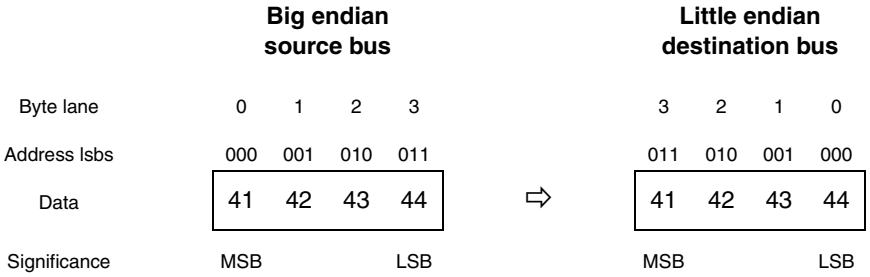


Figure 20-185. Data Invariant Byte Ordering—4 bytes Outbound

Note that the byte significance of the scalar has been preserved, even though the addresses of the individual bytes within the scalar have changed. This has effectively converted an aligned 4-byte big endian scalar to a 4-byte little endian scalar.

Figure 20-186 shows data flowing the other way, from a little endian source to a big endian destination.

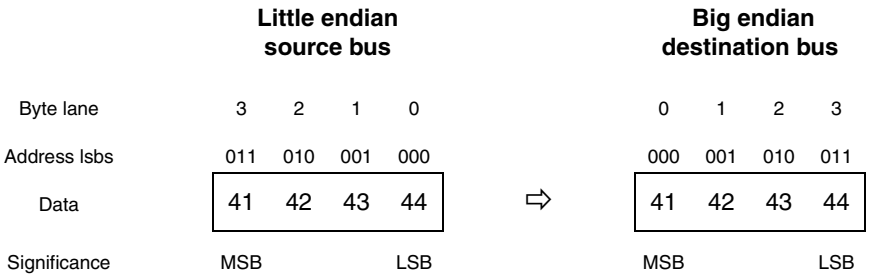


Figure 20-186. Data Invariant Byte Ordering—4 bytes Inbound

As before, the significance of the individual bytes has been preserved. It appears that the aligned 4-byte little endian scalar has been converted to a 4-byte big endian scalar. However, note what happens when a different size scalar is transferred across the bridge. Figure 20-187 shows an outbound transfer of an 8-byte scalar, 0x5455_1617_CDCE_2728, using a 4-byte data invariance policy.

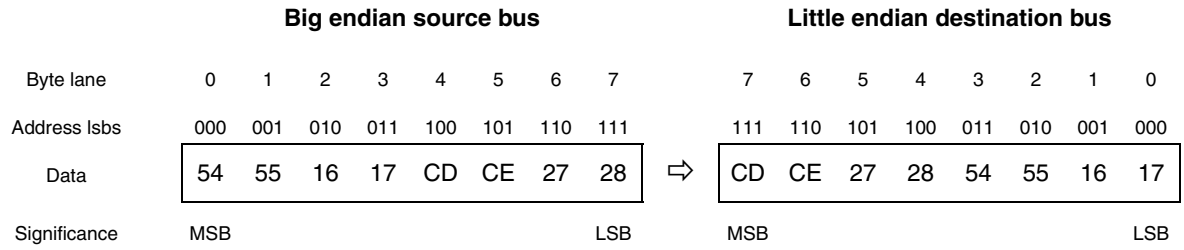


Figure 20-187. Data Invariant Byte Ordering—8 bytes Outbound

Here, the significance of bytes within the scalar has not been preserved. The data invariance has broken the scalar into halves at the 4-byte boundary. If a little endian device needs to access the 8-byte scalar, it will first need to reconstruct the data by rearranging the two halves.

Figure 20-188 shows an inbound transfer of a 2-byte scalar, 0x5837, using data invariance.

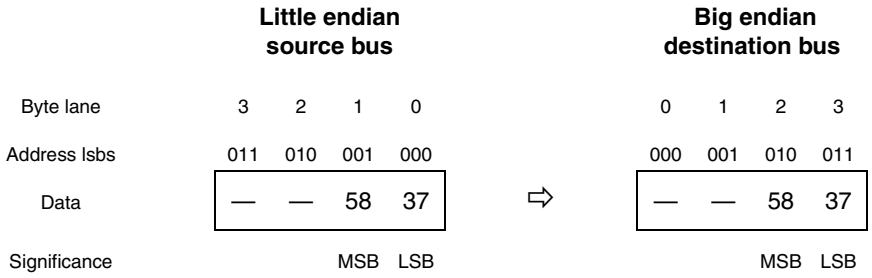


Figure 20-188. Data Invariant Byte Ordering—2 bytes Inbound

Again, the significance of the data has been preserved, but where the little endian device accesses the 2-byte scalar at an address offset 0, a big endian device will need to access it at an address offset by 2.

20.4.1.2.3 Byte Order for Configuration Transactions

All internal memory-mapped registers in the CCSR space use big endian byte ordering. However, the PCI Express specification defines PCI Express configuration registers as little endian. All accesses to the PCI Express configuration port, PEX_CONFIG_DATA, including the those targeting the internal PCI Express configuration registers, use the address invariance policy as shown in Figure 20-189. Therefore, software must access PEX_CONFIG_DATA with little-endian formatted data—either using the **lwbrx/stwbrx** instructions or by manipulating the data before writing to and after reading from PEX_CONFIG_DATA.

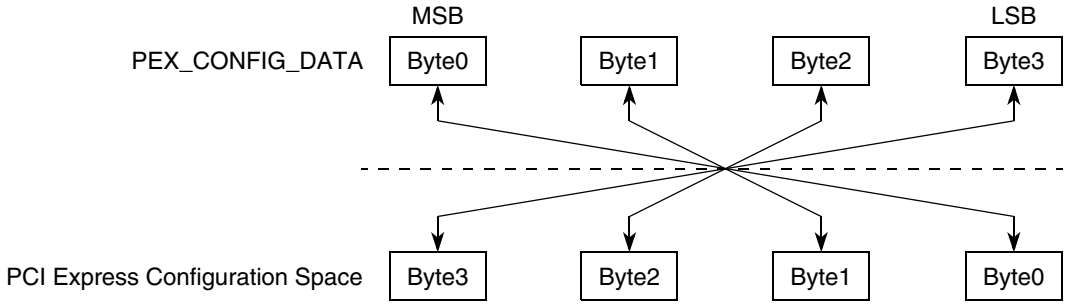


Figure 20-189. PEX_CONFIG_DATA Byte Ordering

20.4.1.3 Lane Reversal

PCI Express lane reversal is supported as part of the PCI Express training operations where the lanes are reversed between the two chips at the end of a PCI Express link. This can occur because of chip packaging, pinning or board placement. Lane reversal is performed automatically as part of the lane-numbering negotiation after the link-width negotiation is complete.

In addition, manual lane reversal is supported to manually reverse lanes in addition to LTSSM automatic lane reversal operation. The manual lane reversal is useful in the following usage scenarios:

- Connecting a narrow upstream port to the upper lanes of a wider downstream port
- Forming a link with broken and reversed lanes

20.4.1.4 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 are the ordering rules for sending the next outstanding request:

- A posted request can and bypasses all other transactions except another posted request.
- A completion can and only bypasses non-posted. It can and bypasses posted requests only if the relaxed ordering (RO) bit is set.
- 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.

20.4.1.5 Memory Space Addressing

A PCI Express memory transaction can address a 32- or 64-bit memory space. The FMT[0] field in the PCI Express TLP header for a 32-bit address packet is 0; a 64-bit address packet has a FMT[0] = 1. The PCI Express TLP header for a memory read transaction has TYPE[4:0] = 00000 and FMT[1] = 0. A memory write transaction has TYPE[4:0] = 00000 and FMT[1] = 1. 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 36-bit internal platform addresses.

20.4.1.6 I/O Space Addressing

The controller does not support I/O transactions as a target. 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. All I/O transactions only access 32-bit address I/O space. The PCI Express TLP header for an I/O read transaction has TYPE[4:0] = 00010 and FMT[1] = 0. The PCI Express TLP header for an I/O write transaction has TYPE[4:0] = 00010 and FMT[1] = 1.

20.4.1.7 Configuration Space Addressing

As an initiator, the controller supports both type 0 and type 1 configuration cycles when configured in RC mode. There are two methods of generating a configuration transaction; refer to Section 20.3.8, "PCI Express Configuration Space Access," for more information. A configuration transaction can hit into the controller's internal configuration space, it can be sent out on the PCI Express link, or it can be internally terminated. The PCI Express TLP header for a type 0 configuration read transaction has TYPE[4:0] = 00100 and FMT[1] = 0; the PCI Express TLP header for a type 0 configuration write transaction has TYPE[4:0] = 00100 and FMT[1] = 1. The PCI Express TLP header for a type 1 configuration read transaction has TYPE[4:0] = 00101 and FMT[1] = 0; the PCI Express TLP header for a type 1 configuration write transaction has TYPE[4:0] = 00101 and FMT[1] = 1. Note that all configuration transactions sent on PCI Express require a response regardless whether they are read or a write configuration transactions.

The controller does not generate configuration transactions in EP mode. Only inbound configuration transactions are supported in EP mode.

20.4.1.8 Serialization of Configuration and I/O Writes

Configuration and I/O writes are serialized by the controller. The logic after issuing a configuration write or IO write does not issue any new transactions until the outstanding configuration or I/O write is finished. This means that an acknowledgement packet from the link partner in the form of a CpL TLP packet must be seen or the transaction has timed out. If the CpL packet contains a CRS status, then the logic re-issues the configuration write transaction. It keeps retrying the request until either a status other than CRS is returned or the transaction times out.

Note that it is possible for outbound configuration read request to be requeued and be placed at the end of the request queue due to CRS condition.

20.4.1.9 Messages

Software message generation is supported in both RC and EP modes.

20.4.1.9.1 Outbound ATMU Message Generation

Software can choose to send a message by programming PEXOWAR*n*[WTT] = 0x5. A message is sent by writing a 4-byte transaction in big-endian format that hits in an outbound window configured to send messages.

Part of the 4-byte data is used to store information such as message code and routing information. The data format is slightly different depending on whether data invariance mode is enabled. Table 20-170 describes the message data format in address invariance mode (PEXOWAR*n*[DIEN] = 0).

Table 20-170. Internal Platform Message Data Format—Address Invariance Mode

Bits	Name	Reset Value	Description
0–15	—	—	Reserved
16–18	Routing	x	Routing mechanism. Contains the message's routing information
19–23	—	—	Reserved
24–31	Code	x	Message code. Contains the actual message type to be sent.

Table 20-171 describes the message data format in data invariance mode (PEXOWAR*n*[DIEN] = 1).

Table 20-171. Internal Platform Message Data Format—Data Invariance Mode

Bits	Name	Reset Value	Description
0–7	Code	x	Message code. Contains the actual message type to be sent.
8–10	Routing	x	Routing mechanism. Contains the message's routing information
11–31	—	—	Reserved

In addition to the outbound ATMU, the PEX PM Command register also provides the capability to send PME_Turn_Off message or PM_PME message by setting bits 31 or 29. See Section 20.3.3.4, "PCI Express Power Management Command Register (PEX_PMCR)," on page 20-27 for more information.

Table 20-172 provides a complete list of supported outbound messages depending on whether RC or EP is configured.

Table 20-172. PCI Express ATMU Outbound Messages

Name	Code[7:0]	Routing[2:0]	RC	EP	Description
PM_Active_State_Nak	0001 0100	100	Yes	N/A	Terminate at receiver
PM_PME	0001 1000	000	N/A	Yes	Sent Upstream by PME-requesting component
PME_Turn_Off	0001 1001	011	Yes	N/A	Broadcast Downstream
PM_TO_Ack	0001 1011	101	N/A	Yes	Sent Upstream by Endpoint
ERR_COR	0011 0000	000	N/A	Yes	Sent by component when it detects a correctable error
ERR_NONFATAL	0011 0001	000	N/A	Yes	Sent by component when it detects a Non-fatal, uncorrectable error
ERR_FATAL	0011 0011	000	N/A	Yes	Sent by component when it detects a Fatal, uncorrectable error
Unlock	0000 0000	011	No	N/A	Not supported

Table 20-172. PCI Express ATMU Outbound Messages (continued)

Name	Code[7:0]	Routing[2:0]	RC	EP	Description
Set_Slot_Power_Limit	0101 0000	100	No	N/A	Set Slot Power Limit in Upstream Port Note that although it is not supported via ATMU, it can still be sent via PCI Express configuration register.
Vendor_Defined Type 0	0111 1110		No	No	Not supported
Vendor_Defined Type 1	0111 1111		No	No	Not supported

20.4.1.9.2 Inbound Messages

Table 20-173 provides a complete list of supported inbound messages in RC mode.

Table 20-173. PCI Express RC Inbound Message Handling

Name	Code[7:0]	Routing[2:0]	Action
Assert_INTA	0010 0000	100	Send to PIC
Assert_INTB	0010 0001	100	Send to PIC
Assert_INTC	0010 0010	100	Send to PIC
Assert_INTD	0010 0011	100	Send to PIC
De-assert_INTA	0010 0100	100	Send to PIC
De-assert_INTB	0010 0101	100	Send to PIC
De-assert_INTC	0010 0110	100	Send to PIC
De-assert_INTD	0010 0111	100	Send to PIC
PM_Active_State_Nak	0001 0100	100	No action taken
PM_PME	0001 1000	000	Generate interrupt to PIC if enabled
PME_Turn_Off	0001 1001	011	No action taken
PME_TO_Ack	0001 1011	101	Set PEX_PME_MES_DR[ENL23] bit and generate interrupt to PIC if enabled
ERR_COR	0011 0000	000	Generate interrupt to PIC if enabled
ERR_NONFATAL	0011 0001	000	Generate interrupt to PIC if enabled
ERR_FATAL	0011 0011	000	Generate interrupt to PIC if enabled
Unlock	0000 0000	000	No action taken
Set_Slot_Power_Limit	0101 0000	100	No action taken
Vendor_Defined Type 0	0111 1110		No action taken
Vendor_Defined Type 1	0111 1111		No action taken

Table 20-174 provides a complete list of supported inbound messages in EP mode.

Table 20-174. PCI Express EP Inbound Message Handling

Name	Code[7:0]	Routing[2:0]	Action
Assert_INTA	0010 0000	100	No action taken
Assert_INTB	0010 0001	100	No action taken
Assert_INTC	0010 0010	100	No action taken
Assert_INTD	0010 0011	100	No action taken
Deassert_INTA	0010 0100	100	No action taken
Deassert_INTB	0010 0101	100	No action taken
Deassert_INTC	0010 0110	100	No action taken
Deassert_INTD	0010 0111	100	No action taken
PM_Active_State_Nak	0001 0100	100	No action taken
PM_PME	0001 1000	000	No action taken
PME_Turn_Off	0001 1001	011	Set PEX_PME_MES_DR[PTO] bit. Send interrupt if enabled.
PM_TO_Ack	0001 1011	101	No action taken
ERR_COR	0011 0000	000	No action taken
ERR_NONFATAL	0011 0001	000	No action taken
ERR_FATAL	0011 0011	000	No action taken
Unlock	0000 0000	000	No action taken
Set_Slot_Power_Limit	0101 0000	100	Update power value in PCI Express device capability register in configuration space.
Vendor_Defined Type 0	0111 1110		No action taken
Vendor_Defined Type 1	0111 1111		No action taken

20.4.1.10 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 Figure 20-190 uncorrectable errors can further be classified as fatal or non-fatal.

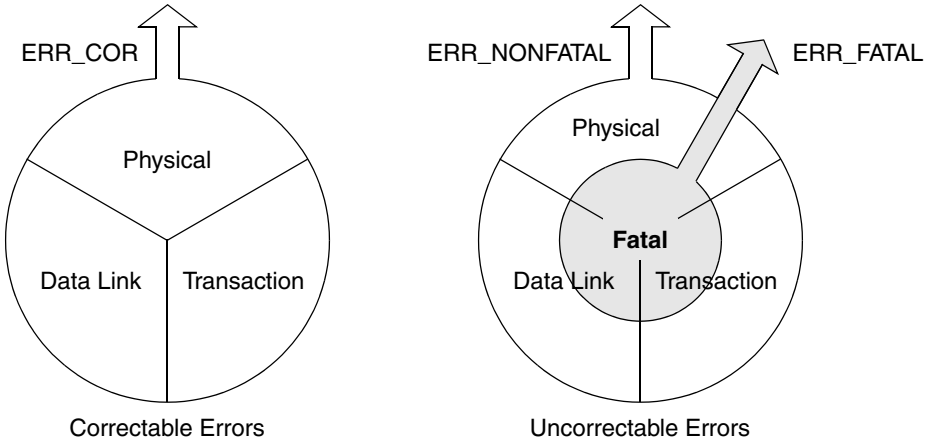


Figure 20-190. PCI Express Error Classification

20.4.1.10.1 PCI Express Error Logging and Signaling

Figure 20-191 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.

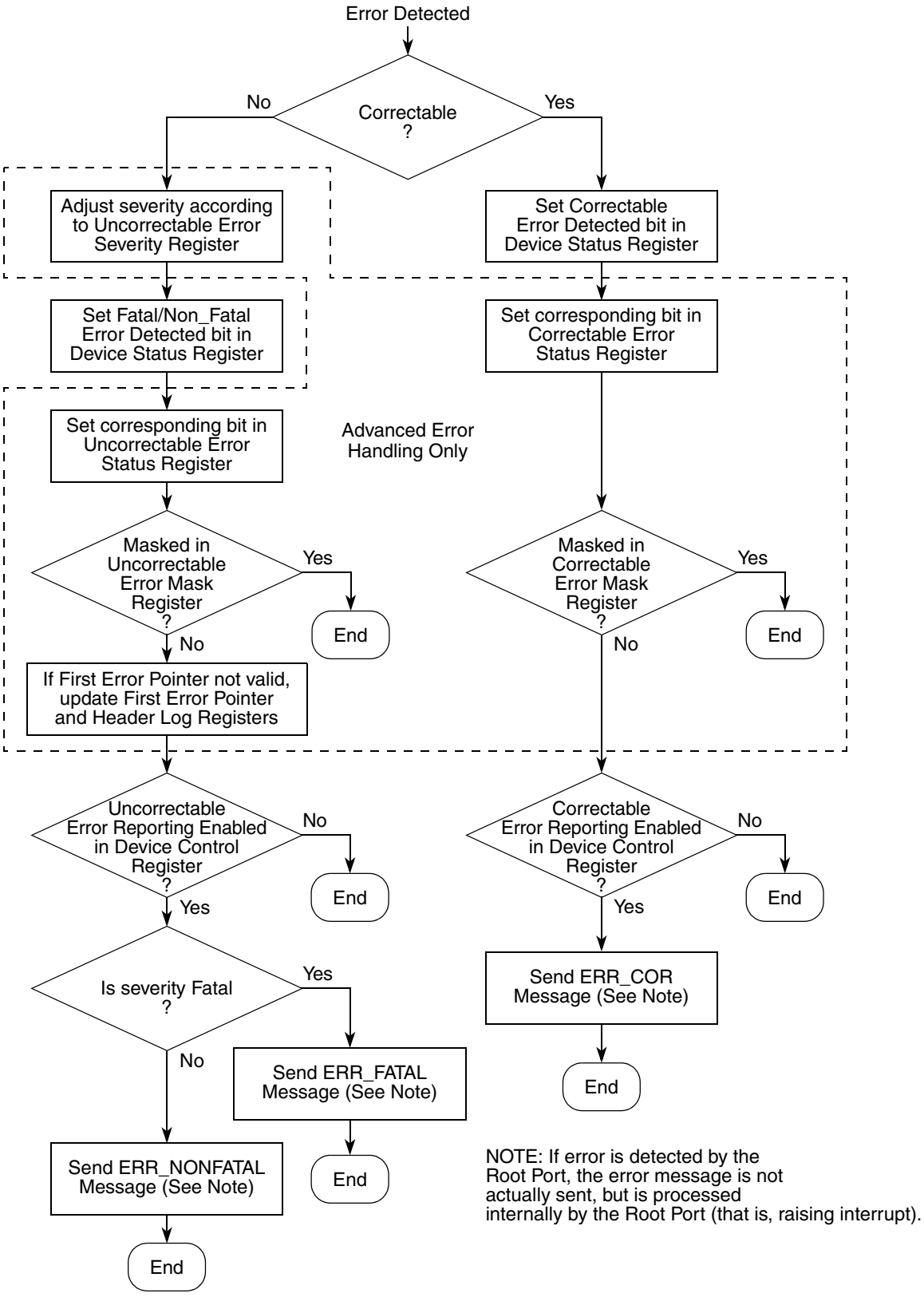


Figure 20-191. PCI Express Device Error Signaling Flowchart

20.4.1.10.2 PCI Express Controller Internal Interrupt Sources

Table 20-175 describes the sources of the PCI Express controller internal interrupt to the PIC and the preconditions for signaling the interrupt.

Table 20-175. PCI Express Internal Controller Interrupt Sources

Status Register Bit	Preconditions
Any bit in PEX_PME_MES_DR set	The corresponding interrupt enable bits must be set in PEX_PME_MES_IER
Any bit in PEX_ERR_DR set	The corresponding interrupt enable bits must be set in PEX_ERR_EN.
PCI Express Root Status Register[16] (PME status) is set	The INTx Assertion Disable bit in the Command register is 0 and the PME Interrupt Enable bit in the Root Controlr egister is set to 1 and the PME Status bit in the Root Status register is set to 1.
Any bit in PCI Express Root Error Status Register set	A reported error condition causes a bit to be set in the Root Error Status register and the associated error message reporting enable bit is set in the Root Error Command register.
Any bit in PCI Express Secondary Status Register bit set	Any bits are set in the Secondary Status register.
Any bit in PCI Express Slot Status Register bit set	The INTx Assertion Disable bit in the Command register is 0 and Hot-Plug interrupts are enabled in the Slot Control register and any bit in the Slot Status register is equal to 1, and the associated event modification is enabled in the Slot Control register.
PCI Express Link Autonomous Bandwidth Status register (Link Status register bit 15) is set	Link Autonomous Bandwidth Status register (Link Status register bit 15) is updated and the Link Autonomous Bandwidth Interrupt Enable (Link Control register bit 11) is set.
Link Bandwidth Management Status register (Link Status register bit 14) is set	Link Bandwidth Management Status register (Link Status register bit 14) is updated and the Link Bandwidth Management Interrupt Enable (Link Control register bit 10) is set.
Link Equalization Request bit in the Link Status 2 register is set.	Link Equalization Request bit in the Link Status 2 register has been set.

20.4.1.10.3 Error Conditions

Table 20-176 describes specific error types and the action taken for various transaction types.

Table 20-176. Error Conditions

Transaction Type	Error Type	Action
Inbound response	PEX response time out. This case happens when the internal platform sends a non-posted request that did not get a response back after a specific amount of time specified in the outbound completion timeout register (PEX_OTB_CPL_TOR)	Log error (PEX_ERR_DR[PCT]) and send interrupt to PIC, if enabled.

Table 20-176. Error Conditions (continued)

Transaction Type	Error Type	Action
Inbound response	Unexpected PEX response. This can happen if, after the response times out and the internal queue entry is deallocated, the response comes back.	Log unexpected completion error (PCI Express Uncorrectable Status Register[16]) and send interrupt to PIC, if enabled.
Inbound response	Unsupported request (UR) response status	Depending upon whether the initial internal request was broken up, the error is not sent until all responses come back for all portions of the internal request. Log the error (PEX_ERR_DR[CDNSC] and PCI Express Uncorrectable Status Register[20]) and send interrupt to PIC, if enabled.
Inbound response	Completer abort (CA) response status	Depending upon whether the initial internal request was broken up, the error is not sent until all responses come back for all portions of the internal request. Log the error (PEX_ERR_DR[PCAC, CDNSC] and PCI Express Uncorrectable Status Register[15] and send interrupt to PIC, if enabled.
Inbound response	Unsupported Request (UR) response status in EP mode	Depending upon whether the initial internal request was broken up, the error is not sent until all responses come back for all portions of the internal request. Log the error (PEX_ERR_DR[UREP] and send interrupt to PIC, if enabled.
Inbound response	Poisoned TLP (EP=1)	Depending upon whether the initial internal request was broken up, the error is not sent until all responses come back for all portions of the internal request. Log the error (PCI Express Uncorrectable Status Register[12]) and send interrupt to PIC, if enabled.
Inbound response	ECRC error	Depending upon whether the initial internal request was broken up, the error is not sent until all responses come back for all portions of the internal request. Log the error (PCI Express Uncorrectable Status Register[19]) and send interrupt to PIC, if enabled.
Inbound response	Configuration Request Retry Status (CRS) timeout for a configuration transaction that originates from PEX_CONFIG_ADDR/ PEX_CONFIG_DATA	1.The controller always retries the transaction as soon as possible until a status other than CRS is returned. However, if a CRS status is returned after the configuration retry timeout (PEXCONF_RTY_TOR) timer expires, then the controller aborts the transaction and sends all 1s (0xFFFF_FFFF) data back to requester. 2. Log the error (PEX_ERR_DR[PCT]) and send interrupt to the PIC, if enabled.
Inbound response	UR response for configuration transaction that originates from PEX_CONFIG_ADDR/ PEX_CONFIG_DATA	1. Send back all 1s (0xFFFF_FFFF) data. 2. Log the error (PEX_ERR_DR[CDNSC] and PCI Express Uncorrectable Status Register[20]) and send interrupt to PIC, if enabled.
Inbound response	CA response for Configuration transaction that originates from PEX_CONFIG_ADDR/ PEX_CONFIG_DATA	1. Send back all 1s (0xFFFF_FFFF) data. 2. Log the error (PEX_ERR_DR[PCAC, CDNSC] and PCI Express Uncorrectable Status Register[15]) and send interrupt to PIC, if enabled.

Table 20-176. Error Conditions (continued)

Transaction Type	Error Type	Action
Inbound response	Poisoned TLP (EP=1) response for Configuration transaction that originates from PEX_CONFIG_ADDR/ PEX_CONFIG_DATA	1. Send back all 1s (0xFFFF_FFFF) data. 2. Log the error (PCI Express Uncorrectable Status Register[12]) and send interrupt to PIC, if enabled.
Inbound response	ECRC error response for Configuration transaction that originates from PEX_CONFIG_ADDR/ PEX_CONFIG_DATA	1. Send back all 1s (0xFFFF_FFFF) data. 2. Log the error (PCI Express Uncorrectable Status Register[19]) and send interrupt to PIC, if enabled.
Inbound response	Configuration Request Retry Status (CRS) response for Configuration transaction that originates from ATMU	1.The controller always retries the transaction as soon as possible until a status other than CRS is returned. However, if a CRS status is returned after the configuration retry timeout (PEXCONF_RTY_TOR) timer expires, then the controller aborts the transaction. 2. Log the error (PEX_ERR_DR[CRST]) and send interrupt to the PIC, if enabled.
Inbound response	UR response for Configuration transaction that originates from ATMU	Log the error (PEX_ERR_DR[CDNSC] and PCI Express Uncorrectable Status Register[20]) and send interrupt to PIC, if enabled.
Inbound response	CA response for Configuration transaction that originates from ATMU	Log the error (PEX_ERR_DR[PCAC, CDNSC] and PCI Express Uncorrectable Status Register[15]) and send interrupt to PIC, if enabled.
Inbound response	Malformed TLP response	PCI Express controller does not pass the response back to the core. Therefore, a completion timeout error eventually occurs.
Inbound request	Poisoned TLP (EP=1)	1. If it is a posted transaction, the controller drops it. 2. If it is a non-posted transaction, the controller returns a completion with UR status. 3. Release the proper credits
Inbound request	ECRC error	1. If it is a posted transaction, the controller drops it. 2. If it is a non-posted transaction, the controller returns a completion with UR status. 3. Release the proper credits
Inbound request	PCI Express nullified request	The packet is dropped.
Inbound request	No map	1. Log the error(PEX_ERR_DR[NM]) and send interrupt to PIC if enable. 2. If it is a non-posted transaction, the controller sends a completion with UR. If it is a posted transaction, the controller drops it.
Inbound request	LIODN mapping error	1. Set PEX_ERR_DR[LDDE] and send interrupt if enabled. 2. Transaction is dropped if posted and UR if non-posted.
Outbound request	Outbound ATMU crossing	Log the error (PEX_ERR_DR[OAC]). The transaction is not sent out on the link.
Outbound request	Illegal message size	Log the error (PEX_ERR_DR[MIS]). The transaction is not sent out on the link.
Outbound request	Illegal I/O size	Log the error (PEX_ERR_DR[IOIS]). The transaction is not sent out on the link.

Table 20-176. Error Conditions (continued)

Transaction Type	Error Type	Action
Outbound request	Illegal I/O address	Log the error (PEX_ERR_DR[IOIA]). The transaction is not sent out on the link.
Outbound request	Illegal configuration size	Log the error (PEX_ERR_DR[CIS]). The transaction is not sent out on the link.
Outbound request	Memory out of range	Log the error (PEX_ERR_DR[IMBA]). The transaction is not sent out on the link.
Outbound request	IO out of range	Log the error (PEX_ERR_DR[IIOBA]). The transaction is not sent out on the link.
Outbound request	Link down	Log the error (PEX_ERR_DR[LDDE]). The transaction is not sent out on the link.
Outbound request	FLR	Log the error (PEX_ERR_DR[FLRDE]). The transaction is not sent out on the link.
Outbound request	Tx Abort	Log the error (PEX_ERR_DR[TAE]). The transaction is not sent out on the link.
Outbound response	Internal platform response with error (for example, an ECC error on a DDR read or the transaction maps to unknown address space).	Send poisoned TLP (EP=1) completion(s) for data that are known bad. If the poison data happens in the middle of the packet, the rest of the response packet(s) is also poisoned.
Link speed change request	1. Auto request with Auto speed disable bit set or 2. Link partner incapable of 5.0 Gb/s and request speed is 5.0 Gb/s	Clear link speed request and set interrupt in PME Message Detect register. Additionally, log error in link speed status register.
Link width change request	1. Auto request with Auto width disable bit set or 2. Upconfiguration not capable and request was to size-up the width or 3. Not enough detected lanes for a given request	Clear link width request and set interrupt in PME Message Detect register. Additionally, log error in link width status register.

20.4.2 Interrupts

Both INTx and message signaled interrupts (MSI) are supported; however there are differences depending on whether the PCI Express controller is configured as an RC or EP device.

20.4.2.1 EP Interrupt Generation

20.4.2.1.1 Hardware INTx Message Generation

Hardware INTx message generation is not supported in EP mode.

20.4.2.1.2 Hardware MSI Generation

In EP mode, the PCI Express controller can be configured to automatically generate MSI transactions to the root complex when an interrupt event occurs.

The PCI Express controller uses *sie1* (an internal signal) to trigger the generation of the MSI. To trigger the MSI, interrupt events must be routed to the appropriate *sie1* by setting the INTTGT field to 0xF1 in the associated Interrupt Level register in the MPIC. Note that sie1 is sent to all PCI Express controllers.

The remote root complex is expected set up the MSI capability structure of all endpoints at system initialization by filling the Message Address and Message Data registers with appropriate values and setting the MSIE bit in the MSI Message Control register.

With the PCI Express controller properly configured, when it detects the leading edge of *sie1* going active, it generates a PCI Express memory write transaction to the address specified in the MSI Message Address register (and MSI Message Upper Address register) with a data payload as specified in the MSI Message Data register (with leading zeros appended).

20.4.2.1.3 Software MSI Generation

Host software has to set up the MSI capability registers to enable MSI mode, and have the correct values for the MSI address and data register.

Then local software has to read the MSI address in the MSI capability register and configure the outbound ATMU window to map the translated address to the MSI address. Software has to determine the number of allocated messages in the MSI capability register and allocates the appropriate data values to use. A write to the ATMU window containing the MSI address with the appropriate data value generates the desired MSI transaction to the remote RC.

20.4.2.2 RC Handling of INTx Message and MSI Interrupts

20.4.2.2.1 INTx Message Handling

MSIs are the preferred interrupt signaling mechanism for PCI Express. However, in RC mode, the PCI Express controller supports the INTx virtual-wire interrupt signaling mechanism (as described in the PCI Express specification).

Whenever the controller receives an inbound INTx (INTA, INTB, INTC, or INTD) asserted or negated message, it asserts or negates an equivalent internal INTx signal (*inta*, *intb*, *intc*, or *intd*) to the interrupt controller (MPIC).

The internal INTx signals from the PCI Express controller may be logically combined with the external interrupt request (IRQ*n*) signals so that they share the same interrupt controlled by the associated EIVPRn and EIDR*n* registers in the PIC. Refer to PCI Express INTx/External Interrupt IRQn Sharing for more information about sharing of PCI Express INTx interrupts and the IRQ*n* signals.

If a PCI Express INT*x* interrupt is being used, then the MPIC must be configured so that external interrupts are level-sensitive (EIVPRn[S] = 1).

20.4.2.2.2 MSI Handling

An inbound MSI cycle must hit into the PEXCSRBAR window with the address offset that points to the MSIIR register in the PIC.

Note that it is the responsibility of the host software to configure each EP's MSI capability registers such that an MSI cycle generated from the EP device is routed to the MSIIR register in the PIC and for the appropriate interrupt to be generated to the core.

20.4.3 MSI-X Operations

MSI-X defines a separate optional extension to basic MSI functionality. Compared to MSI, MSI-X supports a larger maximum number of vectors per function, the ability for software to control aliasing when fewer vectors are allocated than requested, plus the ability for each vector to use an independent address and data value, specified by a MSI-X vector table that resides in PCI Express memory space.

Eight MSI-X vectors are supported for each PF or VF. With two PF and 64 VF for each PF, there is a total of 16 MSI-X vectors for PF only (2 PFs × 8 vectors) and 1024 MSI-X vectors for VFs (64 VFs × 2 PFs × 8 vectors).

The MSI-X vector table structure is defined by the *PCI Express Base Specification, Revision 3.0*. Each MSI-X vector entry is 16 bytes long which consists of four bytes for vector control, four bytes for message data, and eight bytes for message address. The MSI-X vector table entry is summarized in the Table 20-177.

Table 20-177. MSI-X Vector Table Entry Definition

Byte	Bit	Field	Meaning
3:0		Message Address	System-specified PCI Express message address. For MSI-X messages, the contents of this field from an MSI-X Table entry specifies the lower portion of the DWORD-aligned address (AD[31::02]) for the memory write transaction. AD[1::0] must already be 0. This field is read/write.
7:4		Message Upper Address	System-specified message upper address bits. If this field is zero, Single Address Cycle (SAC) messages are used. If this field is non-zero, Dual Address Cycle (DAC) messages are used. This field is read/write.
11:8		Message Data	System-specified message data. For MSI-X messages, the contents of this field from an MSI-X Table entry specifies the data driven on AD[31::00] during the memory write transaction's data phase. C/BE[3::0]# are asserted during the data phase of the memory write transaction. In contrast to message data used for MSI messages, the low-order message data bits in MSI-X messages are not modified by the function. This field is read/write.
15:12	31:1	Reserved	After reset, the state of these bits must be 0.
	0	Mask Bit	When this bit is set, the function is prohibited from sending a message using this MSI-X Table entry. However, any other MSI-X Table entries programmed with the same vector will still be capable of sending an equivalent message unless they are also masked. This bit's state after reset is 1 (entry is masked). This bit is read/write.

The MSI-X PBA is also defined in the *PCI Express Base Specification, Revision 3.0*. Each PBA entry is with QWORD length corresponding to the pending status of 64 MSI-X vector interrupt.

Both MSI-X vector table and PBA structure can be accessed by the root complex through memory read/write transactions. By default, the BAR indicator register (BIR) in the MSI-X capability structure indicates BAR1 of each function is used to access the MSI-X vector table structure as well as the MSI-X pending bt array (PBA) structure as residing in PCI Express memory space. The PBA structure is right on top of the MSI-X vector table.

20.4.3.1 Enable MSI-X Operation

At reset, MSI-X is disabled with MSI-X enable bit in the MSI-X message control register cleared. System software can set the MSI-X enable bit to enable MSI-X operation. For each function, MSI and MSI-X operation can not be enabled at the same time.

When the PEX controller is configured in EP mode, either MSI or MSI-X operation is supported (but not both). When the PEX controller is configured in RC mode, only MSI operation is supported.

20.4.3.2 MSI-X Configuration

Depending upon system software policy, system software, device driver software, or each at different times or environments may configure a function's MSI-X capability and table structures with suitable vectors. Prior to enabling MSI-X operation, device driver needs to initialize the Table Offset and PBA Offset for all PFs. Recommended offset settings are shown in the following table:

PCI Express address assignment (lower 13-bit)	PF/VF assignment	Programmed Offset (lower 16-bit)
0x0000-0x0007F	Based on BAR1 hit (PF0,VF1:64, PF1,VF1:64)	Table Offset = 0x0000
0x1000-0x100F	Bsed on BAR1 hit (PF0,VF1:64, PF1,VF1:64)	PBA Offset = 0x0200

Figure 20-192. MSI-X Table/PBA Offset Assignment

Note that the upper bit offset setting of Table/PBA should be programmed to match the BIR base address (BAR1). In addition, device driver also needs to program the PEXIWAR1[IWS]/PEXVFIWAR1 size to 8 Kbytes for accessing MSI-X vector/PBA tables. Note that there are only 8 vectors per PF/VF, accessing the lower 4K MSI-X vector/PBA table that do not fall within the 8 vector entries will alias.

Software reads the Table Size field from the Message Control register to determine the MSI-X Table size. Software calculates the base address of the MSI-X Table by reading the 32-bit value from the Table Offset/Table BIR register. Software calculates the base address of the MSI-X PBA using the same process with the PBA Offset/PBA BIR register. For each MSI-X Table entry that will be used, software fills in the MSI-X support Function Masking in MSI-X Message Control register.

When the MSI-X Function Mask bit is set, all of the function's entries must behave as being masked, regardless of the per-entry Mask bit states. "Per-vector masking" in MSI-X is controlled by a Mask bit in each MSI-X Table entry.

20.4.3.3 Send MSI-X Transaction

To request MSI-X Send service operation using a given MSI-X Table entry in an EP, Software performs a memory write to a designated MSI-X trap address window enabled by PEXMSIX_TOWAR[EN] register bit.

The MSI-X trap window is defined by a 4K address range, within which any of the 64-Byte aligned address offset can be written to initiate a MSI-X send operation. The MSI-X trap window is defined per PF. To send a VF MSI-X transaction, software should perform arbitration to ensure that the proper VF MSI-X message is being sent, especially when there are multiple threads involved.

When a 4-byte memory write transaction to any of the 64-byte aligned address offset with a specific transaction size within the MSI-X trap window is received by the PEX controller, it is interpreted as a MSI-X trap operation. Hardware will trigger scheduling of a MSI-X transaction for the corresponding MSI-X vector. The 4 bytes byte of the memory write data contains the MSI-X vector information to be used for the MSI-X transaction with the rest of the memory write data ignored by the PEX controller:

Table 20-178. PEXMSIX Trap Data (4-MSB Byte) Field Descriptions

Bits	Name	Description
0	TYPE	0, trigger MSI-X operation for PF of EP with SR-IOV 1, trigger MSI-X operation for VF of EP with SR-IOV
1–9	—	Reserved.
10-15	VF	Virtual Function Number.
16-20	—	Reserved.
21-23	IDX	MSI-X Entry Index in each PF or VF.
24-29	—	Reserved.
30-31	OP	00, trigger the Send operation for the corresponding MSI-X vector. 01, clear the Pending status bit for the corresponding MSI-X vector. This operation is used by local software to clear a pending transaction while a vector is masked. Clearing a Pending MSI-X transaction while the vector is unmasked has undefined results. 10-11, Reserved

A write transaction to MSI-X trap window will be undefined for any of the following conditions:

- PEXMSIX_TOWAR[EN] is cleared.
- The write data size is neither 4 bytes, 32 bytes, nor 64 bytes.
- The write address falls in the MSI-X trap window but is not 64-byte aligned.

Reading from a local physical address that hits in the MSI-X trap window is undefined regardless of the data size.

NOTE

Software is allowed to use cache flush instruction (i.e. DCBF) to optimize the performance of memory write operation to initiate MSI-X operation. Different address offsets are supported to trigger MSI-X operations to allow multiple cores to independently initiate MSI-X operation with different Cache line resident in its private cache.

Undesired behavior could happen if the cache lines containing the MSI-X vector information were made coherent and different cores are accessing the cache line at the same time. Software is responsible to follow the convention of using different 64-byte address offset for triggering MSI-X operations for different physical core or threads.

Hardware keeps track of the MSI-X message request and will send a DW memory transaction with the message address and message data from the corresponding MSI-X vector if all the following conditions are met:

- MSI-X function masking is not set.
- Per-vector masking is not set.
- There is no other outstanding MSI-X transactions being sent.
- The current MSI-X request is the winning request among all the pending MSI-X requests to be sent subject to a Round-Robin priority selection as described in Section 20.4.3.4, "MSI-X Transaction Generation Prioritization.

If any of the above condition is not true and the current MSI-X transaction can not be sent immediately, hardware asserts Pending register bit in the PBA structure. If a MSI-X transaction is successfully sent, hardware clears its Pending register bit in the PBA structure. The Pending register bit status is available to local software via the PEX_AOR and PEX_LDR registers for the corresponding MSI-X vector of each function.

In the case when hardware is able to send a MSI-X transaction, the MSI-X transaction will be sent as a DWORD memory write transaction using the contents of the Message Data field entry for data, the contents of the Message Upper Address field for the upper 32 bits of address, and the contents of the Message Address field entry for the lower 32 bits of address for the corresponding MSI-X vector entry.

When software unmasks a vector whose associated Pending bit is set, hardware schedules sending the associated MSI-X message and clear the Pending bit as soon as the message has been sent. When software clears the MSI-X Function Mask bit, it could results in many messages to be sent and hardware scans through the PBA structure and send each MSI-X message as soon as possible per the prioritization rule described Section 20.4.3.4, "MSI-X Transaction Generation Prioritization.

NOTE

Host software can use the Masking bit in the MSI-X vector table to handle multiple MSI-X interrupt requests. A given vector's interrupt service routine on host can set the vector's Mask bit before it services any associated interrupting events and clears the Mask bit after it has serviced all the events it knows about. Any occurrence of a new event while the Mask bit is set results in the Pending bit being set.

NOTE

If a function sends messages with the same vector multiple times before being acknowledged by software, only one message is guaranteed to be serviced. Software is required to properly acknowledge the MSI-X message interrupts, typically by device driver handshake. Therefore, once a message is sent with Vector A, it can not send Vector A again until it is explicitly enabled to do so by its device driver.

Host software is allowed to mask one or more vectors indefinitely and service the associated interrupt events by polling their Pending bits in PBA structure. The Masking register bit in the Vector Control part of the MSI-X vector table is also made available to the local Software of a function.

20.4.3.4 MSI-X Transaction Generation Prioritization

When there is any Pending MSI-X transactions in PBA, hardware schedules MSI-X transaction with Round-Robin priority selection as described below:

- Hardware maintains one MSI-X vector index register in PEX controller.
- Hardware scans through from the index everytime hardware is scheduling for new MSI-X transaction.
- When a MSI-X vector is selected, the vector number incremented by 1 is recorded by hardware as the index. 0 will be recorded if the selected vector number is the maximum vector number supported by the PEX controller.
- Hardware carries out the handshake with the PCI Express transaction layer to generate the MSI-X transaction for the corresponding MSI-X vector.
- The MSI-X vector index is initialized as 0 at reset.

20.4.4 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 20-179. 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

Table 20-179. Initial credit advertisement

Credit Type	Initial Credit Advertisement
CPLH (Memory Read Completion, IO R/W Completion, Cfg R/W Completion)	Infinite
CPLD (Memory Read Completion, IO Read Completion, Cfg Read Completion)	Infinite

20.4.5 Power Management

All device power states are supported with the exception of D3cold. L2 is supported since main power will remain on during L2/L3 Ready -> L2 transition. 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 20-180. Power Management State Supported

Component D-State	Permissible Interconnect State	Action
D0	L0, L0s	In full operation.
D1	L0, L0s, L1	All outbound traffics are stalled. All inbound traffic is thrown away. The only exceptions are PME messages and configuration transactions. If the device is in RC mode, it is permissible to send a PM_Turn_Off message through the PEX Power Management Command register.
D2	L0, L0s, L1	All outbound traffics are stalled. All inbound traffic is thrown away. The only exceptions are PME messages and configuration transactions. If the device is in RC mode, it is permissible to send a PM_Turn_Off message through the PEX Power Management Command register.
D3hot	L0, L0s, L1, L2/L3 Ready	All outbound traffics are stalled. All inbound traffic is thrown away. The only exceptions are PME messages and configuration transactions. If the device is in RC mode, it is permissible to send a PM_Turn_Off message through the PEX Power Management Command register. Note that if a transition of D3hot->D0 occurs, a reset is performed to the controller's configuration space. In addition, link training restarts.
D3cold	L3	Completely off.

20.4.5.1 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 $\overline{\text{WAKE}}$ 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 $\overline{\text{WAKE}}$ signal, as shown in Figure 20-193.

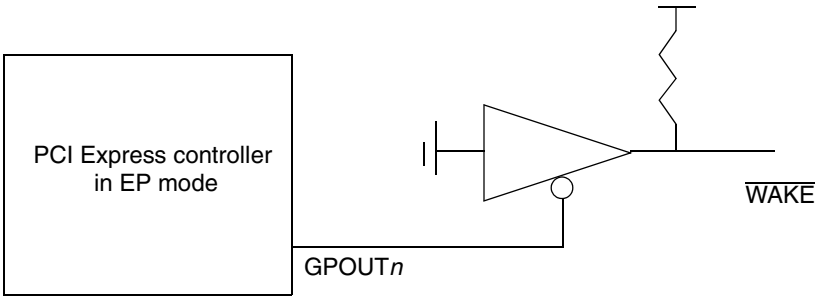


Figure 20-193. $\overline{\text{WAKE}}$ Generation Example

In RC mode, the $\overline{\text{WAKE}}$ signal from the EP device can be connected to one of the external interrupt inputs to service the $\overline{\text{WAKE}}$ request.

20.4.6 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.

20.4.7 Alternative Routing ID (ARI, SR-IOV only)

ARI is not applicable to Root Complex; all other SR-IOV Capable Devices (Devices that include at least one PF) shall implement the ARI Capability in each Function. PF Routing IDs are determined based on the Next Function Number in the Alternative Routing ID Interpretation Extended Capability (ARI) register. VF Routing IDs are determined using the First VF Offset and the VF Stride configuration settings.

20.4.8 Functional Level Reset (FLR, SR-IOV only)

An FLR reset can be issued for a PF or VF by software by setting the IFLR bit in PCI Express device control register associated with the PF/VF. The controller will finish processing all outstanding transactions associated with the PF/VF and return to an idle state. Any new transactions targeted toward the PF or VF while the IFLR bit is set will be dropped (posted transactions) or errored out (outbound non-posted transactions) or returned UR (inbound non-posted transactions) until the IFLR bit is cleared by hardware. Traffic that are toward PF or VF without the IFLR bit set continue as before.

20.4.9 Link Down

Typically, a link down condition occurs after a hot reset event; however, it is possible for the link to go down unexpectedly without a hot reset event. When this occurs, a link down condition is detected (PEX_PME_MSG_DR[LDD]=1). Link down is treated similarly to a hot reset condition.

Subsequently, while the link is down, all new posted outbound transactions are discarded. All new non-posted ATMU transactions are errored out. Non-posted configuration transactions issued using PEX_CONFIG_ADDR/PEX_CONFIG_DATA toward the link returns 0xFFFF_FFFF (all 1s). As soon as the link is up again, the sending of transaction resumes.

A link down condition causes the controller to reset all non-sticky bits in its PCI Express configuration registers as if it had been hot reset.

20.4.10 LIODN Permission Table

The LIODN permission table assigns all inbound transactions toward memory with a 12-bit LIODN field that is used for the purpose of memory's access permission. At initialization, software is responsible for programming the LIODN permission table that maps RID (Routing ID of 16-bit if ARI is enabled or Transaction ID if ARI is not enabled) within a transaction to 10 bit LIODN offset. This 10-bit field is then added to PEX_LBR[LB] to form a 12-bit LIODN field that will identify the transaction as it is forwarded to memory.

Each entry in the LIODN permission table is 64-bit and has the following meaning:

Table 20-181. LIODN permission table entry definition

Bit	Meaning
0:15	RID. Routing ID of a transaction. Bit 0-7 identifies the bus # of a transaction.
16:31	Mask. Identify the bit position of RID to examine. 1 means that the bit should be ignored (ie always result in a match) and 0 means it should be compared.
32:33	Reserved
34:43	LIODN offset. This field identifies the 10-bit LIODN offset that is assigned to a transaction that matches the RID specified in bit 0:15.
61-62	Reserved
63	Enable. This field identifies the matching operation that will be done for the transaction that contains the RID specified in bit 0:15. 0 - disable. This entry is disable. 1 - match on RID & Mask.

The physical size of the LIODN permission table is 64-entry deep. To access the LIODN table, software has to use the PEX_AOR/PEX_UDR/PEX_LDR registers in an indirect manner to program the table's entries to appropriate value. It is assumed that software is responsible for knowing the RIDs within a system and which permission (ie LIODN offset) is used for a particular RID. At startup, the default mode value of entry 0 is enable with a mask of ffff and the LIODN offset for entry 0 is 0.

The following steps are used to determine an LIODN lookup table match:

- If RID matching operation is enabled, an inbound transaction RID is AND'ed with each entry's inverse MASK field.
- Each entry's RID field is AND'ed with the inverse MASK field.
- The results of items #1 and #2 are AND'ed with the entry's EN field and compared to determine if a match has been detected.

- A winning match will be determined by a valid match for the lowest entry.
- If Inbound Memory transaction hits Inbound MSI ATMU window, the resulting LIODN field is calculated from the following
  - {4'b0000, PEXMSI_LBR[LBASE]} +
  - {6'b000000, winning entry's LOFFSET field}
- Otherwise,The resulting LIODN field is calculated from the following
  - {4'b0000, PEX_LBR[LBASE]} +
  - {6'b000000, winning entry's LOFFSET field}

An inbound memory request that does not match any entry of the LIODN lookup table will be considered a logical inbound LIODN error and logged in PEX_ERR_DR[LDDE] register. Inbound Non-Posted memory transaction will return UR and Inbound Posted memory transaction will be dropped.

At startup, the default values of the LIODN permission table look like this:

Table 20-182. LIODN permission table default value

Entry#	Entry value	Meaning
0	0x0000_ffff_0000_0001	All memory transactions are accepted and LIODN offset of 0 is used if the address of the transaction is hitting into any PF or VF space.
1:63	0x0000_0000_0000_0000	Entry is disable.

Here is another example of how the LIODN permission table is programmed.

Table 20-183. LIODN permission table example

Entry#	Entry value	Meaning
0	0x1234_0000_0010_1001	All memory transactions with RID of 0x1234 are allowed to map to PF 0 space only. If the address of this transaction is hitting into PF 0's PF space, then LIODN offset of 1 is used for the transaction. Otherwise, for posted transaction, it will be dropped. For non-posted transaction, a UR is returned.
1	0x5678_0000_0029_8009	All memory transactions with RID of 0x5678 are allowed to map to PF 0, or PF 3, or PF 4 space only. If the address of this transaction is hitting into PF 0's VF 1 space, or PF 3's VF 1 space, or PF 4's VF 1 space, then LIODN offset of 2 is used for the transaction. Otherwise, for posted transaction, it will be dropped. For non-posted transaction, a UR is returned.
2	0x789A_00ff_0010_0019	All memory transactions with RID's bus # of 0x78 are allowed to map to PF 0 space only. If the address of this transaction is hitting into PF 0's VF 3 space, then LIODN offset of 3 is used for the transaction. Otherwise, for posted transaction, it will be dropped. For non-posted transaction, a UR is returned.
3	0xXXXX_ffff_XXXX_xxx1	All memory transactions are accepted and LIODN offset of 0 is used if the address of the transaction is hitting into any PF or VF space. Otherwise, for posted transaction, it will be dropped. For non-posted transaction, a UR is returned.
4:63	0x0000_0000_0000_0000	Entry is disable.

20.5 Initialization/Application Information

20.5.1 Initial start-up programming guide

Depending on whether the device in in EP or RC mode, Table 20-184 shows the minimum registers that must be initialized after link training finishes successfully in order for proper system operation. Software is responsible for initializing the steps.

Table 20-184. Recommended Initialization Steps

Non-SR-IOV Device		SR-IOV Device	
RC	**EP**	**RC**	**EP**
1. Set PCI Express Device Control register's MAX_PAYLOAD_SIZE and MAX_READ_SIZE to appropriate values 2. Program ATMU registers. See Table 20-25 for more inforlmation 3. Set PCI Express Command register's Bus Master and Memory Space bits 4. Begin operation	1. Set PCI Express Device Control register's MAX_PAYLOAD_SIZE and MAX_READ_SIZE to appropriate values 2. Program ATMU registers. See Table 20-25 for more inforlmation 3. Set PEX_CONFIG's CFG_READY bit. 4. Set PCI Express Command register's Bus Master and Memory Space bits 5. Begin operation	1. Set PCI Express Device Control register's MAX_PAYLOAD_SIZE and MAX_READ_SIZE to appropriate values 2. Program ATMU registers. See Table 20-25 for more inforlmation 3. Set PCI Express Command register's Bus Master and Memory Space bits 4. Begin operation	1. Set PCI Express Device Control register's MAX_PAYLOAD_SIZE and MAX_READ_SIZE to appropriate values 2. Program ATMU registers. See Table 20-25 for more inforlmation 3. Set PEX_CONFIG's CFG_READY bit. 4. Set PCI Express Command register's Bus Master and Memory Space bits 5. Begin operation

20.5.2 Boot Mode and Inbound Configuration Transactions

In normal boot mode, the core is allowed to boot and configure the device. During this time, the PCI Express interface retries all inbound PCI Express configuration transactions. When the core has configured the device to a state where it can accept inbound PCI Express configuration transactions, the boot code should set the CFG_READY bit in the PEX_CONFIG register using a read-modify-write operation after which inbound PCI Express configuration transactions are accepted. Refer to Section 20.3.2.5, "PCI Express Configuration Register (PEX_CONFIG)," for more information about the CFG_READY bit.

In boot hold-off mode, the core is prevented from fetching its first instruction by withholding its internal bus grant. During this time, the PCI Express interface accepts all inbound PCI Express configuration transactions which allows an external host/RC to configure the device. When the external host/RC has configured the device to a state where it can allow the core to fetch code from the boot vector, it reconfigures the PCI Express controller to not be in hold-off mode in order for the core to start fetching its first instruction.

20.5.3 Expansion ROM

PCI Express Expansion ROM is supported with shared inbound expansion ROM ATMU window registers (PEXEPROMITAR and PEXEPROMIWAR) across physical functions.

During pre-boot initialization, the PBL can configure the inbound expansion ROM ATMU window registers in the PCI Express controller. Accesses from a remote host to the expansion ROM address range are translated to internal platform accesses to non-volatile memories attached to the local physical space for boot operations.

20.5.4 SR-IOV RID examples

Based on the number of VFs and PFs, different RIDs should be used to ensure approriate accesses to the configuration space of each PF/VF. Please note for RID that has bus number not on the device's captured bus number, it can only be accessed using configuration Type 1 cycle. Please note that SR-IOV Control[ARICH] must always be enabled by RC and RC must also set Device Control 2 [ARI Forwarding Enable] to one.

Table 20-185. SR-IOV configuration

SR-IOV Field	PF0	PF1
NumVF	64	64
First VF Offset	4	4
VF Stride	4	4

Table 20-186. SR-IOV RID examples

Function number	Description	Note
0	VF1, 2	Use configuration type 0 access
1	PF1	Use configuration type 0 access
2	Function not present	
3	Function not present	
4	VF 0,1	Use configuration type 0 access
5	VF 1, 1	Use configuration type 0 access
6	VF not present	
7	VF not present	
8	VF 0, 2	Use configuration type 0 access
9	VF 1, 2	Use configuration type 0 access
10	VF not present	
11	VF not present	
...	...	...
256	VF 0, 64	Use configuration type 1 access
257	VF 1, 64	Use configuration type 1 access

20.5.5 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 0x890.

Chapter 21
Serial RapidIO Interface

21.1 SRIO Overview

The serial RapidIO interface provides a RapidIO port to communicate with other RapidIO devices. This figure shows the RapidIO port unit.

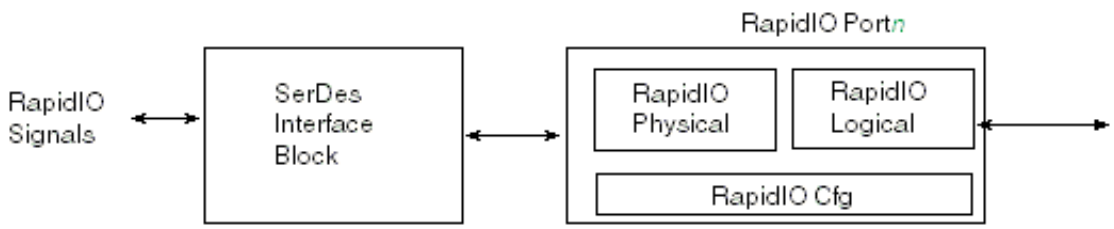


Figure 21-1. RapidIO Endpoint

21.2 SRIO Features Summary

The RapidIO port supports the following features of the RapidIO Interconnect Specification, Revision 2.1:

- Small or large size transport information field
- 34-bit addressing
- Up to 256-byte data payload
- Up to eight outstanding unacknowledged RapidIO transactions
- Hardware recovery only
- All transaction flows and all priorities
- Critical request flow (CRF) as described in the *RapidIO Interconnect Specification, Revision 1.3* Part 6: 1x/rx LP-Serial Physical Layer Specification.
- Register and register bit extensions as described in the *RapidIO Interconnect Specification, Revision 1.3*, *Part VIII: Error Management Extensions Specification*.
- Hot swap
- ATOMIC set/clr/inc/dec for read-modify-write operations

- IO_READ_HOME and FLUSH w/data for accessing cache-coherent data from a remote memory system
- Data streaming (type 9) and traffic management (type 9 extended) packet types as defined in the RapidIO Interconnect Specification, Revision 2.0.1, Part 10: Data Streaming Logical Specification
- Register and register bit extensions as described in the RapidIO Interconnect Specification, Revision 2.0.1, Part 10: Data Streaming Logical Specification
- Only supports receiver-controlled flow control
- Inbound transactions to the configuration registers are limited to 32-bit accesses only.
- Outbound maintenance transactions can be any valid size.

RapidIO endpoint supports the following user-defined features:

- Nine outbound ATMU windows with each window having up to 32 subwindows except the default window
- Five inbound ATMU windows
- Logical outbound packet time-to-live counter to prevent local processor from hanging when the RIO interface fails
- Accept-all mode of operation for failover support
- RapidIO random bit error injection
- Performance monitor interface

RapidIO endpoint does not support or has limited support of the following features of the *RapidIO Interconnect Specification, Revision 2.1*:

- No support for 50- and 66-bit addressing
- No support for software assisted error recovery
- No support for ATOMIC test-and-swap transaction
- No support for coherent (CC-NUMA) transactions with the exception of IO_READ_HOME and FLUSH w/data transactions
- No support for transmitter-controlled flow control
- No decrementing of a maintenance packet hop count (pass-through support does not imply switch functionality)
- No support for multicast event control symbols

RapidIO endpoint supports the following features of RapidIO LP-Serial:

- 1x, 2x, and 4x LP-Serial link interfaces
- Transmission rates of 2.5, 3.125, and 5 Gbaud (data rates of 2.0, 2.5, and 4.0 Gbps) per lane
- Auto detection of 1x, 2x, and 4x mode operation during port initialization
- Error detection for packets and control symbols
- Support for link initialization, synchronization, error recovery, and time-out

RapidIO endpoint does not support the following features of RapidIO LP-Serial:

- RapidIO endpoint cannot be configured as four 1x or two 2x ports

21.3 SRIO RapidIO Port Modes of Operation

The RapidIO port's primary operating modes include the following:

- 1x, 2x, or 4x LP-Serial link interfaces
- Transmission rates of 2.5, 3.125, or 5 Gbaud (data rates of 2.0, 2.5, or 4.0 Gbps) per lane
- Small or large size transport information field
- Accept-all mode of operation-all packets are accepted regardless of the target ID

21.4 LP-Serial Signal Descriptions

The high-speed interface signals used for the serial RapidIO interface are shared with other I/O ports and must be configured at power-on reset for use with the serial RapidIO controller

The following sections describe the serial RapidIO signal functionality. Refer to the *RapidIO Interconnect Specification,* Revision 2.0.1, Part 6: LP-Serial Physical Layer Specification, Chapter 8, Common Electrical Specifications, for electrical characteristic details.

21.4.1 Serial Rapid I/O Interface Overview

The serial Rapid I/O (SRIO) interface is compatible with the *RapidIO Interconnect Specification,* Revision 2.0.1, Part 6: LP-Serial Physical Layer Specification.

21.4.2 Serial Rapid I/O Interface Detailed Signal Descriptions

21.4.2.1 SD*n*_TX[*n*]/SD*n*_TX[*n*]_B-Outputs

These are the serial data outputs. They are differential pairs, one for each lane in 4x mode of operation. In the case of 2x mode of operation on the serial RapidIO interface, only the first and second lane may be used. In the case of 1x mode of operation on the serial RapidIO interface, only the first and third lanes may be used.

See *Part VI: Physical Layer 1x/4x LP-Serial Specification*, *RapidIO Interconnect Specification, Revision 2.1,* for complete details.

These outputs are asynchronous, as described in Part VI of the *RapidIO Interconnect Specification, Revision 2.1*. This implementation supports data rates of 2.5, 3.125, and 5 Gbaud.

21.4.2.2 SD*n*_RX[*n*]/SD*n*_RX[*n*]_B-Inputs

These are the serial data input pads. They are differential pairs, one for each lane in 4x mode of operation. In the case of 2x mode of operation on the serial RapidIO interface, only the first and second lane may be used. In the case of 1x mode of operation on the serial RapidIO interface, only the first lane is used.

See *Part VI: Physical Layer 1x/4x LP-Serial Specification*, *RapidIO Interconnect Specification, Revision 1.2.* for complete details.

These inputs are asynchronous, as described in Part VI of the *RapidIO Interconnect Specification, Revision 1.2*. This implementation supports data rates of 2.5, 3.125, and 5 Gbaud.

21.5 SRIO Memory Map/Register Definition

This table is the memory map of the RapidIO configuration registers.

SRIO memory map

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
C_0000	Device identity capability register (SRIO_DIDCAR)	32	R	0830_0002h	21.5.1/1171
C_0004	Device information capability register (SRIO_DICAR)	32	R	0000_0000h	21.5.2/1172
C_0008	Assembly identity capability register (SRIO_AIDCAR)	32	R/W	0000_0000h	21.5.3/1172
C_000C	Assembly information capability register (SRIO_AICAR)	32	R/W	0000_0100h	21.5.4/1173

Table continues on the next page...

SRIO memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
C_0010	Processing element features capability register (SRIO_PEFCAR)	32	R	See section	21.5.5/1173
C_0018	Source operations capability register (SRIO_SOCAR)	32	R	060C_FCF4h	21.5.6/1176
C_001C	Destination operations capability register (SRIO_DOCAR)	32	R	000C_FCF4h	21.5.7/1179
C_003C	Data streaming information capability register (SRIO_DSICAR)	32	R	0000_0018h	21.5.8/1182
C_0048	Data streaming logic layer command and status register (SRIO_DSLLCSR)	32	R	8000_0040h	21.5.9/1182
C_004C	Processing element logical layer control command and status register (SRIO_PELLCCSR)	32	R	0000_0001h	21.5.10/ 1183
C_005C	Local configuration space base address 1 command and status register (SRIO_LCSBA1CSR)	32	R/W	0000_0000h	21.5.11/ 1184
C_0060	Base device ID command and status register (SRIO_BDIDCSR)	32	R/W	See section	21.5.12/ 1185
C_0068	Host base device ID lock command and status register (SRIO_HBDIDLCSR)	32	R/W	0000_FFFFh	21.5.13/ 1186
C_006C	Component tag command and status register (SRIO_CTCSR)	32	R/W	0000_0000h	21.5.14/ 1186
C_0100	Port maintenance block header 0 (SRIO_PMBH0)	32	R	0600_0001h	21.5.15/ 1187
C_0120	Port link time-out control command and status register (SRIO_PLTOCCSR)	32	R/W	FFFF_FF00h	21.5.16/ 1187
C_0124	Port response time-out control command and status register (SRIO_PRTOCCSR)	32	R/W	FFFF_FF00h	21.5.17/ 1188
C_013C	Port General control command and status register (SRIO_GCCSR)	32	R/W	See section	21.5.18/ 1189
C_0140	Port 1 Link maintenance request command and status register (SRIO_P1LMREQCSR)	32	R/W	0000_0000h	21.5.19/ 1190
C_0144	Port 1 Link maintenance response command and status register (SRIO_P1LMRESPCSR)	32	R	0000_0000h	21.5.20/ 1191
C_0148	Port 1 Local ackID status command and status register (SRIO_P1LASCSR)	32	R/W	0000_0000h	21.5.21/ 1192
C_0158	Port 1 Error and status command and status register (SRIO_P1ESCSR)	32	R/W	0000_0001h	21.5.22/ 1193
C_015C	Port 1 Control command and status register (SRIO_P1CCSR)	32	R/W	D0C2_0001h	21.5.23/ 1195
C_0160	Port 2 Link maintenance request command and status register (SRIO_P2LMREQCSR)	32	R/W	0000_0000h	21.5.24/ 1197
C_0164	Port 2 Link maintenance response command and status register (SRIO_P2LMRESPCSR)	32	R	0000_0000h	21.5.25/ 1198
C_0168	Port 2 Local ackID status command and status register (SRIO_P2LASCSR)	32	R/W	0000_0000h	21.5.26/ 1199
C_0178	Port 2 Error and status command and status register (SRIO_P2ESCSR)	32	R/W	0000_0001h	21.5.27/ 1200

Table continues on the next page...

SRIO memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
C_017C	Port 2 Control command and status register (SRIO_P2CCSR)	32	R/W	D0C2_0001h	21.5.28/ 1202
C_0600	Error reporting block header (SRIO_ERBH)	32	R	0000_0007h	21.5.29/ 1204
C_0608	Logical/Transport layer error detect command and status register (SRIO_LTLEDCSR)	32	R/W	0000_0000h	21.5.30/ 1205
C_060C	Logical/Transport layer error enable command and status register (SRIO_LTLEECSR)	32	R/W	0000_0000h	21.5.31/ 1208
C_0614	Logical/Transport layer address capture command and status register (SRIO_LTLACCSR)	32	R/W	0000_0000h	21.5.32/ 1209
C_0618	Logical/Transport layer device ID capture command and status register (SRIO_LTLDIDCCSR)	32	R/W	0000_0000h	21.5.33/ 1210
C_061C	Logical/Transport layer control capture command and status register (SRIO_LTLCCCSR)	32	R/W	0000_0000h	21.5.34/ 1211
C_0640	Port 1 Error detect command and status register (SRIO_P1EDCSR)	32	R/W	0000_0000h	21.5.35/ 1212
C_0644	Port 1 Error rate enable command and status register (SRIO_P1ERECSR)	32	R/W	0000_0000h	21.5.36/ 1214
C_0648	Port 1 Error capture attributes command and status register (SRIO_P1ECACSR)	32	R/W	0000_0000h	21.5.37/ 1216
C_064C	Port 1 Packet/control symbol error capture command and status register 0 (SRIO_P1PCSECCSR0)	32	R/W	0000_0000h	21.5.38/ 1218
C_0650	Port 1 Packet error capture command and status register 1 (SRIO_P1PECCSR1)	32	R/W	0000_0000h	21.5.39/ 1218
C_0654	Port 1 Packet error capture command and status register 2 (SRIO_P1PECCSR2)	32	R/W	0000_0000h	21.5.40/ 1219
C_0658	Port 1 Packet error capture command and status register 3 (SRIO_P1PECCSR3)	32	R/W	0000_0000h	21.5.41/ 1219
C_0668	Port 1 Error rate command and status register (SRIO_P1ERCSR)	32	R/W	8000_0000h	21.5.42/ 1220
C_066C	Port 1 Error rate threshold command and status register (SRIO_P1ERTCSR)	32	R/W	FFFF_0000h	21.5.43/ 1221
C_0680	Port 2 Error detect command and status register (SRIO_P2EDCSR)	32	R/W	0000_0000h	21.5.44/ 1222
C_0684	Port 2 Error rate enable command and status register (SRIO_P2ERECSR)	32	R/W	0000_0000h	21.5.45/ 1224
C_0688	Port 2 Error capture attributes command and status register (SRIO_P2ECACSR)	32	R/W	0000_0000h	21.5.46/ 1226
C_068C	Port 2 Packet/control symbol error capture command and status register 0 (SRIO_P2PCSECCSR0)	32	R/W	0000_0000h	21.5.47/ 1228
C_0690	Port 2 Packet error capture command and status register 1 (SRIO_P2PECCSR1)	32	R/W	0000_0000h	21.5.48/ 1228
C_0694	Port 2 Packet error capture command and status register 2 (SRIO_P2PECCSR2)	32	R/W	0000_0000h	21.5.49/ 1229

Table continues on the next page...

SRIO memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
C_0698	Port 2 Packet error capture command and status register 3 (SRIO_P2PECCSR3)	32	R/W	0000_0000h	21.5.50/ 1229
C_06A8	Port 2 Error rate command and status register (SRIO_P2ERCSR)	32	R/W	8000_0000h	21.5.51/ 1230
C_06AC	Port 2 Error rate threshold command and status register (SRIO_P2ERTCSR)	32	R/W	FFFF_0000h	21.5.52/ 1231
D_0004	Logical layer configuration register (SRIO_LLCR)	32	R/W	0000_0000h	21.5.53/ 1232
D_0010	Error / port-write interrupt status register (SRIO_EPWISR)	32	R	0000_0000h	21.5.54/ 1233
D_0020	Logical retry error threshold configuration register (SRIO_LRETCR)	32	R/W	0000_00FFh	21.5.55/ 1234
D_0080	Physical retry error threshold configuration register (SRIO_PRETCR)	32	R/W	0000_00FFh	21.5.56/ 1235
D_0100	Port 1 Alternate device ID command and status register (SRIO_P1ADIDCSR)	32	R/W	0000_0000h	21.5.57/ 1235
D_0120	Port 1 Accept-all configuration register (SRIO_P1AACR)	32	R/W	8000_0001h	21.5.58/ 1236
D_0124	Port 1 Logical Outbound Packet time-to-live configuration register (SRIO_P1LOPTTLCR)	32	R/W	0000_0000h	21.5.59/ 1237
D_0130	Port 1 Implementation error command and status register (SRIO_P1IECSR)	32	w1c	0000_0000h	21.5.60/ 1238
D_0140	Port 1 Physical configuration register (SRIO_P1PCR)	32	R/W	0000_8010h	21.5.61/ 1239
D_0158	Port 1 Serial link command and status register (SRIO_P1SLCSR)	32	w1c	0000_0000h	21.5.62/ 1240
D_0160	Port 1 Serial link error injection configuration register (SRIO_P1SLEICR)	32	R/W	0000_0000h	21.5.63/ 1241
D_0164	Port 1 Arbitration 0 Tx Configuration Register (SRIO_P1A0TxCR)	32	R/W	0000_03FFh	21.5.64/ 1242
D_0168	Port 1 Arbitration 1 Tx Configuration Register (SRIO_P1A1TxCR)	32	R/W	8000_001Fh	21.5.65/ 1243
D_016C	Port 1 Arbitration 2 Tx Configuration Register (SRIO_P1A2TxCR)	32	R/W	8000_001Fh	21.5.66/ 1244
D_0170	Port 1 Message Request Tx Buffer Allocation Configuration Register 0 (SRIO_P1MReqTxBACR0)	32	R/W	0101_0101h	21.5.67/ 1245
D_0178	Port 1 Message Request Tx Buffer Allocation Configuration Register 2 (SRIO_P1MReqTxBACR2)	32	R/W	0700_0000h	21.5.68/ 1247
D_017C	Port 1 Message Response / Flow Control Tx Buffer Allocation Configuration Register (SRIO_P1MRspFcTxBACR)	32	R/W	0101_0E00h	21.5.69/ 1248
D_0180	Port 2 Alternate device ID command and status register (SRIO_P2ADIDCSR)	32	R/W	0000_0000h	21.5.70/ 1249
D_01A0	Port 2 Accept-all configuration register (SRIO_P2AACR)	32	R/W	8000_0001h	21.5.71/ 1250

Table continues on the next page...

SRIO memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
D_01A4	Port 2 Logical Outbound Packet time-to-live configuration register (SRIO_P2LOPTTLCR)	32	R/W	0000_0000h	21.5.72/ 1251
D_01B0	Port 2 Implementation error command and status register (SRIO_P2IECSR)	32	w1c	0000_0000h	21.5.73/ 1252
D_01C0	Port 2 Physical configuration register (SRIO_P2PCR)	32	R/W	0000_8010h	21.5.74/ 1253
D_01D8	Port 2 Serial link command and status register (SRIO_P2SLCSR)	32	w1c	0000_0000h	21.5.75/ 1254
D_01E0	Port 2 Serial link error injection configuration register (SRIO_P2SLEICR)	32	R/W	0000_0000h	21.5.76/ 1255
D_01E4	Port 2 Arbitration 0 Tx Configuration Register (SRIO_P2A0TxCR)	32	R/W	0000_03FFh	21.5.77/ 1256
D_01E8	Port 2 Arbitration 1 Tx Configuration Register (SRIO_P2A1TxCR)	32	R/W	8000_001Fh	21.5.78/ 1257
D_01EC	Port 2 Arbitration 2 Tx Configuration Register (SRIO_P2A2TxCR)	32	R/W	8000_001Fh	21.5.79/ 1258
D_01F0	Port 2 Message Request Tx Buffer Allocation Configuration Register 0 (SRIO_P2MReqTxBACR0)	32	R/W	0101_0101h	21.5.80/ 1259
D_01F8	Port 2 Message Request Tx Buffer Allocation Configuration Register 2 (SRIO_P2MReqTxBACR2)	32	R/W	0700_0000h	21.5.81/ 1261
D_01FC	Port 2 Message Response / Flow Control Tx Buffer Allocation Configuration Register (SRIO_P2MRspFcTxBACR)	32	R/W	0101_0E00h	21.5.82/ 1262
D_0BF8	IP Block Revision Register 1 (SRIO_IPBRR1)	32	R	01C0_0201h	21.5.83/ 1263
D_0BFC	IP Block Revision Register 2 (SRIO_IPBRR2)	32	R	0000_0000h	21.5.84/ 1264
D_0C00	Port 1 RapidIO outbound window translation address register n (SRIO_P1ROWTAR0)	32	R/W	FF80_0000h	21.5.85/ 1264
D_0C04	Port 1 RapidIO outbound window translation extended address register 0 (SRIO_P1ROWTEAR0)	32	R/W	0000_003Fh	21.5.86/ 1265
D_0C10	Port 1 RapidIO outbound window attributes register 0 (SRIO_P1ROWAR0)	32	R/W	8004_4027h	21.5.87/ 1266
D_0C20	Port 1 RapidIO outbound window translation address register n (SRIO_P1ROWTAR1)	32	R/W	0000_0000h	21.5.88/ 1268
D_0C24	Port 1 RapidIO outbound window translation extended address register n (SRIO_P1ROWTEAR1)	32	R/W	0000_0000h	21.5.89/ 1269
D_0C28	Port 1 RapidIO outbound window base address register n (SRIO_P1ROWBAR1)	32	R/W	0000_0000h	21.5.90/ 1270
D_0C30	Port 1 RapidIO outbound window attributes register n (SRIO_P1ROWAR1)	32	R/W	0004_4027h	21.5.91/ 1271
D_0C34	Port 1 RapidIO outbound window segment 1 register n (SRIO_P1ROWS1R1)	32	R/W	0044_0000h	21.5.92/ 1274
D_0C38	Port 1 RapidIO outbound window segment 2 register n (SRIO_P1ROWS2R1)	32	R/W	0044_0000h	21.5.93/ 1276

Table continues on the next page...

SRIO memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
D_0C3C	Port 1 RapidIO outbound window segment 3 register n (SRIO_P1ROWS3R1)	32	R/W	0044_0000h	21.5.94/ 1278
D_0C40	Port 1 RapidIO outbound window translation address register n (SRIO_P1ROWTAR2)	32	R/W	0000_0000h	21.5.88/ 1268
D_0C44	Port 1 RapidIO outbound window translation extended address register n (SRIO_P1ROWTEAR2)	32	R/W	0000_0000h	21.5.89/ 1269
D_0C48	Port 1 RapidIO outbound window base address register n (SRIO_P1ROWBAR2)	32	R/W	0000_0000h	21.5.90/ 1270
D_0C50	Port 1 RapidIO outbound window attributes register n (SRIO_P1ROWAR2)	32	R/W	0004_4027h	21.5.91/ 1271
D_0C54	Port 1 RapidIO outbound window segment 1 register n (SRIO_P1ROWS1R2)	32	R/W	0044_0000h	21.5.92/ 1274
D_0C58	Port 1 RapidIO outbound window segment 2 register n (SRIO_P1ROWS2R2)	32	R/W	0044_0000h	21.5.93/ 1276
D_0C5C	Port 1 RapidIO outbound window segment 3 register n (SRIO_P1ROWS3R2)	32	R/W	0044_0000h	21.5.94/ 1278
D_0C60	Port 1 RapidIO outbound window translation address register n (SRIO_P1ROWTAR3)	32	R/W	0000_0000h	21.5.88/ 1268
D_0C64	Port 1 RapidIO outbound window translation extended address register n (SRIO_P1ROWTEAR3)	32	R/W	0000_0000h	21.5.89/ 1269
D_0C68	Port 1 RapidIO outbound window base address register n (SRIO_P1ROWBAR3)	32	R/W	0000_0000h	21.5.90/ 1270
D_0C70	Port 1 RapidIO outbound window attributes register n (SRIO_P1ROWAR3)	32	R/W	0004_4027h	21.5.91/ 1271
D_0C74	Port 1 RapidIO outbound window segment 1 register n (SRIO_P1ROWS1R3)	32	R/W	0044_0000h	21.5.92/ 1274
D_0C78	Port 1 RapidIO outbound window segment 2 register n (SRIO_P1ROWS2R3)	32	R/W	0044_0000h	21.5.93/ 1276
D_0C7C	Port 1 RapidIO outbound window segment 3 register n (SRIO_P1ROWS3R3)	32	R/W	0044_0000h	21.5.94/ 1278
D_0C80	Port 1 RapidIO outbound window translation address register n (SRIO_P1ROWTAR4)	32	R/W	0000_0000h	21.5.88/ 1268
D_0C84	Port 1 RapidIO outbound window translation extended address register n (SRIO_P1ROWTEAR4)	32	R/W	0000_0000h	21.5.89/ 1269
D_0C88	Port 1 RapidIO outbound window base address register n (SRIO_P1ROWBAR4)	32	R/W	0000_0000h	21.5.90/ 1270
D_0C90	Port 1 RapidIO outbound window attributes register n (SRIO_P1ROWAR4)	32	R/W	0004_4027h	21.5.91/ 1271
D_0C94	Port 1 RapidIO outbound window segment 1 register n (SRIO_P1ROWS1R4)	32	R/W	0044_0000h	21.5.92/ 1274
D_0C98	Port 1 RapidIO outbound window segment 2 register n (SRIO_P1ROWS2R4)	32	R/W	0044_0000h	21.5.93/ 1276
D_0C9C	Port 1 RapidIO outbound window segment 3 register n (SRIO_P1ROWS3R4)	32	R/W	0044_0000h	21.5.94/ 1278

Table continues on the next page...

SRIO memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
D_0CA0	Port 1 RapidIO outbound window translation address register n (SRIO_P1ROWTAR5)	32	R/W	0000_0000h	21.5.88/ 1268
D_0CA4	Port 1 RapidIO outbound window translation extended address register n (SRIO_P1ROWTEAR5)	32	R/W	0000_0000h	21.5.89/ 1269
D_0CA8	Port 1 RapidIO outbound window base address register n (SRIO_P1ROWBAR5)	32	R/W	0000_0000h	21.5.90/ 1270
D_0CB0	Port 1 RapidIO outbound window attributes register n (SRIO_P1ROWAR5)	32	R/W	0004_4027h	21.5.91/ 1271
D_0CB4	Port 1 RapidIO outbound window segment 1 register n (SRIO_P1ROWS1R5)	32	R/W	0044_0000h	21.5.92/ 1274
D_0CB8	Port 1 RapidIO outbound window segment 2 register n (SRIO_P1ROWS2R5)	32	R/W	0044_0000h	21.5.93/ 1276
D_0CBC	Port 1 RapidIO outbound window segment 3 register n (SRIO_P1ROWS3R5)	32	R/W	0044_0000h	21.5.94/ 1278
D_0CC0	Port 1 RapidIO outbound window translation address register n (SRIO_P1ROWTAR6)	32	R/W	0000_0000h	21.5.88/ 1268
D_0CC4	Port 1 RapidIO outbound window translation extended address register n (SRIO_P1ROWTEAR6)	32	R/W	0000_0000h	21.5.89/ 1269
D_0CC8	Port 1 RapidIO outbound window base address register n (SRIO_P1ROWBAR6)	32	R/W	0000_0000h	21.5.90/ 1270
D_0CD0	Port 1 RapidIO outbound window attributes register n (SRIO_P1ROWAR6)	32	R/W	0004_4027h	21.5.91/ 1271
D_0CD4	Port 1 RapidIO outbound window segment 1 register n (SRIO_P1ROWS1R6)	32	R/W	0044_0000h	21.5.92/ 1274
D_0CD8	Port 1 RapidIO outbound window segment 2 register n (SRIO_P1ROWS2R6)	32	R/W	0044_0000h	21.5.93/ 1276
D_0CDC	Port 1 RapidIO outbound window segment 3 register n (SRIO_P1ROWS3R6)	32	R/W	0044_0000h	21.5.94/ 1278
D_0CE0	Port 1 RapidIO outbound window translation address register n (SRIO_P1ROWTAR7)	32	R/W	0000_0000h	21.5.88/ 1268
D_0CE4	Port 1 RapidIO outbound window translation extended address register n (SRIO_P1ROWTEAR7)	32	R/W	0000_0000h	21.5.89/ 1269
D_0CE8	Port 1 RapidIO outbound window base address register n (SRIO_P1ROWBAR7)	32	R/W	0000_0000h	21.5.90/ 1270
D_0CF0	Port 1 RapidIO outbound window attributes register n (SRIO_P1ROWAR7)	32	R/W	0004_4027h	21.5.91/ 1271
D_0CF4	Port 1 RapidIO outbound window segment 1 register n (SRIO_P1ROWS1R7)	32	R/W	0044_0000h	21.5.92/ 1274
D_0CF8	Port 1 RapidIO outbound window segment 2 register n (SRIO_P1ROWS2R7)	32	R/W	0044_0000h	21.5.93/ 1276
D_0CFC	Port 1 RapidIO outbound window segment 3 register n (SRIO_P1ROWS3R7)	32	R/W	0044_0000h	21.5.94/ 1278
D_0D00	Port 1 RapidIO outbound window translation address register n (SRIO_P1ROWTAR8)	32	R/W	0000_0000h	21.5.88/ 1268

Table continues on the next page...

SRIO memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
D_0D04	Port 1 RapidIO outbound window translation extended address register n (SRIO_P1ROWTEAR8)	32	R/W	0000_0000h	21.5.89/ 1269
D_0D08	Port 1 RapidIO outbound window base address register n (SRIO_P1ROWBAR8)	32	R/W	0000_0000h	21.5.90/ 1270
D_0D10	Port 1 RapidIO outbound window attributes register n (SRIO_P1ROWAR8)	32	R/W	0004_4027h	21.5.91/ 1271
D_0D14	Port 1 RapidIO outbound window segment 1 register n (SRIO_P1ROWS1R8)	32	R/W	0044_0000h	21.5.92/ 1274
D_0D18	Port 1 RapidIO outbound window segment 2 register n (SRIO_P1ROWS2R8)	32	R/W	0044_0000h	21.5.93/ 1276
D_0D1C	Port 1 RapidIO outbound window segment 3 register n (SRIO_P1ROWS3R8)	32	R/W	0044_0000h	21.5.94/ 1278
D_0D60	Port 1 RapidIO Inbound window translation address register n (SRIO_P1RIWTAR4)	32	R/W	0000_0000h	21.5.95/ 1280
D_0D68	Port 1 RapidIO Inbound window base address register n (SRIO_P1RIWBAR4)	32	R/W	0000_0000h	21.5.96/ 1281
D_0D70	Port 1 RapidIO inbound window attributes register n (SRIO_P1RIWAR4)	32	R/W	0004_4021h	21.5.97/ 1282
D_0D80	Port 1 RapidIO Inbound window translation address register n (SRIO_P1RIWTAR3)	32	R/W	0000_0000h	21.5.95/ 1280
D_0D88	Port 1 RapidIO Inbound window base address register n (SRIO_P1RIWBAR3)	32	R/W	0000_0000h	21.5.96/ 1281
D_0D90	Port 1 RapidIO inbound window attributes register n (SRIO_P1RIWAR3)	32	R/W	0004_4021h	21.5.97/ 1282
D_0DA0	Port 1 RapidIO Inbound window translation address register n (SRIO_P1RIWTAR2)	32	R/W	0000_0000h	21.5.95/ 1280
D_0DA8	Port 1 RapidIO Inbound window base address register n (SRIO_P1RIWBAR2)	32	R/W	0000_0000h	21.5.96/ 1281
D_0DB0	Port 1 RapidIO inbound window attributes register n (SRIO_P1RIWAR2)	32	R/W	0004_4021h	21.5.97/ 1282
D_0DC0	Port 1 RapidIO Inbound window translation address register n (SRIO_P1RIWTAR1)	32	R/W	0000_0000h	21.5.95/ 1280
D_0DC8	Port 1 RapidIO Inbound window base address register n (SRIO_P1RIWBAR1)	32	R/W	0000_0000h	21.5.96/ 1281
D_0DD0	Port 1 RapidIO inbound window attributes register n (SRIO_P1RIWAR1)	32	R/W	0004_4021h	21.5.97/ 1282
D_0DE0	Port 1 RapidIO inbound window translation address register 0 (SRIO_P1RIWTAR0)	32	R/W	0000_0000h	21.5.98/ 1284
D_0DF0	Port 1 RapidIO inbound window attributes register 0 (SRIO_P1RIWAR0)	32	R/W	8004_4021h	21.5.99/ 1285
D_0E00	Port 2 RapidIO outbound window translation address register n (SRIO_P2ROWTAR0)	32	R/W	FF80_0000h	21.5.100/ 1287
D_0E04	Port 2 RapidIO outbound window translation extended address register 0 (SRIO_P2ROWTEAR0)	32	R/W	0000_003Fh	21.5.101/ 1288

Table continues on the next page...

SRIO memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
D_0E10	Port 2 RapidIO outbound window attributes register 0 (SRIO_P2ROWAR0)	32	R/W	8004_4027h	21.5.102/ 1288
D_0E20	Port 2 RapidIO outbound window translation address register n (SRIO_P2ROWTAR1)	32	R/W	0000_0000h	21.5.103/ 1291
D_0E24	Port 2 RapidIO outbound window translation extended address register n (SRIO_P2ROWTEAR1)	32	R/W	0000_0000h	21.5.104/ 1292
D_0E28	Port 2 RapidIO outbound window base address register n (SRIO_P2ROWBAR1)	32	R/W	0000_0000h	21.5.105/ 1292
D_0E30	Port 2 RapidIO outbound window attributes register n (SRIO_P2ROWAR1)	32	R/W	0004_4027h	21.5.106/ 1293
D_0E34	Port 2 RapidIO outbound window segment 1 register n (SRIO_P2ROWS1R1)	32	R/W	0044_0000h	21.5.107/ 1296
D_0E38	Port 2 RapidIO outbound window segment 2 register n (SRIO_P2ROWS2R1)	32	R/W	0044_0000h	21.5.108/ 1298
D_0E3C	Port 2 RapidIO outbound window segment 3 register n (SRIO_P2ROWS3R1)	32	R/W	0044_0000h	21.5.109/ 1300
D_0E40	Port 2 RapidIO outbound window translation address register n (SRIO_P2ROWTAR2)	32	R/W	0000_0000h	21.5.103/ 1291
D_0E44	Port 2 RapidIO outbound window translation extended address register n (SRIO_P2ROWTEAR2)	32	R/W	0000_0000h	21.5.104/ 1292
D_0E48	Port 2 RapidIO outbound window base address register n (SRIO_P2ROWBAR2)	32	R/W	0000_0000h	21.5.105/ 1292
D_0E50	Port 2 RapidIO outbound window attributes register n (SRIO_P2ROWAR2)	32	R/W	0004_4027h	21.5.106/ 1293
D_0E54	Port 2 RapidIO outbound window segment 1 register n (SRIO_P2ROWS1R2)	32	R/W	0044_0000h	21.5.107/ 1296
D_0E58	Port 2 RapidIO outbound window segment 2 register n (SRIO_P2ROWS2R2)	32	R/W	0044_0000h	21.5.108/ 1298
D_0E5C	Port 2 RapidIO outbound window segment 3 register n (SRIO_P2ROWS3R2)	32	R/W	0044_0000h	21.5.109/ 1300
D_0E60	Port 2 RapidIO outbound window translation address register n (SRIO_P2ROWTAR3)	32	R/W	0000_0000h	21.5.103/ 1291
D_0E64	Port 2 RapidIO outbound window translation extended address register n (SRIO_P2ROWTEAR3)	32	R/W	0000_0000h	21.5.104/ 1292
D_0E68	Port 2 RapidIO outbound window base address register n (SRIO_P2ROWBAR3)	32	R/W	0000_0000h	21.5.105/ 1292
D_0E70	Port 2 RapidIO outbound window attributes register n (SRIO_P2ROWAR3)	32	R/W	0004_4027h	21.5.106/ 1293
D_0E74	Port 2 RapidIO outbound window segment 1 register n (SRIO_P2ROWS1R3)	32	R/W	0044_0000h	21.5.107/ 1296
D_0E78	Port 2 RapidIO outbound window segment 2 register n (SRIO_P2ROWS2R3)	32	R/W	0044_0000h	21.5.108/ 1298
D_0E7C	Port 2 RapidIO outbound window segment 3 register n (SRIO_P2ROWS3R3)	32	R/W	0044_0000h	21.5.109/ 1300

Table continues on the next page...

SRIO memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
D_0E80	Port 2 RapidIO outbound window translation address register n (SRIO_P2ROWTAR4)	32	R/W	0000_0000h	21.5.103/ 1291
D_0E84	Port 2 RapidIO outbound window translation extended address register n (SRIO_P2ROWTEAR4)	32	R/W	0000_0000h	21.5.104/ 1292
D_0E88	Port 2 RapidIO outbound window base address register n (SRIO_P2ROWBAR4)	32	R/W	0000_0000h	21.5.105/ 1292
D_0E90	Port 2 RapidIO outbound window attributes register n (SRIO_P2ROWAR4)	32	R/W	0004_4027h	21.5.106/ 1293
D_0E94	Port 2 RapidIO outbound window segment 1 register n (SRIO_P2ROWS1R4)	32	R/W	0044_0000h	21.5.107/ 1296
D_0E98	Port 2 RapidIO outbound window segment 2 register n (SRIO_P2ROWS2R4)	32	R/W	0044_0000h	21.5.108/ 1298
D_0E9C	Port 2 RapidIO outbound window segment 3 register n (SRIO_P2ROWS3R4)	32	R/W	0044_0000h	21.5.109/ 1300
D_0EA0	Port 2 RapidIO outbound window translation address register n (SRIO_P2ROWTAR5)	32	R/W	0000_0000h	21.5.103/ 1291
D_0EA4	Port 2 RapidIO outbound window translation extended address register n (SRIO_P2ROWTEAR5)	32	R/W	0000_0000h	21.5.104/ 1292
D_0EA8	Port 2 RapidIO outbound window base address register n (SRIO_P2ROWBAR5)	32	R/W	0000_0000h	21.5.105/ 1292
D_0EB0	Port 2 RapidIO outbound window attributes register n (SRIO_P2ROWAR5)	32	R/W	0004_4027h	21.5.106/ 1293
D_0EB4	Port 2 RapidIO outbound window segment 1 register n (SRIO_P2ROWS1R5)	32	R/W	0044_0000h	21.5.107/ 1296
D_0EB8	Port 2 RapidIO outbound window segment 2 register n (SRIO_P2ROWS2R5)	32	R/W	0044_0000h	21.5.108/ 1298
D_0EBC	Port 2 RapidIO outbound window segment 3 register n (SRIO_P2ROWS3R5)	32	R/W	0044_0000h	21.5.109/ 1300
D_0EC0	Port 2 RapidIO outbound window translation address register n (SRIO_P2ROWTAR6)	32	R/W	0000_0000h	21.5.103/ 1291
D_0EC4	Port 2 RapidIO outbound window translation extended address register n (SRIO_P2ROWTEAR6)	32	R/W	0000_0000h	21.5.104/ 1292
D_0EC8	Port 2 RapidIO outbound window base address register n (SRIO_P2ROWBAR6)	32	R/W	0000_0000h	21.5.105/ 1292
D_0ED0	Port 2 RapidIO outbound window attributes register n (SRIO_P2ROWAR6)	32	R/W	0004_4027h	21.5.106/ 1293
D_0ED4	Port 2 RapidIO outbound window segment 1 register n (SRIO_P2ROWS1R6)	32	R/W	0044_0000h	21.5.107/ 1296
D_0ED8	Port 2 RapidIO outbound window segment 2 register n (SRIO_P2ROWS2R6)	32	R/W	0044_0000h	21.5.108/ 1298
D_0EDC	Port 2 RapidIO outbound window segment 3 register n (SRIO_P2ROWS3R6)	32	R/W	0044_0000h	21.5.109/ 1300
D_0EE0	Port 2 RapidIO outbound window translation address register n (SRIO_P2ROWTAR7)	32	R/W	0000_0000h	21.5.103/ 1291

Table continues on the next page...

SRIO memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
D_0EE4	Port 2 RapidIO outbound window translation extended address register n (SRIO_P2ROWTEAR7)	32	R/W	0000_0000h	21.5.104/ 1292
D_0EE8	Port 2 RapidIO outbound window base address register n (SRIO_P2ROWBAR7)	32	R/W	0000_0000h	21.5.105/ 1292
D_0EF0	Port 2 RapidIO outbound window attributes register n (SRIO_P2ROWAR7)	32	R/W	0004_4027h	21.5.106/ 1293
D_0EF4	Port 2 RapidIO outbound window segment 1 register n (SRIO_P2ROWS1R7)	32	R/W	0044_0000h	21.5.107/ 1296
D_0EF8	Port 2 RapidIO outbound window segment 2 register n (SRIO_P2ROWS2R7)	32	R/W	0044_0000h	21.5.108/ 1298
D_0EFC	Port 2 RapidIO outbound window segment 3 register n (SRIO_P2ROWS3R7)	32	R/W	0044_0000h	21.5.109/ 1300
D_0F00	Port 2 RapidIO outbound window translation address register n (SRIO_P2ROWTAR8)	32	R/W	0000_0000h	21.5.103/ 1291
D_0F04	Port 2 RapidIO outbound window translation extended address register n (SRIO_P2ROWTEAR8)	32	R/W	0000_0000h	21.5.104/ 1292
D_0F08	Port 2 RapidIO outbound window base address register n (SRIO_P2ROWBAR8)	32	R/W	0000_0000h	21.5.105/ 1292
D_0F10	Port 2 RapidIO outbound window attributes register n (SRIO_P2ROWAR8)	32	R/W	0004_4027h	21.5.106/ 1293
D_0F14	Port 2 RapidIO outbound window segment 1 register n (SRIO_P2ROWS1R8)	32	R/W	0044_0000h	21.5.107/ 1296
D_0F18	Port 2 RapidIO outbound window segment 2 register n (SRIO_P2ROWS2R8)	32	R/W	0044_0000h	21.5.108/ 1298
D_0F1C	Port 2 RapidIO outbound window segment 3 register n (SRIO_P2ROWS3R8)	32	R/W	0044_0000h	21.5.109/ 1300
D_0F60	Port 2 RapidIO Inbound window translation address register n (SRIO_P2RIWTAR4)	32	R/W	0000_0000h	21.5.110/ 1302
D_0F68	Port 2 RapidIO Inbound window base address register n (SRIO_P2RIWBAR4)	32	R/W	0000_0000h	21.5.111/ 1303
D_0F70	Port 2 RapidIO inbound window attributes register n (SRIO_P2RIWAR4)	32	R/W	0004_4021h	21.5.112/ 1304
D_0F80	Port 2 RapidIO Inbound window translation address register n (SRIO_P2RIWTAR3)	32	R/W	0000_0000h	21.5.110/ 1302
D_0F88	Port 2 RapidIO Inbound window base address register n (SRIO_P2RIWBAR3)	32	R/W	0000_0000h	21.5.111/ 1303
D_0F90	Port 2 RapidIO inbound window attributes register n (SRIO_P2RIWAR3)	32	R/W	0004_4021h	21.5.112/ 1304
D_0FA0	Port 2 RapidIO Inbound window translation address register n (SRIO_P2RIWTAR2)	32	R/W	0000_0000h	21.5.110/ 1302
D_0FA8	Port 2 RapidIO Inbound window base address register n (SRIO_P2RIWBAR2)	32	R/W	0000_0000h	21.5.111/ 1303
D_0FB0	Port 2 RapidIO inbound window attributes register n (SRIO_P2RIWAR2)	32	R/W	0004_4021h	21.5.112/ 1304

Table continues on the next page...

SRIO memory map (continued)

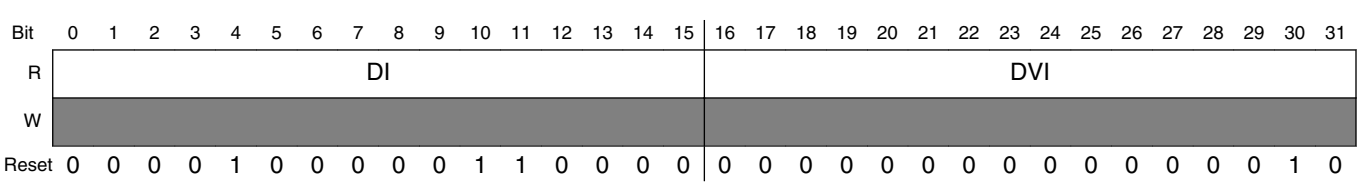
Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
D_0FC0	Port 2 RapidIO Inbound window translation address register n (SRIO_P2RIWTAR1)	32	R/W	0000_0000h	21.5.110/ 1302
D_0FC8	Port 2 RapidIO Inbound window base address register n (SRIO_P2RIWBAR1)	32	R/W	0000_0000h	21.5.111/ 1303
D_0FD0	Port 2 RapidIO inbound window attributes register n (SRIO_P2RIWAR1)	32	R/W	0004_4021h	21.5.112/ 1304
D_0FE0	Port 2 RapidIO inbound window translation address register 0 (SRIO_P2RIWTAR0)	32	R/W	0000_0000h	21.5.113/ 1306
D_0FF0	Port 2 RapidIO inbound window attributes register 0 (SRIO_P2RIWAR0)	32	R/W	8004_4021h	21.5.114/ 1307
D_3000	Port 1 LIODN Base Register (SRIO_P1LBR)	32	R/W	0000_0000h	21.5.115/ 1309
D_3020	Port 1 LIODN Address Offset Register (SRIO_P1LAOR)	32	R/W	0000_0000h	21.5.116/ 1310
D_3030	Port 1 LIODN upper data Register (SRIO_P1LUDR)	32	R/W	0000_0000h	21.5.117/ 1310
D_3034	Port 1 LIODN lower data register (SRIO_P1LLDR)	32	R/W	0000_0000h	21.5.118/ 1311
D_3200	Port 2 LIODN Base Register (SRIO_P2LBR)	32	R/W	0000_0000h	21.5.119/ 1312
D_3220	Port 2 LIODN Address Offset Register (SRIO_P2LAOR)	32	R/W	0000_0000h	21.5.120/ 1312
D_3230	Port 2 LIODN upper data Register (SRIO_P2LUDR)	32	R/W	0000_0000h	21.5.121/ 1313
D_3234	Port 2 LIODN lower data Register (SRIO_P2LLDR)	32	R/W	0000_0000h	21.5.122/ 1314

21.5.1 Device identity capability register (SRIO_DIDCAR)

The device vendor identity field (DVI) identifies the vendor that manufactured the device containing the processing element. A value for DVI is uniquely assigned to a device vendor by the registration authority of the RapidIO Trade Association.

The device identity field (DI) is intended to uniquely identify the type of device from the vendor specified by DVI. The values for DI are assigned and managed by the respective vendor.

Address: C_0000h base + 0h offset = C_0000h



SRIO_DIDCAR field descriptions

Field	Description
0–15 DI	Device identity 0x0830 (T2080 with security) 0x0831 (T2080 without security) 0x0838 T2081 with security 0x0839 T2081 without security
16–31 DVI	Device vendor identity 0x0002 (Freescale)

21.5.2 Device information capability register (SRIO_DICAR)

The device revision field (DR) is intended to identify the revision level of the device. The value for DR is assigned and managed by the vendor specified by DIDCAR[DVI]. DICAR represents a copy of the device's system version register (SVR). See Section SVR for more information.

Address: C_0000h base + 4h offset = C_0004h

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	DR																															
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

SRIO_DICAR field descriptions

Field	Description
0–31 DR	Device revision. This is a copy of the device's system version register (SVR). See SVR for additional information.

21.5.3 Assembly identity capability register (SRIO_AIDCAR)

The assembly vendor identity field (AVI) identifies the vendor that manufactured the assembly or subsystem containing the device. A value for AVI is uniquely assigned to an assembly vendor by the registration authority of the RapidIO Trade Association.

The assembly identity field (AI) is intended to uniquely identify the type of assembly from the vendor specified by AVI. The values for AI are assigned and managed by the respective vendor.

Address: C_0000h base + 8h offset = C_0008h

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	AI																AVI															
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

SRIO_AIDCAR field descriptions

Field	Description
0–15 AI	Assembly identity (all zeros) **NOTE:** This field is write-able until boot completion.
16–31 AVI	Assembly vendor identity (all zeros) **NOTE:** This field is write-able until boot completion.

21.5.4 Assembly information capability register (SRIO_AICAR)

AICAR contains additional information about the assembly and the pointer to the first entry in the extended features list.

Address: C_0000h base + Ch offset = C_000Ch

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	AR																EFP															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

SRIO_AICAR field descriptions

Field	Description
0–15 AR	AssyRev field (all zeros) **NOTE:** This field is write-able until boot completion.
16–31 EFP	ExtendedFeaturesPtr field (0x100)

21.5.5 Processing element features capability register (SRIO_PEFCAR)

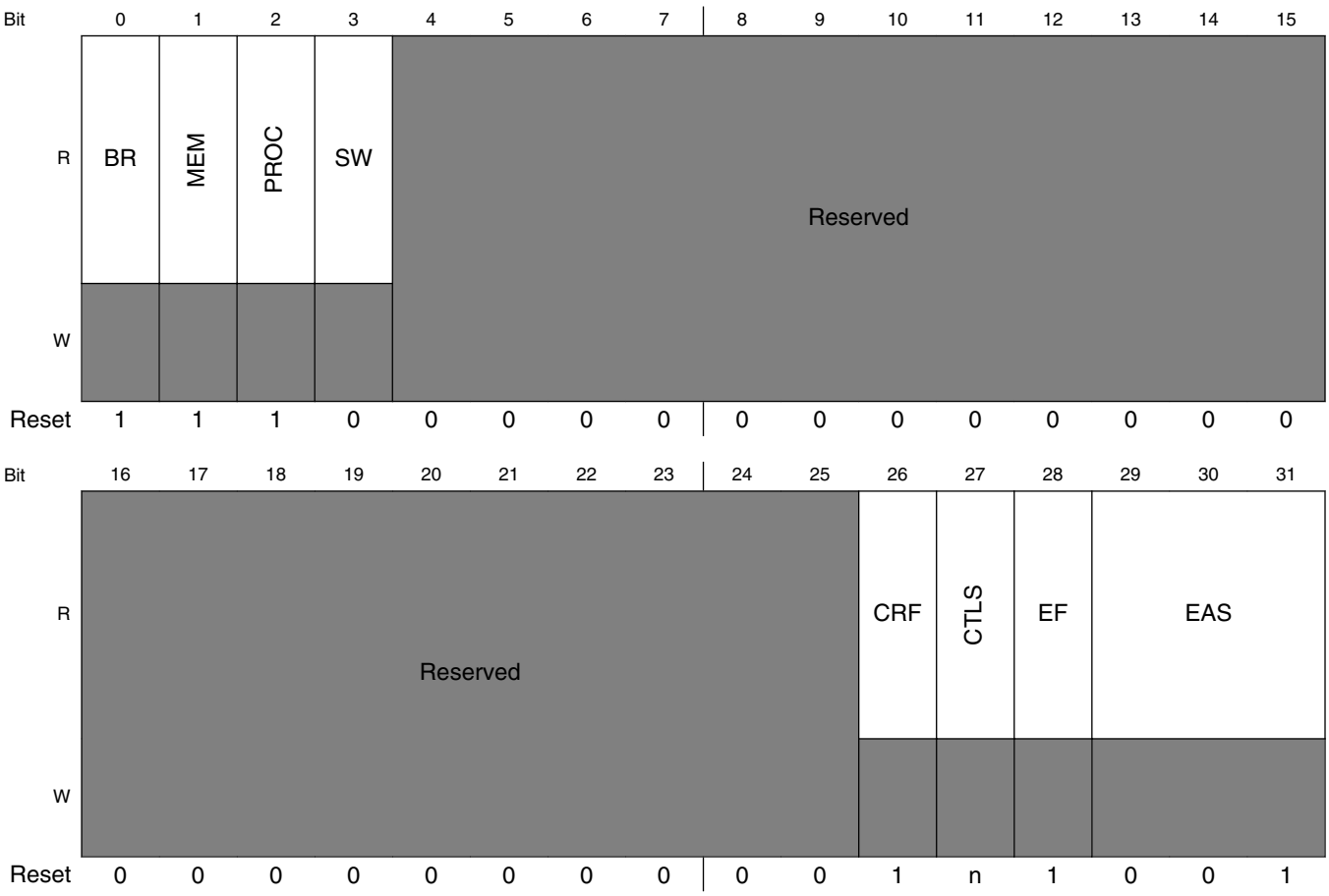
The processing element features capability register (PEFCAR), identifies the major functionality provided by the processing element. PEFCAR is a read-only register

NOTE

Values indicated with n are determined at power-on reset. See Section 4.4.3.12, "RapidIO System Size," for POR configuration details.

.

Address: C_0000h base + 10h offset = C_0010h



SRIO_PEFCAR field descriptions

Field	Description
0 BR	Bridge. PE can bridge to another interface. (BR = 1)
1 MEM	Memory. PE has physically addressable local address space and can be accessed as an endpoint through non-maintenance operations. (MEM = 1)
2 PROC	Processor. PE physically contains a local processor that executes code. (PROC = 1)
3 SW	Switch. PE does not bridge to another external RapidIO interface. (SW = 0)
4–25 -	This field is reserved. Reserved

Table continues on the next page...

SRIO_PEFCAR field descriptions (continued)

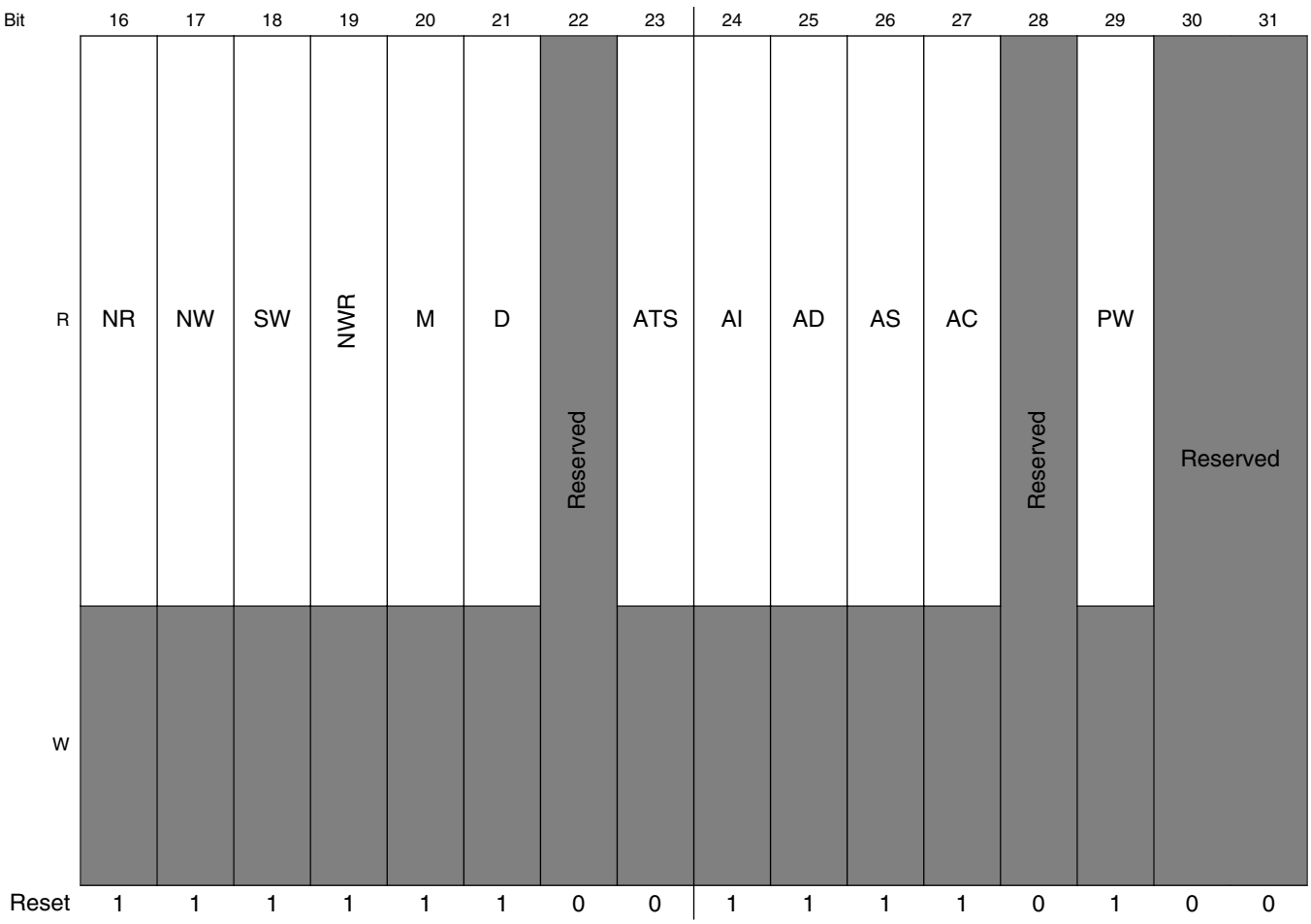
Field	Description
26 CRF	Critical request flow. PE supports the critical request flow (CRF) function. (CRF = 1)
27 CTLS	This bit reflects the selected RapidIO common transport system size. The RapidIO common transport system size is determined at power-on reset. 0 PE only supports common transport small systems (up to 256 devices). 1 PE supports common transport large systems (up to 65,536 devices).
28 EF	Extended features pointer is valid. (EF = 1)
29–31 EAS	Indicates the number of address bits supported by the PE both as a source and target of an operation. EAS = 001(34-bit addresses)

21.5.6 Source operations capability register (SRIO_SOCAR)

The source operations capability register (SOCAR), defines the set of RapidIO I/O logical operations that can be issued by this processing element. SOCAR is a read-only register. SOCAR settings are configurable via pins and can be modified. Must add note to NOT modify these bits to something other than their settings below.

Address: C_0000h base + 18h offset = C_0018h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	R	IR	RO	DCI	C	F	IOR	ICI	TIE	TIES	Reserved		DSTM	DS	Reserved	
W																
Reset	0	0	0	0	0	1	1	0	0	0	0	0	1	1	0	0



SRIO_SOCAR field descriptions

Field	Description
0 R	PE does not support a Read operation. (R = 0)
1 IR	PE does not support an IRead operation. (IR = 0)
2 RO	PE does not support a Read_To_Own operation. (RO = 0)
3 DCI	PE does not support a Data Cache Invalidate operation. (DCI = 0)
4 C	PE does not support a Castout operation. (C = 0)
5 F	PE supports a Flush operation. (F = 1)
6 IOR	PE supports an I/O-Read operation. (IOR = 1)
7 ICI	PE does not support an Instruction Cache Invalidate operation. (ICI = 0)
8 TIE	PE does not support a TLBIE operation. (TIE = 0)

Table continues on the next page...

SRIO_SOCAR field descriptions (continued)

Field	Description
9 TIES	PE does not support a TLBSYNC operation. (TIES = 0)
10–11 -	This field is reserved. Reserved
12 DSTM	PE can support data streaming traffic management. (DSTM = 1)
13 DS	PE can support a data streaming operation. (DS = 1)
14–15 -	This field is reserved. Reserved
16 NR	PE supports a Nread operation. (NR = 1)
17 NW	PE supports a Nwrite operation. (NW = 1)
18 SW	PE supports an Swrite operation. (SW = 1)
19 NWR	PE supports a Nwrite_R operation. (NWR = 1)
20 M	PE supports a Message operation. (M = 1)
21 D	PE supports a Doorbell operation. (D = 1)
22 -	This field is reserved. Reserved
23 ATS	PE does not support an Atomic-Test-and-Swap operation. (ATS = 0)
24 AI	PE supports an Atomic-Inc operation. (AI = 1)
25 AD	PE supports an Atomic-Dec operation. (AD = 1)
26 AS	PE supports an Atomic-Set operation. (AS = 1)
27 AC	PE supports an Atomic-Clr operation. (AC = 1)
28 -	This field is reserved. Reserved
29 PW	PE supports a Port-Write operation. (PW = 1)
30–31 -	This field is reserved. Reserved

21.5.7 Destination operations capability register (SRIO_DOCAR)

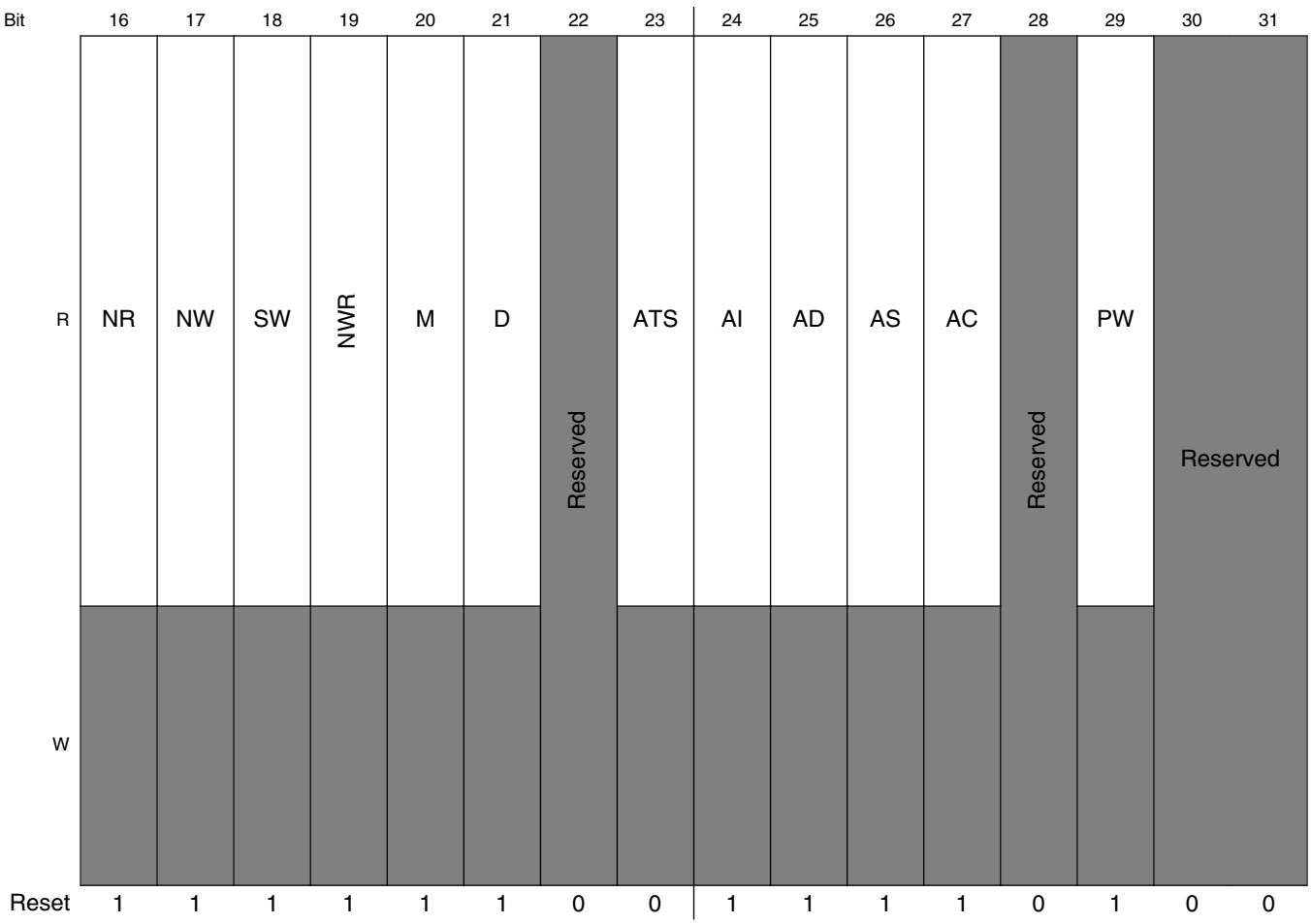
The destination operations capability register (DOCAR), defines the set of RapidIO I/O operations that can be supported by this processing element. DOCAR is a read-only register. DOCAR settings are configurable via pins and can be modified.

NOTE

Do not modify these bits to something other than the settings stated in Table 19-12, because unsupported transaction errors only apply to atomic transactions.

Address: C_0000h base + 1Ch offset = C_001Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	R	IR	RO	DCI	C	F	IOR	ICI	TIE	TIES	Reserved		DSTM	DS	Reserved	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0



SRIO_DOCAR field descriptions

Field	Description
0 R	PE does not support a Read operation. (R = 0)
1 IR	PE does not support an IRead operation. (IR = 0)
2 RO	PE does not support a Read_To_Own operation. (RO = 0)
3 DCI	PE does not support a Data Cache Invalidate operation. (DCI = 0)
4 C	PE does not support a Castout operation. (C = 0)
5 F	PE does not support a Flush operation. (F = 0)
6 IOR	PE does not support an I/O-Read operation. (IOR = 0)
7 ICI	PE does not support an Instruction Cache Invalidate operation. (ICI = 0)
8 TIE	PE does not support a TLBIE operation. (TIE = 0)

Table continues on the next page...

SRIO_DOCAR field descriptions (continued)

Field	Description
9 TIES	PE does not support a TLBSYNC operation. (TIES = 0)
10–11 -	This field is reserved. Reserved
12 DSTM	PE can support data streaming traffic management. (DSTM = 1)
13 DS	PE can support a data streaming operation. (DS = 1)
14–15 -	This field is reserved. Reserved
16 NR	PE supports a Nread operation. (NR = 1)
17 NW	PE supports a Nwrite operation. (NW = 1)
18 SW	PE supports an Swrite operation. (SW = 1)
19 NWR	PE supports a Nwrite_R operation. (NWR = 1)
20 M	PE supports a Message operation. (M = 1)
21 D	PE supports a Doorbell operation. (D = 1)
22 -	This field is reserved. Reserved
23 ATS	PE does not support an Atomic-Test-and-Swap operation. (ATS = 0)
24 AI	PE supports an Atomic-Inc operation. (AI = 1)
25 AD	PE supports an Atomic-Dec operation. (AD = 1)
26 AS	PE supports an Atomic-Set operation. (AS = 1)
27 AC	PE supports an Atomic-Clr operation. (AC = 1)
28 -	This field is reserved. Reserved
29 PW	PE supports a Port-Write operation. (PW = 1)
30–31 -	This field is reserved. Reserved

21.5.8 Data streaming information capability register (SRIO_DSICAR)

The data streaming information capability register (DSICAR), defines the data streaming capabilities of a processing element. It is required for destination end point devices. DSICAR is a read-only register.

Address: C_0000h base + 3Ch offset = C_003Ch

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	MaxPDU																SegSupport															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0

SRIO_DSICAR field descriptions

Field	Description
0–15 MaxPDU	Maximum PDU - The maximum PDU size in bytes supported by the destination end point (MaxPDU = 16'h0000). 0x0000 64 KBytes
16–31 SegSupport	Segment Support - The number of segmentation contexts supported by the destination end point (SegSupport = 16'h0018). 0x0018 24 segmentation contexts

21.5.9 Data streaming logic layer command and status register (SRIO_DSLLCSR)

The data streaming logic layer control command and status register (DSLLCCSR), is used for general command and status information for the logical interface.

Address: C_0000h base + 48h offset = C_0048h

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	TMTypes_ Support				TMMode				Reserved																MTU							
W																																
Reset	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	1	0	0	0	0	0	0

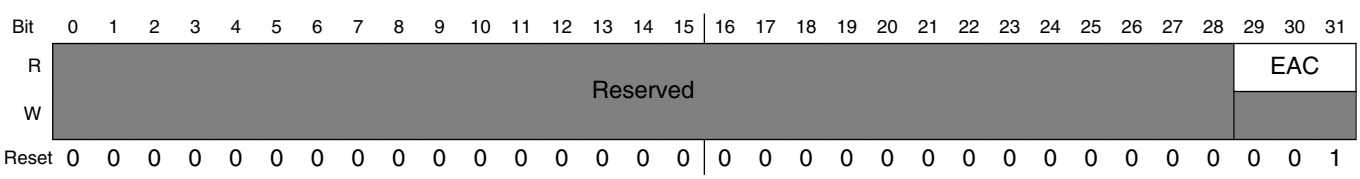
SRIO_DSLLCSR field descriptions

Field	Description
0–3 TMTypes_ Support	The traffic management types supported (this field is read-only). Only basic traffic management type is supported. 0x8 Basic type supported
4–7 TMMode	Traffic Management Mode of operation 0x0 Traffic management disabled 0x1 Basic 0x2 Reserved 0xF Reserved
8–23 -	This field is reserved. Reserved
24–31 MTU	Maximum Transmission Unit Controls the data payload size for segments of an encapsulated PDU. Only single segment PDUs and end segments are permitted to have a data payload that is less this value. The MTU can be specified in increments of 4 bytes. Support for the entire range is required.i 0x00 Reserved 0x07 Reserved 0x08 32 byte block size 0x09 36 byte block size 0x0A 40 byte block size 0x40 256 byte block size 0x41 Reserved 0xFF Reserved

21.5.10 Processing element logical layer control command and status register (SRIO_PELLCCSR)

The processing element logical layer control command and status register (PELLCCSR), controls the extended addressing abilities. The RapidIO endpoint only supports 34-bit addressing. PELLCCSR is a read-only register.

Address: C_0000h base + 4Ch offset = C_004Ch



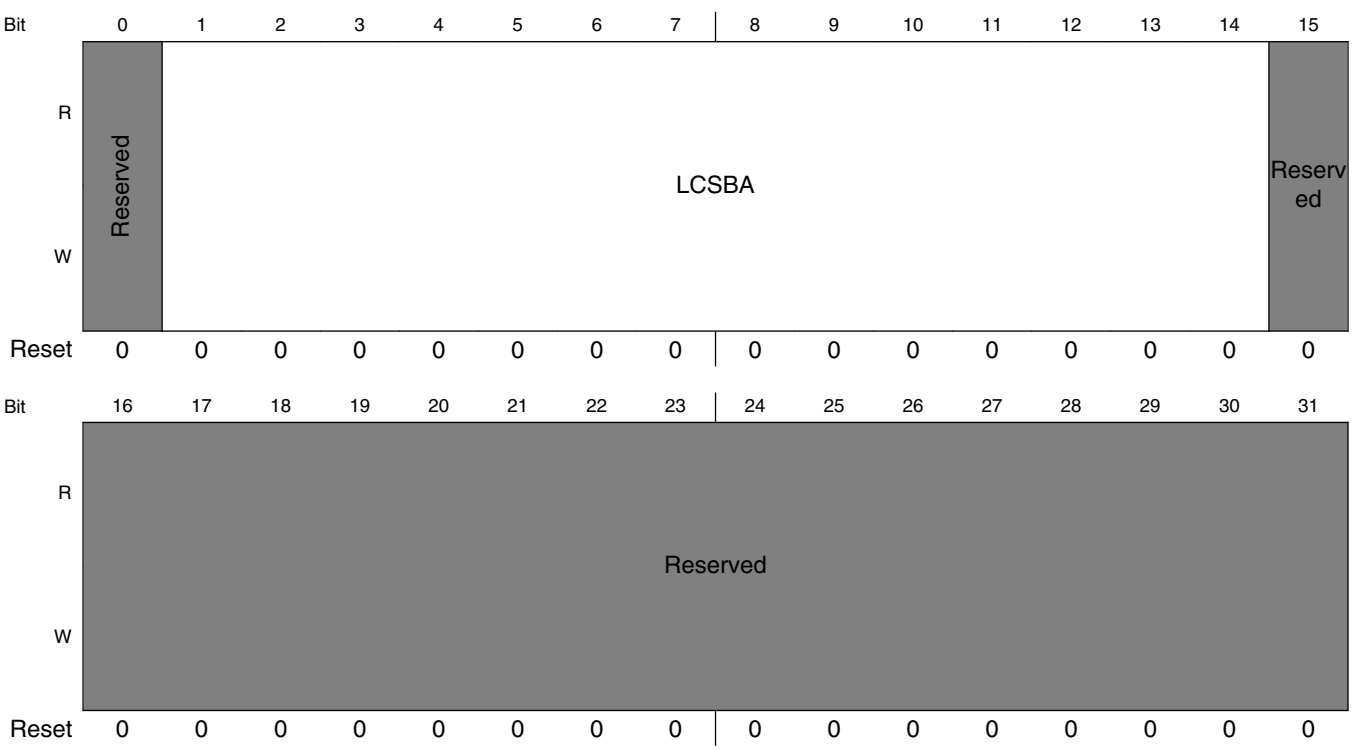
SRIO_PELLCCSR field descriptions

Field	Description
0–28 -	This field is reserved. Reserved
29–31 EAC	Extended addressing control. Read-only value is 0b001.

21.5.11 Local configuration space base address 1 command and status register (SRIO_LCSBA1CSR)

The local configuration space base address 1 command and status register (LCSBA1CSR), specifies bits of the local physical address double-word offset for the processing element's configuration register space, allowing the configuration register space to be physically mapped in the processing element. This register allows configuration and maintenance of a processing element through regular read and write operations rather than maintenance operations. The double-word offset is right-justified in the register. This window has priority over all ATMU windows. As is the case with all registers, an external processor writing to LCSBA1CSR should not assume it has been written until a response has been received.

Address: C_0000h base + 5Ch offset = C_005Ch



SRIO_LCSBA1CSR field descriptions

Field	Description
0 -	This field is reserved. Reserved
1–14 LCSBA	Local configuration space base address. These bits correspond to the highest 14 bits of the 34-bit RapidIO address space.
15–31 -	This field is reserved. Reserved

21.5.12 Base device ID command and status register (SRIO_BDIDCSR)

BDIDCSR, contains the base device ID values for the processing element.

Values indicated with x are determined at power-on reset. Values indicated with n are read from configuration signals, sampled at reset.

Address: C_0000h base + 60h offset = C_0060h

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								BDID								LBDID															
Reset	0	0	0	0	0	0	0	0	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n

SRIO_BDIDCSR field descriptions

Field	Description
0–7 -	This field is reserved. Reserved
8–15 BDID	Base device ID of the device in a small common transport system (RapidIO device ID). If RapidIO is configured as a host, BDID = {0b0000_0, RCW[RIO_DEVICE_ID](260-262)}. If RapidIO is configured as an agent, BDID = {0b1111_111, RCW[RIO_DEVICE_ID](262)}. See for details.)
16–31 LBDID	Large base device ID of the device in a large common transport system. This field is valid only if PEFCAR[CTLS] is set. If RapidIO is configured as a host, LBDID = {0b0000_0000_0000_0, RCW[RIO_DEVICE_ID](260-262)}. If RapidIO is configured as an agent, LBDID = {0b1111_1111_1111_111, RCW[RIO_DEVICE_ID](262)}. See for details.)

21.5.13 Host base device ID lock command and status register (SRIO_HBDIDLCSR)

The host base device ID lock command and status register (HBDIDLCSR) contains the base device ID value for the processing element in the system that is responsible for initializing this processing element. The HBDID field is a write-once/resettable field which provides a lock function. Once HBDID is written, all subsequent writes to the field are ignored, except in the case that the value written matches the value contained in the field. In this case, the register is re-initialized to 0xFFFF. After writing HBDID, a processing element must then read the host base device ID lock CSR to verify that it owns the lock before attempting to initialize this processing element.

Address: C_0000h base + 68h offset = C_0068h

Bit	0–15	16–31
R W	Reserved	HBDID
Reset	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

SRIO_HBDIDLCSR field descriptions

Field	Description
0–15 -	This field is reserved. Reserved
16–31 HBDID	This is the host base device ID for the processing element that is responsible for initializing this device. Only the first write to this field is accepted; all other writes are ignored, except in the case that the value written matches the value contained in the field. In this case, the register is re-initialized to 0xFFFF.

21.5.14 Component tag command and status register (SRIO_CTCSR)

The component tag command and status register (CTCSR), contains a component tag value for the processing element and can be assigned by software when the device is initialized. It is unused internally in the RapidIO endpoint.

Address: C_0000h base + 6Ch offset = C_006Ch

Bit	0–15	16–31
R W	CT	
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

SRIO_CTCSR field descriptions

Field	Description
0–31 CT	Component tag

21.5.15 Port maintenance block header 0 (SRIO_PMBH0)

The port maintenance block header 0 register (PMBH0), contains a pointer to the next EF_BLK (Extended Features Space, Error Management) and the EF_ID that identifies this as the generic end point port maintenance block header. Note that while registers defined by software-assisted error recovery are supported, software-assisted error recovery is not (these registers are included for hot insertion only); therefore, the RapidIO endpoint is defined here as not supporting software-assisted error recovery. PMBH0 is a read-only register.

Address: C_0000h base + 100h offset = C_0100h

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	EF_PTR																EF_ID															
W																																
Reset	0	0	0	0	0	1	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	1

SRIO_PMBH0 field descriptions

Field	Description
0–15 EF_PTR	Extended features pointer
16–31 EF_ID	Extended features ID

21.5.16 Port link time-out control command and status register (SRIO_PLTOCCSR)

The port link time-out control command and status register (PLTOCCSR), contains the link time-out value for all ports on a device. This time-out is for link events such as sending a packet to receiving the corresponding acknowledge and sending a link-request to receiving the corresponding link-response. The reset value is the maximum time-out interval. The timer resolution is specified between 178-298ns, which correlates to a maximum time-out value of approximately 3-5s. Minimal programmable TV value supported is 0x200

.

Address: C_0000h base + 120h offset = C_0120h

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	TV																Reserved															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0

SRIO_PLTOCCSR field descriptions

Field	Description
0–23 TV	Time-out value. Setting to all zeros disables the link time-out timer. This value is loaded each time the link time-out timer starts.
24–31 -	This field is reserved. Reserved

21.5.17 Port response time-out control command and status register (SRIO_PRTOCCSR)

The port response time-out control command and status register (PRTOCCSR), contains the time-out timer count for all ports on a device. This time-out is for sending a request packet to receiving the corresponding response packet.The reset value is the maximum time-out interval. The timer resolution is specified between 178-298ns, which correlates to a maximum time-out value of approximately 3-5s. Note that this applies to the RapidIO endpoint and the messaging unit.

Address: C_0000h base + 124h offset = C_0124h

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	TV																Reserved															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0

SRIO_PRTOCCSR field descriptions

Field	Description
0–23 TV	Time-out value. Setting to all zeros disables the response time-out timer. This value is loaded each time the response time-out timer starts.
24–31 -	This field is reserved. Reserved

21.5.18 Port General control command and status register (SRIO_GCCSR)

The port general control command and status register (PGCCSR), contains control register bits applicable to the RapidIO interface.

NOTE

Values indicated with n are read from configuration signals, sampled at reset.

Address: C_0000h base + 13Ch offset = C_013Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	H	M	D	Reserved												
Reset	n	n	n	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

SRIO_GCCSR field descriptions

Field	Description
0 H	Host. The reset value of this bit is assigned by the RCW fields HOST_AGT_B1, HOST_AGT_B3, which determine host/agent configuration for the device. See RCW Field Definitions for more information. A host device is a device that is responsible for system exploration, initialization, and maintenance. Agent or slave devices are initialized by host devices. (Note that although this status bit is Read/Write, manually changing its value does not affect logical operation.) 0 Agent device 1 Host device
1 M	Master. The reset value of this bit is assigned by the RCW fields HOST_AGT_B1, HOST_AGT_B3, which determine host/agent configuration for the device. See RCW Field Definitions for more information. This bit controls whether or not a device is allowed to issue requests into the system. If M = 0, the device may only respond to requests or send Type 9 Flow Control Packets. Software must ensure no requests are sent to the device by disabling any local access windows and disabling inbound ATMU windows from other devices targeting SRIO while M=0. 0 Device is not enabled to issue requests into the system. 1 Device is enabled to issue requests into the system.
2 D	Discovered. The reset value of this bit is identical to PGCCSR[H] which is assigned by the RCW fields HOST_AGT_B1, HOST_AGT_B3, which determine host/agent configuration for the device. Master. The reset value of this bit is identical to PGCCSR[H] which is assigned by the RCW fields HOST_AGT_B1, HOST_AGT_B3, which determine host/agent configuration for the device. See for more information. This bit controls whether or not a device is allowed to issue requests into the system. If M = 0, the device may

Table continues on the next page...

SRIO_GCCSR field descriptions (continued)

Field	Description
	only respond to requests. Software must ensure no requests are sent to the device by disabling any local access windows and disabling inbound ATMU windows from other devices targeting SRIO while M=0. 0 Device is not enabled to issue requests into the system. This device has been located by the processing element responsible for system configuration. (Note that although this status bit is Read/Write, manually changing its value does not affect logical operation.) 0 Device has not been discovered by system host. 1 Device has been discovered by system host.
3–31 -	This field is reserved. Reserved

21.5.19 Port 1 Link maintenance request command and status register (SRIO_P1LMREQCSR)

The port 1 link maintenance request command and status register (P1LMREQCSR), is accessible both by the local processor and an external device. A write to this register generates a link-request control symbol on the RapidIO endpoint port 1 interface. Care should be taken when writing this register that it is only used for hot swap and not for software-assisted error recovery (which is not supported).

Address: C_0000h base + 140h offset = C_0140h

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																													C		
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

SRIO_P1LMREQCSR field descriptions

Field	Description
0–28 -	This field is reserved. Reserved
29–31 C	LINK_REQUEST command to send. If read, this field returns the last written value. If written with a value other than 0b011 (reset-device) or 0b100 (input-status), the resulting operation is undefined, as all other values are reserved in the RapidIO specification.

21.5.20 Port 1 Link maintenance response command and status register (SRIO_P1LMRESPCSR)

The port 1 link maintenance response command and status register (P1LMRESPCSR), is accessible both by the local processor and an external device. A read to this register returns the status received in a link-response control symbol. This register is read only.

Address: C_0000h base + 144h offset = C_0144h

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

SRIO_P1LMRESPCSR field descriptions

Field	Description
0 RV	Response valid • If the link-request causes a link-response, this bit indicates that the link-response has been received and the status fields are valid. • If the link-request does not cause a link-response, this bit indicates that the link-request has been transmitted (clears on read)
1–21 -	This field is reserved. Reserved
22–26 AS	AckID_status field from LINK_RESPONSE.
27–31 LS	Link_status field from LINK_RESPONSE

21.5.21 Port 1 Local ackID status command and status register (SRIO_P1LASCSR)

The port 1 local ackID status command and status register (P1LASCSR), is accessible both by the local processor and an external device. A read to this register returns the local ackID status for both the output and input ports of the device. Care should be taken to use this register only for hot swap and not software error management.

Address: C_0000h base + 148h offset = C_0148h

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			IA					Reserved											OUTA					Reserved			OBA				
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

SRIO_P1LASCSR field descriptions

Field	Description
0–2 -	This field is reserved. Reserved
3–7 IA	Inbound ackID: Input port next expected ackID value.
8–18 -	This field is reserved. Reserved
19–23 OUTA	Outstanding port unacknowledged ackID status. Next expected acknowledge control symbol ackID field that indicates the ackID value expected in the next received acknowledge control symbol. Note that this value is read only even though RapidIO specification allows for it to be writable.
24–26 -	This field is reserved. Reserved
27–31 OBA	Outbound ackID output port next transmitted ackID value. This can be written by software but only if there are no outstanding unacknowledged packets. If there are, the newly-written value is ignored.

21.5.22 Port 1 Error and status command and status register (SRIO_P1ESCSR)

The port 1 error and status command and status register (P1ESCSR), is accessed when the local processor or an external device wishes to examine the port error and status information.

Address: C_0000h base + 158h offset = C_0158h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Reserved					OPD	OFE	ODE	Reserved			ORE	OR	ORS	OEE	OES
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					IRS	IEE	IES	Reserved			PWP	Reserved	PE	PO	PU
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

SRIO_P1ESCSR field descriptions

Field	Description
0–4 -	This field is reserved. Reserved
5 OPD	Output Packet-dropped. Output port has discarded a packet. A packet is discarded if: • It is received while OFE is set and PnCCSR[DPE] (drop packet enable) is set and PnCCSR[SPF] (stop on port failed) is set. • It is received while PnPCR[OBDEN] (output buffer drain enable) is set. • It is not-accepted by the link-partner while PnERCSR[ERFTT] (error rate failed threshold trigger) is met or exceeded and PnCCSR[DPE] is set and PnCCSR[SPF] is not set (and link-response returns expected ackID). Once OPD is set, it remains set until written with a logic 1 to clear.

Table continues on the next page...

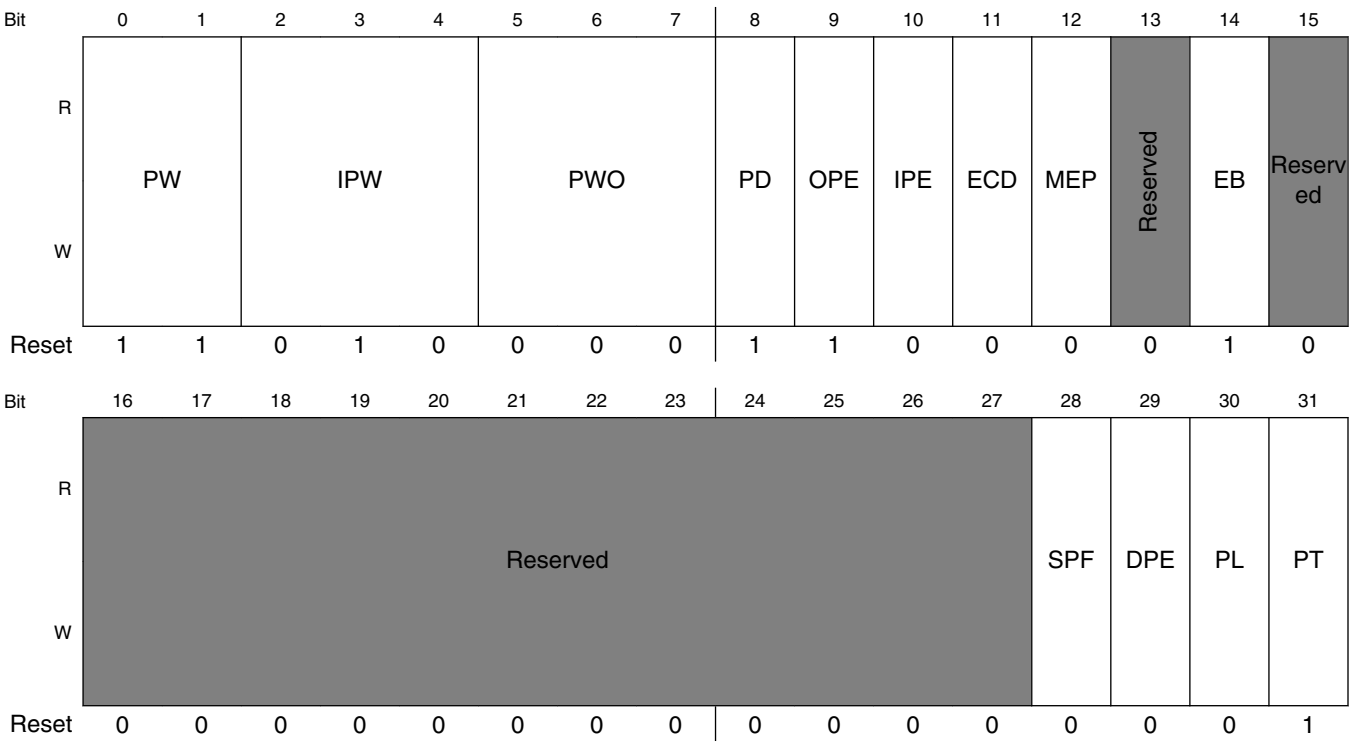
SRIO_P1ESCSR field descriptions (continued)

Field	Description
6 OFE	Output Failed-encountered. Output port has encountered a failed condition, meaning that the Error Rate Counter has met or exceeded the port's failed error threshold (ERFTT) Once set, remains set until written with a logic 1 to clear. Once cleared, does not assert again unless the Error Rate Counter dips below the port's failed error threshold and then meets or exceeds it again.
7 ODE	Output port has encountered a degraded condition, meaning that the Error Rate Counter has met or exceeded the port's degraded error threshold. Once set, remains set until written with a logic 1 to clear. Once cleared, does not assert again unless the Error Rate Counter dips below the port's degraded error threshold and then meets or exceeds it again.
8–10 -	This field is reserved. Reserved
11 ORE	Output port has encountered a retry condition. This bit is set when bit 13 is set. Once set, remains set until written with a logic 1 to clear.
12 OR	Output port has received a packet retry control symbol and cannot make forward progress. This bit is set when bit 13 is set and cleared when a packet-accepted or packet-not-accepted control symbol is received. (read only)
13 ORS	Output port is stopped due to a retry (read only)
14 OEE	Output port has encountered (and possibly recovered from) a transmission error. This bit is set when bit 15 is set. Once set, remains set until written with a logic 1 to clear.
15 OES	Output port is stopped due to a transmission error (read only)
16–20 -	This field is reserved. Reserved
21 IRS	Input port is stopped due to a retry (read only)
22 IEE	Input port has encountered (and possibly recovered from) a transmission error. This bit is set when bit 23 is set. Once set, remains set until written with a logic 1 to clear.
23 IES	Input port is stopped due to a transmission error (read-only)
24–26 -	This field is reserved. Reserved
27 PWP	Port has encountered a condition which required it to initiate a maintenance port-write operation. This bit is only valid if the device is capable of issuing an auto-generated maintenance port-write transaction. The RapidIO endpoint is not capable of issuing auto-generated port-writes. This bit is hardwired to 0.
28 -	This field is reserved. Reserved
29 PE	Input or output port has encountered an error from which hardware was unable to recover. Once set, remains set until written with a logic 1 to clear. This bit indicates that OFE is set while PnCCSR[SPF] is set; in other words, the failed threshold has been reached which has caused the output port to stop transmitting packets.
30 PO	The input and output ports are initialized and the port is exchanging error-free control symbols with the attached device. (read-only).
31 PU	Input and output ports are not initialized. This bit and bit 30 are mutually exclusive (read-only).

21.5.23 Port 1 Control command and status register (SRIO_P1CCSR)

The port 1 control command and status register (P1CCSR), contains control register bits for the RapidIO port.

Address: C_0000h base + 15Ch offset = C_015Ch



SRIO_P1CCSR field descriptions

Field	Description
0–1 PW	Indicates port width modes supported by the port (read-only). The bits of this field indicates the port width modes supported by the port in addition to 1x mode, which is required. 0x 4x mode not supported 1x 4x mode supported x0 2x mode not supported x1 2x mode supported
2–4 IPW	Width of the ports after initialized (read-only). 000 Single-lane port, lane 0 001 Single-lane port, lane R 010 Four-lane port, lanes 0-3 011 Two-lane port, lanes 0-1 100-111 Reserved

Table continues on the next page...

SRIO_P1CCSR field descriptions (continued)

Field	Description
5–7 PWO	Soft port configuration to control the width modes available for port initialization. The meaning of bits 6 and 7 depend on the value of bit 5. When bit [5] = 0b1, bit 6 controls the enabling of 4x mode, bit 7 controls the enabling of 2x mode, A change in the value of this field shall cause the port to re-initialize using the new field value. This field should be changed only when the port is uninitialized. To achieve this, first disable the RapidIO port. Then change PWO to any valid value. Finally, re-enable the RapidIO port. To achieve this, first set PnCCSR[PD] (port disabled). Then change PWO to any legal value. Finally, clear PnCCSR[PD] (enabled). 000 No override, all supported mode widths enabled. 001 Reserved 010 Force single lane, lane R not forced 011 Force single lane, force lane R 100 Reserved 101 2x mode enabled, 4x mode disabled 110 2x mode disabled, 4x mode enabled 111 2x mode enabled, 4x mode enabled
8 PD	Port disable. 0b0 - port receivers/drivers are enabled 0b1 - port receivers/drivers are disabled and are unable to receive/transmit If set, the output will congest if packets are sent to it.
9 OPE	Output port transmit enable. Initial value read from configuration pins. 0 Port is stopped and not enabled to issue any packets except to route or respond to I/O logical MAINTENANCE packets. Control symbols are not affected and are sent normally. 1 Port is enabled to issue any packets.
10 IPE	Input port receive enable. 0 Port is stopped and only enabled to route or respond to I/O logical MAINTENANCE packets. Other packets generate packet-not-accepted control symbols to force an error condition to be signaled by the sending device. Control symbols are not affected and are received and handled normally. 1 Port is enabled to respond to any packet.
11 ECD	Error checking disable. This bit is hardwired to 0. This bit disables all RapidIO transmission error checking This bit value must equal the value of the output port enable (OPE) bit in order for the RapidIO controller to function properly. Initial value read from configuration pins. 0 Error checking and recovery is enabled. 1 Error checking and recovery is disabled.

Table continues on the next page...

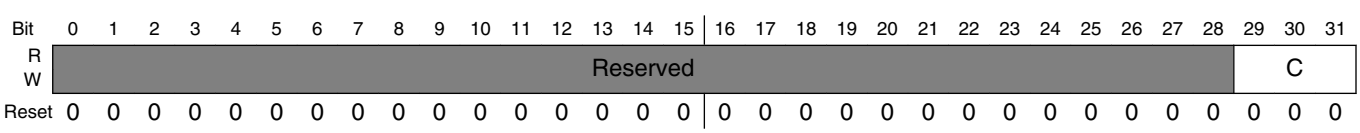
SRIO_P1CCSR field descriptions (continued)

Field	Description
12 MEP	Multicast-event participant. This bit is hard-wired to 0.
13 -	This field is reserved. Reserved
14 EB	Enumeration boundary. An enumeration boundary aware system enumeration algorithm shall honor this flag. The algorithm, on either the ingress or the egress port, shall not enumerate past a port with this bit set.
15–27 -	This field is reserved. Reserved
28 SPF	Stop on port failed-encountered enable. This bit is used with the Drop Packet Enable bit to force certain behavior when the Error Rate Failed Threshold has been met or exceeded.
29 DPE	Drop packet enable. This bit is used with the Stop on Port Failed-encountered Enable bit to force certain behavior when the Error Rate Failed Threshold has been met or exceeded.
30 PL	Port lockout. 0 The packets that may be received and issued are controlled by the state of the OPE and IPE bits. 1 This port is stopped and is not enabled to issue or receive any packets; the input port can still follow the training procedure and can still send and respond to link-requests; all received packets return packet-not-accepted control symbols to force an error condition to be signaled by the sending device.
31 PT	Port type (read-only). 0 Reserved 1 Serial port.

21.5.24 Port 2 Link maintenance request command and status register (SRIO_P2LMREQCSR)

The port 2 link maintenance request command and status register (P2LMREQCSR), is accessible both by the local processor and an external device. A write to this register generates a link-request control symbol on the RapidIO endpoint port 2 interface. Care should be taken when writing this register that it is only used for hot swap and not for software-assisted error recovery (which is not supported).

Address: C_0000h base + 160h offset = C_0160h



SRIO_P2LMREQCSR field descriptions

Field	Description
0–28 -	This field is reserved. Reserved
29–31 C	LINK_REQUEST command to send. If read, this field returns the last written value. If written with a value other than 0b011 (reset-device) or 0b100 (input-status), the resulting operation is undefined, as all other values are reserved in the RapidIO specification.

21.5.25 Port 2 Link maintenance response command and status register (SRIO_P2LMRESPCSR)

The port 2 link maintenance response command and status register (P2LMRESPCSR), is accessible both by the local processor and an external device. A read to this register returns the status received in a link-response control symbol. This register is read only.

Address: C_0000h base + 164h offset = C_0164h

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

SRIO_P2LMRESPCSR field descriptions

Field	Description
0 RV	Response valid • If the link-request causes a link-response, this bit indicates that the link-response has been received and the status fields are valid. • If the link-request does not cause a link-response, this bit indicates that the link-request has been transmitted (clears on read)
1–21 -	This field is reserved. Reserved
22–26 AS	AckID_status field from LINK_RESPONSE.
27–31 LS	Link_status field from LINK_RESPONSE

21.5.26 Port 2 Local ackID status command and status register (SRIO_P2LASCSR)

The port 2 local ackID status command and status register (P2LASCSR), is accessible both by the local processor and an external device. A read to this register returns the local ackID status for both the output and input ports of the device. Care should be taken to use this register only for hot swap and not software error management.

Address: C_0000h base + 168h offset = C_0168h

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			IA					Reserved											OUTA					Reserved			OBA				
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

SRIO_P2LASCSR field descriptions

Field	Description
0–2 -	This field is reserved. Reserved
3–7 IA	Inbound ackID: Input port next expected ackID value.
8–18 -	This field is reserved. Reserved
19–23 OUTA	Outstanding port unacknowledged ackID status. Next expected acknowledge control symbol ackID field that indicates the ackID value expected in the next received acknowledge control symbol. Note that this value is read only even though RapidIO specification allows for it to be writable.
24–26 -	This field is reserved. Reserved
27–31 OBA	Outbound ackID output port next transmitted ackID value. This can be written by software but only if there are no outstanding unacknowledged packets. If there are, the newly-written value is ignored.

21.5.27 Port 2 Error and status command and status register (SRIO_P2ESCSR)

The port 2 error and status command and status register (P2ESCSR), is accessed when the local processor or an external device wishes to examine the port error and status information.

Address: C_0000h base + 178h offset = C_0178h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved					OPD	OFE	ODE	Reserved			ORE	OR	ORS	OEE	OES
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					IRS	IEE	IES	Reserved			PWP	Reserved	PE	PO	PU
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

SRIO_P2ESCSR field descriptions

Field	Description
0–4 -	This field is reserved. Reserved
5 OPD	Output Packet-dropped. Output port has discarded a packet. A packet is discarded if: • It is received while OFE is set and PnCCSR[DPE] (drop packet enable) is set and PnCCSR[SPF] (stop on port failed) is set. • It is received while PnPCR[OBDEN] (output buffer drain enable) is set. • It is not-accepted by the link-partner while PnERCSR[ERFTT] (error rate failed threshold trigger) is met or exceeded and PnCCSR[DPE] is set and PnCCSR[SPF] is not set (and link-response returns expected ackID). Once OPD is set, it remains set until written with a logic 1 to clear.

Table continues on the next page...

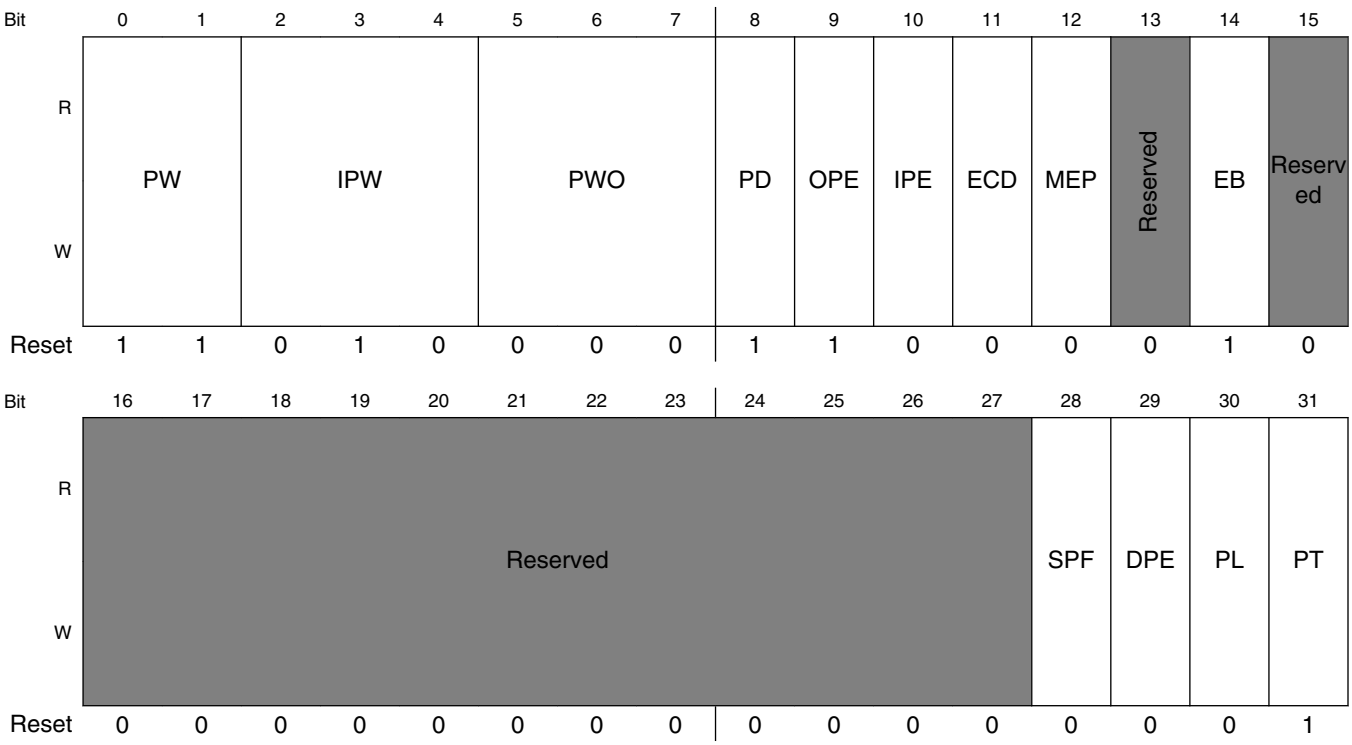
SRIO_P2ESCSR field descriptions (continued)

Field	Description
6 OFE	Output Failed-encountered. Output port has encountered a failed condition, meaning that the Error Rate Counter has met or exceeded the port's failed error threshold (ERFTT) Once set, remains set until written with a logic 1 to clear. Once cleared, does not assert again unless the Error Rate Counter dips below the port's failed error threshold and then meets or exceeds it again.
7 ODE	Output port has encountered a degraded condition, meaning that the Error Rate Counter has met or exceeded the port's degraded error threshold. Once set, remains set until written with a logic 1 to clear. Once cleared, does not assert again unless the Error Rate Counter dips below the port's degraded error threshold and then meets or exceeds it again.
8–10 -	This field is reserved. Reserved
11 ORE	Output port has encountered a retry condition. This bit is set when bit 13 is set. Once set, remains set until written with a logic 1 to clear.
12 OR	Output port has received a packet retry control symbol and cannot make forward progress. This bit is set when bit 13 is set and cleared when a packet-accepted or packet-not-accepted control symbol is received. (read only)
13 ORS	Output port is stopped due to a retry (read only)
14 OEE	Output port has encountered (and possibly recovered from) a transmission error. This bit is set when bit 15 is set. Once set, remains set until written with a logic 1 to clear.
15 OES	Output port is stopped due to a transmission error (read only)
16–20 -	This field is reserved. Reserved
21 IRS	Input port is stopped due to a retry (read only)
22 IEE	Input port has encountered (and possibly recovered from) a transmission error. This bit is set when bit 23 is set. Once set, remains set until written with a logic 1 to clear.
23 IES	Input port is stopped due to a transmission error (read-only)
24–26 -	This field is reserved. Reserved
27 PWP	Port has encountered a condition which required it to initiate a maintenance port-write operation. This bit is only valid if the device is capable of issuing an auto-generated maintenance port-write transaction. The RapidIO endpoint is not capable of issuing auto-generated port-writes. This bit is hardwired to 0.
28 -	This field is reserved. Reserved
29 PE	Input or output port has encountered an error from which hardware was unable to recover. Once set, remains set until written with a logic 1 to clear. This bit indicates that OFE is set while PnCCSR[SPF] is set; in other words, the failed threshold has been reached which has caused the output port to stop transmitting packets.
30 PO	The input and output ports are initialized and the port is exchanging error-free control symbols with the attached device. (read-only).
31 PU	Input and output ports are not initialized. This bit and bit 30 are mutually exclusive (read-only).

21.5.28 Port 2 Control command and status register (SRIO_P2CCSR)

The port 2 control command and status register (P2CCSR), contains control register bits for the RapidIO port.

Address: C_0000h base + 17Ch offset = C_017Ch



SRIO_P2CCSR field descriptions

Field	Description
0–1 PW	Indicates port width modes supported by the port (read-only). The bits of this field indicates the port width modes supported by the port in addition to 1x mode, which is required. 0x 4x mode not supported 1x 4x mode supported x0 2x mode not supported x1 2x mode supported
2–4 IPW	Width of the ports after initialized (read-only). 000 Single-lane port, lane 0 001 Single-lane port, lane 2 010 Four-lane port, lanes 0-3 011 Two-lane port, lanes 0-1 100-111 Reserved

Table continues on the next page...

SRIO_P2CCSR field descriptions (continued)

Field	Description
5–7 PWO	Soft port configuration to control the width modes available for port initialization. The meaning of bits 6 and 7 depend on the value of bit 5. When bit [5] = 0b1, bit 6 controls the enabling of 4x mode, bit 7 controls the enabling of 2x mode, A change in the value of this field shall cause the port to re-initialize using the new field value. This field should be changed only when the port is uninitialized. To achieve this, first disable the RapidIO port. Then change PWO to any valid value. Finally, re-enable the RapidIO port. To achieve this, first set PnCCSR[PD] (port disabled). Then change PWO to any legal value. Finally, clear PnCCSR[PD] (enabled). 000 No override, all supported mode widths enabled. 001 Reserved 010 Force single lane, lane R not forced 011 Force single lane, force lane R 100 Reserved 101 2x mode enabled, 4x mode disabled 110 2x mode disabled, 4x mode enabled 111 2x mode enabled, 4x mode enabled
8 PD	Port disable. 0b0 - port receivers/drivers are enabled 0b1 - port receivers/drivers are disabled and are unable to receive/transmit If set, the output will congest if packets are sent to it.
9 OPE	Output port transmit enable. Initial value read from configuration pins. 0 Port is stopped and not enabled to issue any packets except to route or respond to I/O logical MAINTENANCE packets. Control symbols are not affected and are sent normally. 1 Port is enabled to issue any packets.
10 IPE	Input port receive enable. 0 Port is stopped and only enabled to route or respond to I/O logical MAINTENANCE packets. Other packets generate packet-not-accepted control symbols to force an error condition to be signaled by the sending device. Control symbols are not affected and are received and handled normally. 1 Port is enabled to respond to any packet.
11 ECD	Error checking disable. This bit is hardwired to 0. This bit disables all RapidIO transmission error checking This bit value must equal the value of the output port enable (OPE) bit in order for the RapidIO controller to function properly. Initial value read from configuration pins. 0 Error checking and recovery is enabled. 1 Error checking and recovery is disabled.

Table continues on the next page...

SRIO_P2CCSR field descriptions (continued)

Field	Description
12 MEP	Multicast-event participant. This bit is hard-wired to 0.
13 -	This field is reserved. Reserved
14 EB	Enumeration boundary. An enumeration boundary aware system enumeration algorithm shall honor this flag. The algorithm, on either the ingress or the egress port, shall not enumerate past a port with this bit set.
15–27 -	This field is reserved. Reserved
28 SPF	Stop on port failed-encountered enable. This bit is used with the Drop Packet Enable bit to force certain behavior when the Error Rate Failed Threshold has been met or exceeded.
29 DPE	Drop packet enable. This bit is used with the Stop on Port Failed-encountered Enable bit to force certain behavior when the Error Rate Failed Threshold has been met or exceeded.
30 PL	Port lockout. 0 The packets that may be received and issued are controlled by the state of the OPE and IPE bits. 1 This port is stopped and is not enabled to issue or receive any packets; the input port can still follow the training procedure and can still send and respond to link-requests; all received packets return packet-not-accepted control symbols to force an error condition to be signaled by the sending device.
31 PT	Port type (read-only). 0 Reserved 1 Serial port.

21.5.29 Error reporting block header (SRIO_ERBH)

The error reporting block header register contains the EF_PTR to the next EF_BLK (the next EF_PTR is 0x0000 since this is the last set of registers in the extended features space) and the EF_ID that identifies this as the error reporting block header. ERBH is a read-only register.

Address: C_0000h base + 600h offset = C_0600h

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	EF_PTR																EF_ID															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

SRIO_ERBH field descriptions

Field	Description
0–15 EF_PTR	Extended features pointer
16–31 EF_ID	Extended features ID

21.5.30 Logical/Transport layer error detect command and status register (SRIO_LTLEDCSR)

This register indicates the error that was detected by the Logical or Transport logic layer. Software should write this register with all 0s to clear the detected error and unlock the capture registers.

Error information that corresponds to two or more different error events are not captured on the same clock cycle. However, one error event (such as a corrupted inbound packet) can cause multiple bits to be set. The priority of errors is PRT and all other errors. OACB error results in PRT. When PRT bit is set with OACB bit, error capture is for an internal platform transaction for which an Outbound ATMU window boundary was crossed or a segment or subsegment boundary was crossed.

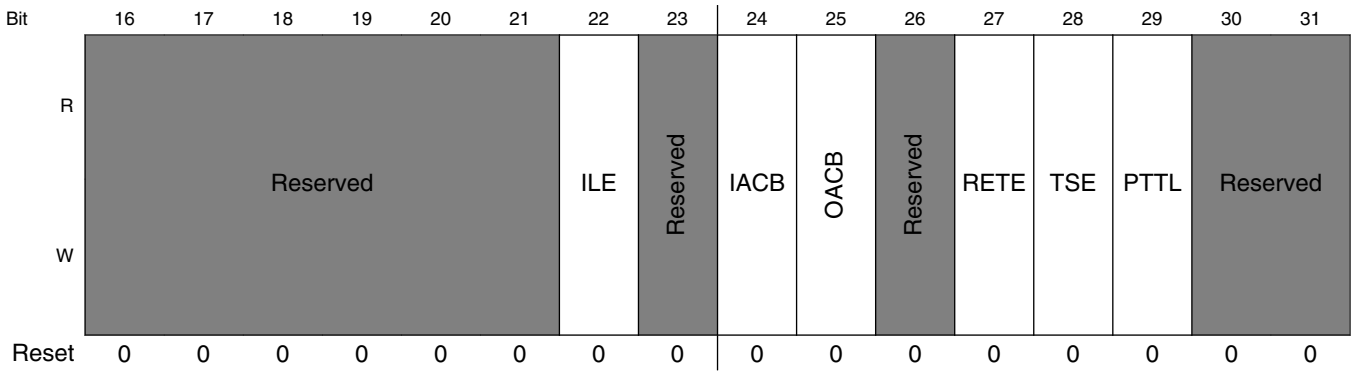
An error that is not enabled will set the detect bit in this register as long as a capture has not yet occurred.

This register is shared among all ports. Error detect bits for each type of error are OR'd together with the upper port taking precedence.

Note that fields in this register can be set by writing to this register. This can be used to emulate a hardware error during software development. However, undefined results occur if fields in this register are set while an actual Logical/Transport Layer error is being detected. Also, note that this register can be written with an invalid combination of bits set and care should be taken to avoid this.

Address: C_0000h base + 608h offset = C_0608h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	IER	Reserved	GER	Reserved	ITD	ITTE	Reserved	PRT	UR	UT	Reserved					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



SRIO_LTLEDCSR field descriptions

Field	Description
0 IER	IO error response. Received a response of ERROR for an IO logical layer request.
1 -	This field is reserved. Reserved
2 GER	GSM error response. Received a response of ERROR for a GSM logical layer request.
3 -	This field is reserved. Reserved
4 ITD	Illegal transaction decode. Received illegal fields in the request/response packet for a supported transaction (IO/GSM logical)
5 ITTE	Illegal transaction target error. Received a packet that contained a destination ID that is not defined for this end point when pass-through and accept-all are not enabled. Endpoints with multiple ports and a built-in switch function may not report this as an error (transport)
6 -	This field is reserved. Reserved
7 PRT	Packet response time-out. A required response has not been received within the specified time out interval (IO/GSM logical)
8 UR	Unsolicited response. An unsolicited/unexpected response packet was received (IO/GSM logical; only maintenance response for switches)
9 UT	Unsupported transaction. A transaction is received that is not supported in the destination operations CAR (IO/GSM logical; only maintenance port-write for switches)
10–21 -	This field is reserved. Reserved
22 ILE	Inbound LIODN error. An error occurred during the inbound RapidIO source ID to LIODN translation of a received request
23 -	This field is reserved. Reserved

Table continues on the next page...

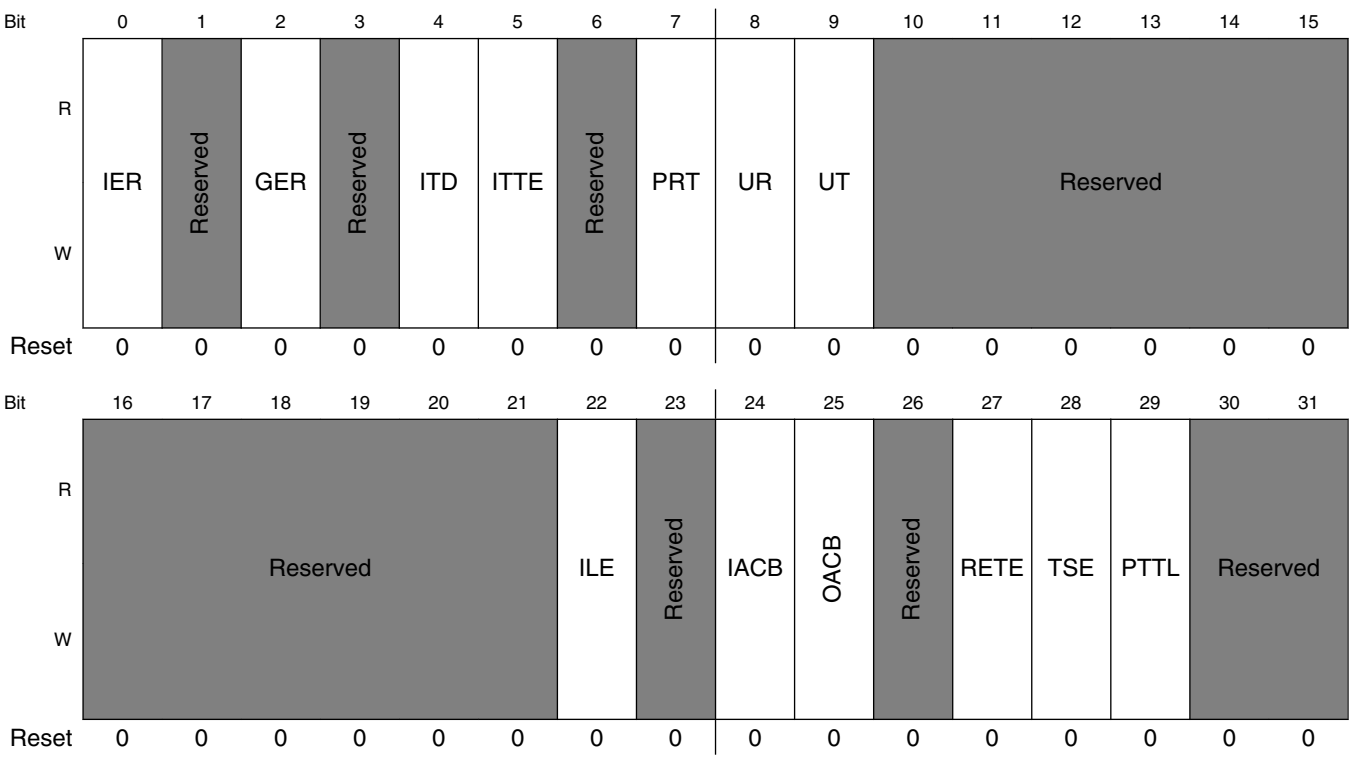
SRIO_LTLEDCSR field descriptions (continued)

Field	Description
24 IACB	Inbound ATMU crossed boundary. A transaction is received that crosses an inbound ATMU boundary.
25 OACB	Outbound ATMU crossed boundary. A transaction is being sent that crosses an outbound ATMU boundary, a segment boundary, or a subsegment boundary.
26 -	This field is reserved. Reserved
27 RETE	Retry error threshold exceeded. The allowed number of logical retries (given by LRETCR[RET] has been exceeded.
28 TSE	Transport size error. The tt field is not consistent with bit 27 of the processing element features CAR (that is, the tt value is reserved or indicates a common transport system that is unsupported by this device).
29 PTTL	Packet time-to-live error. A packet time-to-live error occurred (a packet could not be successfully transmitted before the packet time-to-live counter expired).
30–31 -	This field is reserved. Reserved

21.5.31 Logical/Transport layer error enable command and status register (SRIO_LTLEECSR)

This register contains the bits that control whether an error condition locks the logical/ transport layer error detect and capture registers and is reported to the system host.

Address: C_0000h base + 60Ch offset = C_060Ch



SRIO_LTLEECSR field descriptions

Field	Description
0 IER	IO error response enable.Enable reporting of an IO error response. Capture and lock the error.
1 -	This field is reserved. Reserved
2 GER	GSM error response enable. Enable reporting of a GSM error response. Capture and lock the error.
3 -	This field is reserved. Reserved
4 ITD	Illegal transaction decode enable. Enable reporting of an illegal transaction decode error. Capture and lock the error.
5 ITTE	Illegal transaction target error enable.

Table continues on the next page...

SRIO_LTLEECSR field descriptions (continued)

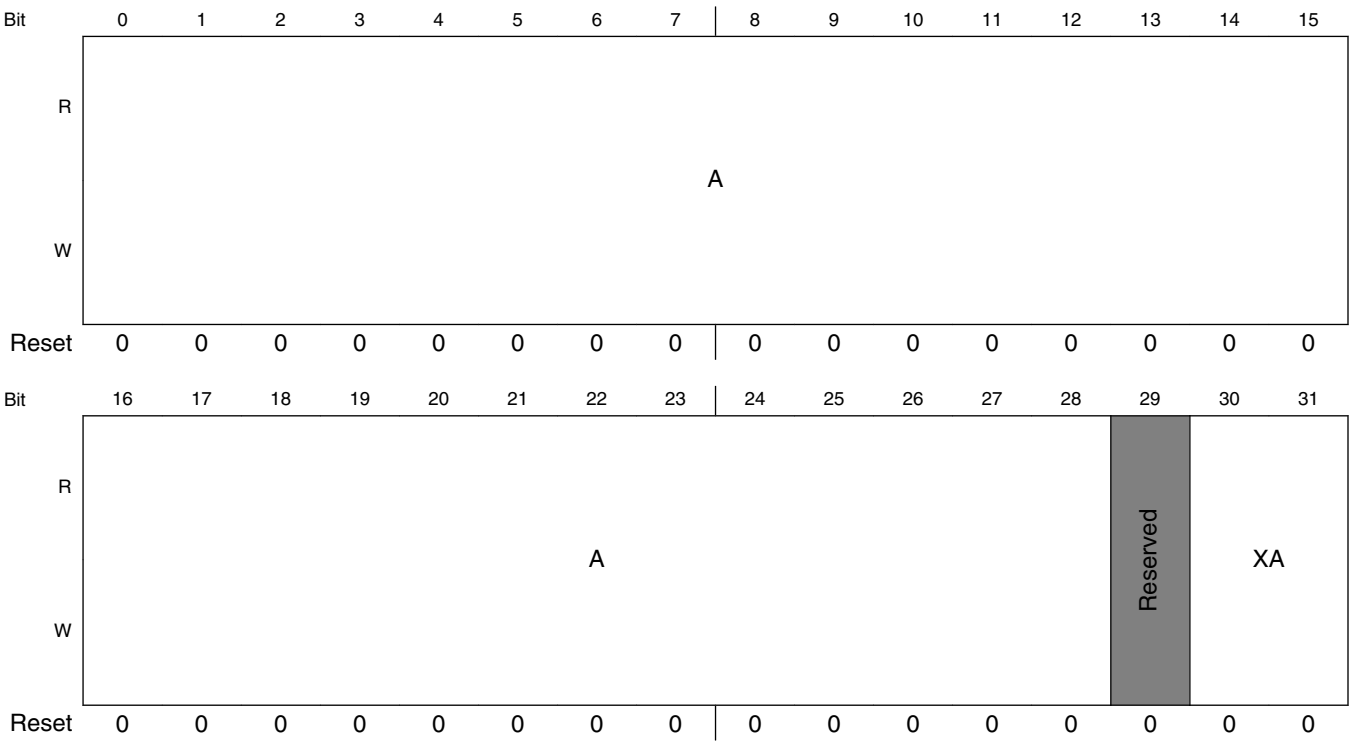
Field	Description
	Enable reporting of an illegal transaction target error. Capture and lock the error.
6 -	This field is reserved. Reserved
7 PRT	Packet response time-out error enable. Enable reporting of a packet response time-out error. Capture and lock the error.
8 UR	Unsolicited response error enable. Enable reporting of an unsolicited response error. Capture and lock the error.
9 UT	Unsupported transaction error enable. Enable reporting of an unsupported transaction error. Capture and lock the error.
10–21 -	This field is reserved. Reserved
22 ILE	Inbound LIODN error. An error occurred during the inbound RapidIO source ID to LIODN translation of a received request.
23 -	This field is reserved. Reserved
24 IACB	Inbound ATMU crossed boundary error enable. Enable reporting of a received transaction that crosses an inbound ATMU boundary. Capture and lock the error.
25 OACB	Outbound ATMU crossed boundary error enable. Enable reporting of a transaction that crosses an outbound ATMU boundary, a segment boundary, or a subsegment boundary. Capture and lock the error.
26 -	This field is reserved. Reserved
27 RETE	Retry error threshold exceeded. Enable error reporting when the allowed number of logical retries has been exceeded.
28 TSE	Transport size error. Enable error reporting when the tt field is not consistent with bit 27 of the Processing Element Features CAR (that is, the tt value is reserved or indicates a common transport system that is unsupported by this device).
29 PTTL	Packet time-to-live error. Enable reporting of a packet time-to-live time-out error. Capture and lock the error.
30–31 -	This field is reserved. Reserved

21.5.32 Logical/Transport layer address capture command and status register (SRIO_LTLACCSR)

The logical/transport layer address capture command and status register (LTLACCSR), provides error information. It is locked when a logical/transport error is detected, and the corresponding enable bit is set. Undefined results occur if this register is written while actual logical/transport layer errors are being detected by the port.

Note that fields in this register can be set by to writing this register. This can be used to emulate a hardware error during software development. However, undefined results occur if fields in this register are set while an actual logical/transport layer error is being detected.

Address: C_0000h base + 614h offset = C_0614h



SRIO_LTLACCSR field descriptions

Field	Description
0–28 A	Normally the least significant 29 bits of the address associated with the error (for requests, for responses if available). See Logical Layer Errors and Error Handling," for details.
29 -	This field is reserved. Reserved
30–31 XA	xamsbs. Normally the extended address bits of the address associated with the error (for requests, responses, if available). See Logical Layer Errors and Error Handling," for details.

21.5.33 Logical/Transport layer device ID capture command and status register (SRIO_LTLDIDCCSR)

The logical/transport layer device ID capture command and status register (LTLDIDCCSR), contains error information. It is locked when a logical/transport error is detected, and the corresponding enable bit is set. Undefined results occur if this register is written while actual logical/transport layer errors are being detected by the port.

Note that fields in this register can be set by writing to this register, in order to emulate a hardware error during software development. However, undefined results occur if fields in this register are set while an actual logical/transport layer error is being detected.

Address: C_0000h base + 618h offset = C_0618h

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	DIDMSB								DID								SIDMSB								SID							
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

SRIO_LTLDIDCCSR field descriptions

Field	Description
0–7 DIDMSB	Normally the most significant byte of the destinationID associated with the error. This field is valid only if bit 27 of the Processing Element Features CAR is set (large transport systems only). See Logical Layer Errors and Error Handling," for details.
8–15 DID	Normally the destinationID (or least significant byte of the destination ID if large transport system) associated with the error. See Logical Layer Errors and Error Handling," for details.
16–23 SIDMSB	Normally the most significant byte of the sourceID associated with the error. This field is valid only if bit 27 of the Processing Element Features CAR is set (large transport systems only). See Logical Layer Errors and Error Handling," for details.
24–31 SID	Normally the sourceID (or least significant byte of the source ID if large transport system) associated with the error. See Logical Layer Errors and Error Handling," for details.

21.5.34 Logical/Transport layer control capture command and status register (SRIO_LTLCCCSR)

The logical/transport layer control capture command and status register (LTLCCCSR), contains error information. Undefined results occur if this register is written while actual logical/transport layer errors are being detected by the port.

Note that fields in this register can be set by writing to this register, in order to emulate a hardware error during software development. However, undefined results occur if fields in this register are set while an actual logical/transport layer error is being detected.

Address: C_0000h base + 61Ch offset = C_061Ch

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	FT				TT				MI								EPN				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

SRIO_LTLCCCSR field descriptions

Field	Description
0–3 FT	Normally the format type associated with the error. See Logical Layer Errors and Error Handling for details.

Table continues on the next page...

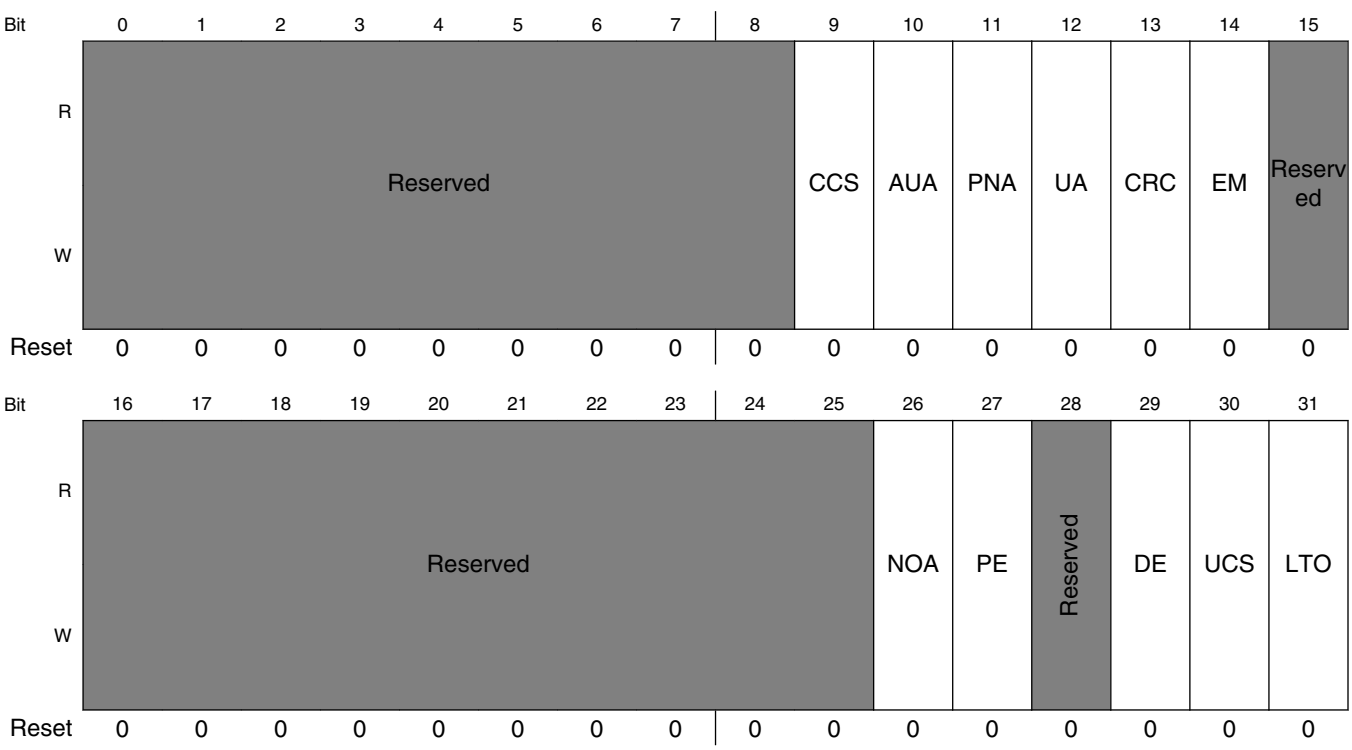
SRIO_LTLCCCSR field descriptions (continued)

Field	Description
4–7 TT	Normally the transaction type associated with the error. See Logical Layer Errors and Error Handling for details.
8–15 MI	Normally the message Information: letter, mbox, and msgseg for the last message request received for the mailbox that had an error (message errors only). See Logical Layer Errors and Error Handling for details.
16–19 EPN	Encoded port number: indicates the port that the error information was captured from.
20–31 -	This field is reserved. Reserved

21.5.35 Port 1 Error detect command and status register (SRIO_P1EDCSR)

The port 1 error detect command and status register (P1EDCSR), indicates transmission errors that are detected by the hardware. Software can write bits in this register with 1 to cause the Error Rate Counter to increment. Undefined results occur if this register is written while actual physical layer errors are being detected by the port.

Address: C_0000h base + 640h offset = C_0640h



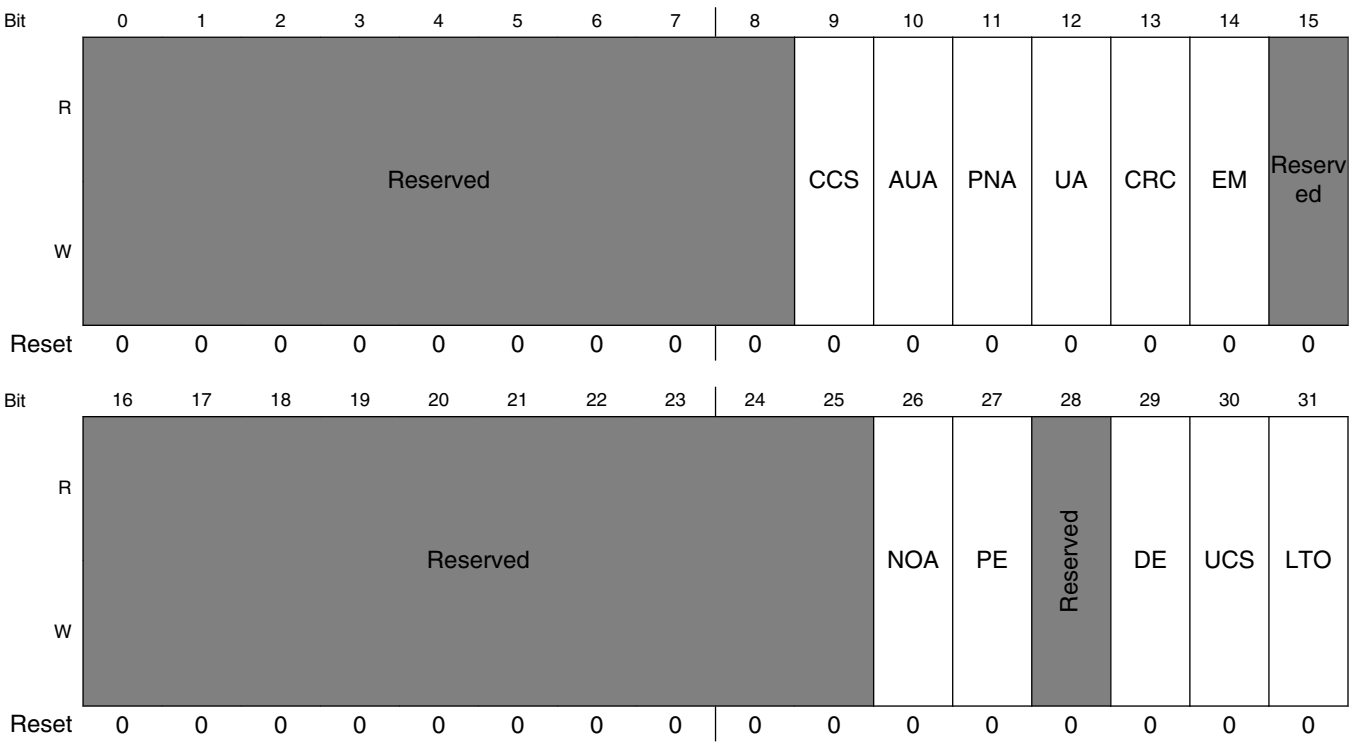
SRIO_P1EDCSR field descriptions

Field	Description
0–8 -	This field is reserved. Reserved
9 CCS	Received a control symbol with a bad CRC value.
10 AUA	Received acknowledge control symbol with unexpected ackID (packet-accepted or packet-retry).
11 PNA	Received packet-not-accepted acknowledge control symbol
12 UA	Received packet with unexpected ackID value.
13 CRC	Received a packet with a bad CRC value
14 EM	Received packet which exceed the maximum allowed size (276 bytes).
15–25 -	This field is reserved. Reserved
26 NOA	Link-response received with an ackID that is not outstanding.
27 PE	Protocol Error: An unexpected packet or control symbol was received.
28 -	This field is reserved. Reserved
29 DE	Received unaligned /SC/ or /PD/ or undefined code-group.
30 UCS	An unexpected acknowledge control symbol was received.
31 LTO	An acknowledge or link-response control symbol is not received within the specified time-out interval.

21.5.36 Port 1 Error rate enable command and status register (SRIO_P1ERECSR)

The port 1 error rate enable command and status register (P1ERECSR), contains the bits that control when an error condition is allowed to increment the error rate counter in the port *n*error rate threshold register and lock the port 1 error capture registers.

Address: C_0000h base + 644h offset = C_0644h



SRIO_P1ERECSR field descriptions

Field	Description
0–8 -	This field is reserved. Reserved
9 CCS	Enable error rate counting of a corrupt control symbol
10 AUA	Enable error rate counting of an acknowledge control symbol with an unexpected ackID
11 PNA	Enable error rate counting of received packet-not-accepted control symbols.
12 UA	Enable error rate counting of packet with unexpected ackID value.
13 CRC	Enable error rate counting of packet with a bad CRC value.

Table continues on the next page...

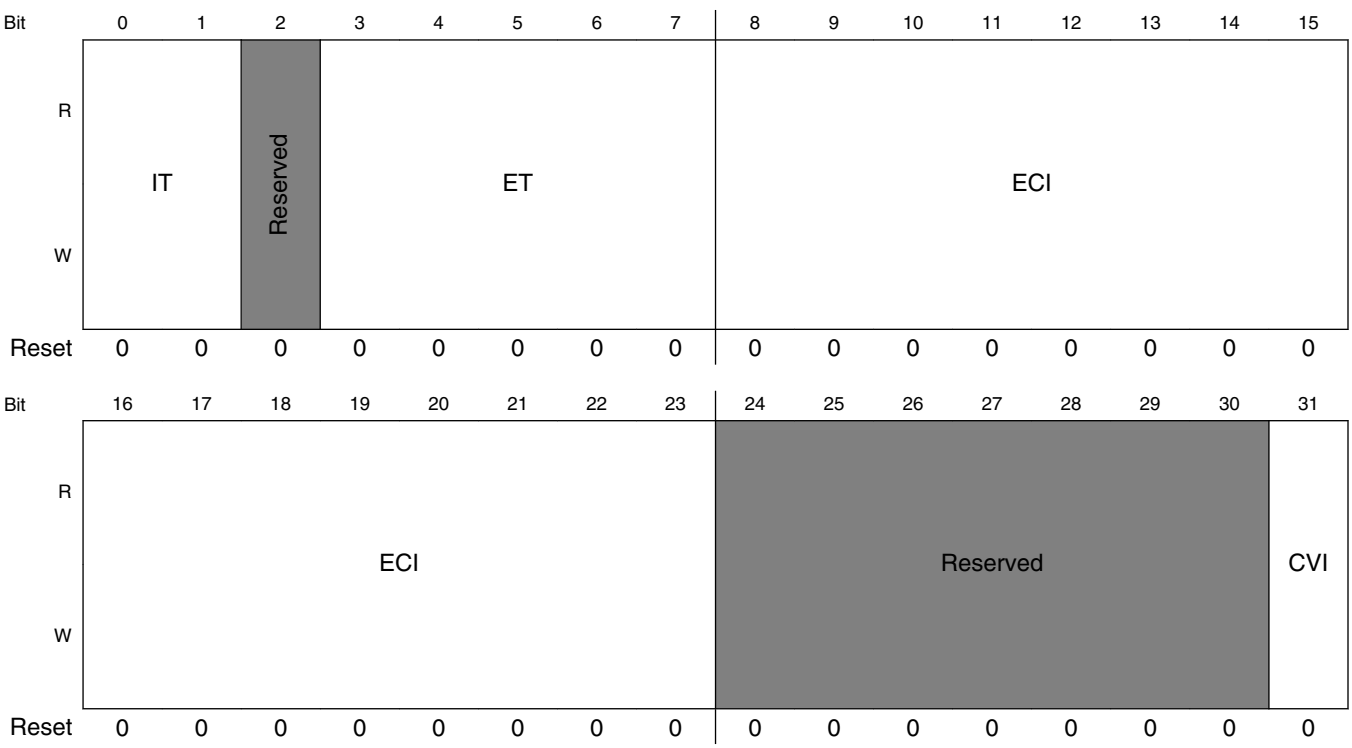
SRIO_P1ERECSR field descriptions (continued)

Field	Description
14 EM	Enable error rate counting of packet which exceeds the maximum allowed size
15–25 -	This field is reserved. Reserved
26 NOA	Enable error rate counting of link-responses received with an ackID that is not outstanding.
27 PE	Enable error rate counting of protocol errors
28 -	This field is reserved. Reserved
29 DE	Enable error rate counting of delineation errors.
30 UCS	Enable error rate counting of unsolicited acknowledge control symbol errors.
31 LTO	Enable error rate counting of link time-out errors.

21.5.37 Port 1 Error capture attributes command and status register (SRIO_P1ECACSR)

The port 1 error capture attribute command and status register (P1ECACSR), indicates the type of information contained in the port 1 error capture registers. In the case of multiple detected errors during the same clock cycle one of the errors must be reflected in the Error type field. Undefined results occur if this register is written while actual physical layer errors are being detected by the port. Software should check that the P1ECACSR[CVI] bit is set before reading the capture registers to ensure that the error has been properly captured.

Address: C_0000h base + 648h offset = C_0648h



SRIO_P1ECACSR field descriptions

Field	Description
0–1 IT	Type of information logged: 00 Packet (error capture registers hold the first 4 words of the packet, or the entire packet if it is less than 4 words long). 01 Control symbol (only error capture register 0 is valid)

Table continues on the next page...

SRIO_P1ECACSR field descriptions (continued)

Field	Description
	10 Reserved 11 Undefined (not clearly a control symbol or packet error. Error capture registers hold the symbol that caused the error and the next 3 symbols.)
2 -	This field is reserved. Reserved
3–7 ET	The encoded value of the bit in the port 1 error detect CSR that describes the error captured in the port 1 error capture CSRs. 0x00 No Error 0x09 CCS Error Type 0x0A AUA Error Type 0x0B PNA Error Type 0x0C UA Error Type 0x0D CRC Error Type 0x0E EM Error Type 0x1A NOA Error Type 0x1B PE Error Type 0x1D DE Error Type 0x1E UCS Error Type Other values are reserved and cause undefined operation.
8–23 ECI	Extended capture information [0:15]. ECI contains the control/data character signal corresponding to each byte of captured data. ECI[08:11] == control/data character for ECCSR0 ECI[12:15] == control/data character for ECCSR1 ECI[16:19] == control/data character for ECCSR2 ECI[20:23] == control/data character for ECCSR3 Where control/data characters are defined as the following: 0x0 == Data symbol. 0x8 == Control character. 0xF == Idle symbol. All other values reserved.
24–30 -	This field is reserved. Reserved
31 CVI	This bit is set by hardware to indicate that the Packet/control symbol capture registers contain valid information. For control symbols, only capture register 0 contains meaningful information.

21.5.38 Port 1 Packet/control symbol error capture command and status register 0 (SRIO_P1PCSECCSR0)

The port 1 package/control symbol error capture command and status register 0 (P1PCSECCSR0), contains the first four bytes of captured packet symbol information or a control character and control symbol. Undefined results occur if this register is written while actual physical layer errors are being detected by the port. Software should check that the P1ECACSR[CVI] bit is set before reading the capture registers to ensure that the error has been properly captured.

Address: C_0000h base + 64Ch offset = C_064Ch

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	C0																															
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

SRIO_P1PCSECCSR0 field descriptions

Field	Description
0–31 C0	Capture 0: Control Character and control symbol or bytes 0 to 3 of packet header.

21.5.39 Port 1 Packet error capture command and status register 1 (SRIO_P1PECCSR1)

The port 1 package error capture command and status register 1 (P1PECCSR1), contains bytes 4 through 7 of the packet header. Undefined results occur if this register is written while actual physical layer errors are being detected by the port. Software should check that the P1ECACSR[CVI] bit is set before reading the capture registers to ensure that the error has been properly captured.

Address: C_0000h base + 650h offset = C_0650h

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	C1																															
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

SRIO_P1PECCSR1 field descriptions

Field	Description
0–31 C1	Capture 1. Bytes 4 to 7 of the packet header

21.5.40 Port 1 Packet error capture command and status register 2 (SRIO_P1PECCSR2)

The port 1 package error capture command and status register 2 (P1PECCSR2), contains bytes 8 through 11 of the packet header. Undefined results occur if this register is written while actual physical layer errors are being detected by the port. Software should check that the P1ECACSR[CVI] bit is set before reading the capture registers to ensure that the error has been properly captured.

Address: C_0000h base + 654h offset = C_0654h

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	C2																															
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

SRIO_P1PECCSR2 field descriptions

Field	Description
0–31 C2	Capture 2. Bytes 8 to 11 of the packet header

21.5.41 Port 1 Packet error capture command and status register 3 (SRIO_P1PECCSR3)

The port 1 package error capture command and status register 3 (P1PECCSR3), contains bytes 12 through 15 of the packet header. Undefined results occur if this register is written while actual physical layer errors are being detected by the port. Software should check that the P1ECACSR[CVI] bit is set before reading the capture registers to ensure that the error has been properly captured.

Address: C_0000h base + 658h offset = C_0658h

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	C3																															
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

SRIO_P1PECCSR3 field descriptions

Field	Description
0–31 C3	Capture 3. Bytes 12 to 15 of the packet header

21.5.42 Port 1 Error rate command and status register (SRIO_P1ERCSR)

The port 1 error rate command and status register (P1ERCSR), is a 32-bit register used with the port 1 error rate threshold register to monitor and control the reporting of transmission errors.

Address: C_0000h base + 668h offset = C_0668h

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	ERB								Reserved						ERR		PER								ERC							
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRIO_P1ERCSR field descriptions

Field	Description
0–7 ERB	These bits provide the error rate bias value. 0x00 Do not decrement the error rate counter 0x01 Decrement every 1 ms (±34%) 0x02 Decrement every 10 ms (±34%) 0x04 Decrement every 100 ms (±34%) 0x08 Decrement every 1 s (±34%) 0x10 Decrement every 10 s (±34%) 0x20 Decrement every 100 s (±34%) 0x40 Decrement every 1000 s (±34%) 0x80 Decrement every 10000 s (±34%) Other values are reserved and cause undefined operation.
8–13 -	This field is reserved. Reserved
14–15 ERR	These bits limit the incrementing of the error rate counter above the failed threshold trigger. 00 Only count 2 errors above 01 Only count 4 errors above 10 Only count 16 error above 11 Do not limit incrementing the error rate count Note that the Error Rate Counter never increments above 0xFF, even if the combination of the settings of ERR and the failed threshold trigger might imply that it might.

Table continues on the next page...

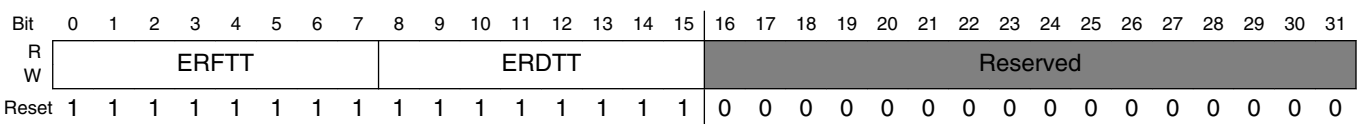
SRIO_P1ERCSR field descriptions (continued)

Field	Description
16–23 PER	Peak error rate. Contains the peak value attained by the error rate counter
24–31 ERC	Error rate counter. These bits maintain a count of the number of transmission errors that have been detected by the port, decremented by the Error Rate Bias mechanism, to create an indication of the link error rate. Software should not attempt to write this field to a value higher than failed threshold trigger plus the number of errors specified in the ERR field (the maximum ERC value).

21.5.43 Port 1 Error rate threshold command and status register (SRIO_P1ERTCSR)

The port 1 error rate threshold command and status register (P1ERTCSR), is a 32-bit register used to control the reporting of the link status to the system host.

Address: C_0000h base + 66Ch offset = C_066Ch



SRIO_P1ERTCSR field descriptions

Field	Description
0–7 ERFTT	Error rate failed threshold trigger. These bits provide the threshold value for reporting an error condition due to a possibly broken link. The PnESCSR[OFE] bit is not set if ERFTT is written to a value lower than or equal to the PnERCSR[ERC]. Please see Table 21-267 for more details. 0x00 Disable the Error Rate Failed Threshold Trigger. 0x01 Set the error reporting threshold to 1. 0x02 Set the error reporting threshold to 2. 0xFF Set the error reporting threshold to 255.
8–15 ERDTT	Error rate degraded threshold trigger. These bits provide the threshold value for reporting an error condition due to a degrading link. The PnESCSR[ODE] bit is not set if ERDTT is written to a value lower than or equal to the PnERCSR[ERC]. Please see Table 21-267 for more details. 0x00 Disable the Error Rate Degraded Threshold Trigger. 0x01 Set the error reporting threshold to 1.

Table continues on the next page...

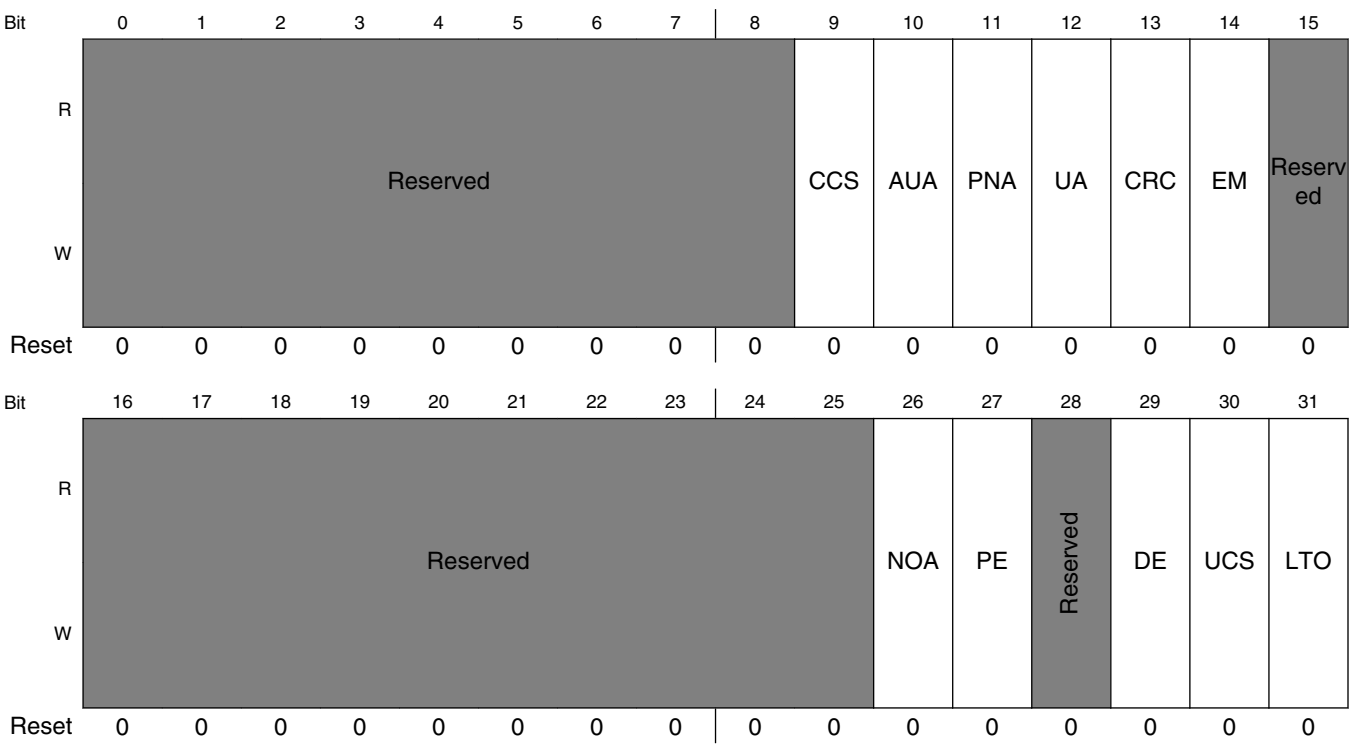
SRIO_P1ERTCSR field descriptions (continued)

Field	Description
	0x02 Set the error reporting threshold to 2. 0xFF Set the error reporting threshold to 255.
16–31 -	This field is reserved. Reserved

21.5.44 Port 2 Error detect command and status register (SRIO_P2EDCSR)

The port 2 error detect command and status register (P2EDCSR), indicates transmission errors that are detected by the hardware. Software can write bits in this register with 1 to cause the Error Rate Counter to increment. Undefined results occur if this register is written while actual physical layer errors are being detected by the port.

Address: C_0000h base + 680h offset = C_0680h



SRIO_P2EDCSR field descriptions

Field	Description
0–8 -	This field is reserved. Reserved

Table continues on the next page...

SRIO_P2EDCSR field descriptions (continued)

Field	Description
9 CCS	Received a control symbol with a bad CRC value.
10 AUA	Received acknowledge control symbol with unexpected ackID (packet-accepted or packet-retry).
11 PNA	Received packet-not-accepted acknowledge control symbol
12 UA	Received packet with unexpected ackID value.
13 CRC	Received a packet with a bad CRC value
14 EM	Received packet which exceed the maximum allowed size (276 bytes).
15–25 -	This field is reserved. Reserved
26 NOA	Link-response received with an ackID that is not outstanding.
27 PE	Protocol Error: An unexpected packet or control symbol was received.
28 -	This field is reserved. Reserved
29 DE	Received unaligned /SC/ or /PD/ or undefined code-group.
30 UCS	An unexpected acknowledge control symbol was received.
31 LTO	An acknowledge or link-response control symbol is not received within the specified time-out interval.

21.5.45 Port 2 Error rate enable command and status register (SRIO_P2ERECSR)

The port 2 error rate enable command and status register (P2ERECSR), contains the bits that control when an error condition is allowed to increment the error rate counter in the port *n*error rate threshold register and lock the port 2 error capture registers.

Address: C_0000h base + 684h offset = C_0684h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved									CCS	AUA	PNA	UA	CRC	EM	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										NOA	PE	Reserved	DE	UCS	LTO
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRIO_P2ERECSR field descriptions

Field	Description
0–8 -	This field is reserved. Reserved
9 CCS	Enable error rate counting of a corrupt control symbol
10 AUA	Enable error rate counting of an acknowledge control symbol with an unexpected ackID
11 PNA	Enable error rate counting of received packet-not-accepted control symbols.
12 UA	Enable error rate counting of packet with unexpected ackID value.
13 CRC	Enable error rate counting of packet with a bad CRC value.

Table continues on the next page...

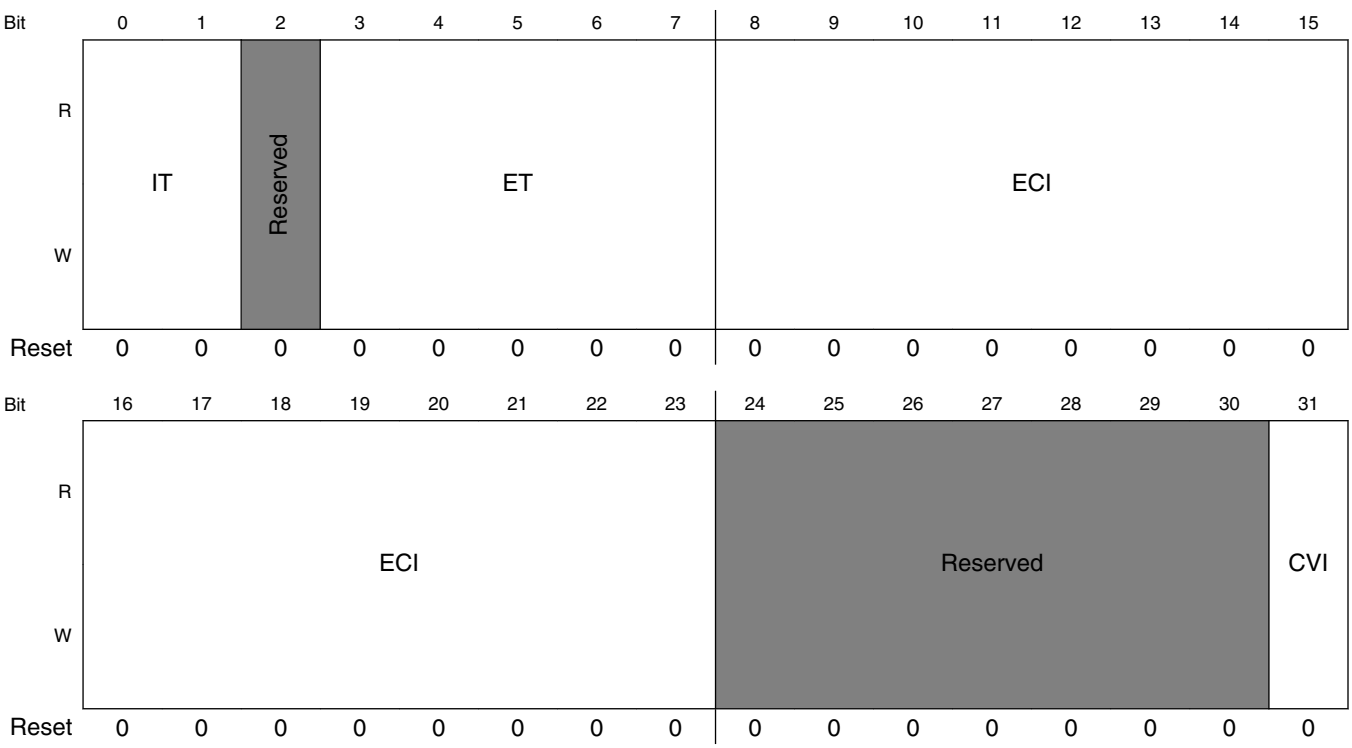
SRIO_P2ERECSR field descriptions (continued)

Field	Description
14 EM	Enable error rate counting of packet which exceeds the maximum allowed size
15–25 -	This field is reserved. Reserved
26 NOA	Enable error rate counting of link-responses received with an ackID that is not outstanding.
27 PE	Enable error rate counting of protocol errors
28 -	This field is reserved. Reserved
29 DE	Enable error rate counting of delineation errors.
30 UCS	Enable error rate counting of unsolicited acknowledge control symbol errors.
31 LTO	Enable error rate counting of link time-out errors.

21.5.46 Port 2 Error capture attributes command and status register (SRIO_P2ECACSR)

The port 2 error capture attribute command and status register (P2ECACSR), indicates the type of information contained in the port 2 error capture registers. In the case of multiple detected errors during the same clock cycle one of the errors must be reflected in the Error type field. Undefined results occur if this register is written while actual physical layer errors are being detected by the port. Software should check that the P2ECACSR[CVI] bit is set before reading the capture registers to ensure that the error has been properly captured.

Address: C_0000h base + 688h offset = C_0688h



SRIO_P2ECACSR field descriptions

Field	Description
0–1 IT	Type of information logged: 00 Packet (error capture registers hold the first 4 words of the packet, or the entire packet if it is less than 4 words long). 01 Control symbol (only error capture register 0 is valid)

Table continues on the next page...

SRIO_P2ECACSR field descriptions (continued)

Field	Description
	10 Reserved 11 Undefined (not clearly a control symbol or packet error. Error capture registers hold the symbol that caused the error and the next 3 symbols.)
2 -	This field is reserved. Reserved
3–7 ET	The encoded value of the bit in the port 2 error detect CSR that describes the error captured in the port 2 error capture CSRs. 0x00 No Error 0x09 CCS Error Type 0x0A AUA Error Type 0x0B PNA Error Type 0x0C UA Error Type 0x0D CRC Error Type 0x0E EM Error Type 0x1A NOA Error Type 0x1B PE Error Type 0x1D DE Error Type 0x1E UCS Error Type Other values are reserved and cause undefined operation.
8–23 ECI	Extended capture information [0:15]. ECI contains the control/data character signal corresponding to each byte of captured data. ECI[08:11] == control/data character for ECCSR0 ECI[12:15] == control/data character for ECCSR1 ECI[16:19] == control/data character for ECCSR2 ECI[20:23] == control/data character for ECCSR3 Where control/data characters are defined as the following: 0x0 == Data symbol. 0x8 == Control character. 0xF == Idle symbol. All other values reserved.
24–30 -	This field is reserved. Reserved
31 CVI	This bit is set by hardware to indicate that the Packet/control symbol capture registers contain valid information. For control symbols, only capture register 0 contains meaningful information.

21.5.47 Port 2 Packet/control symbol error capture command and status register 0 (SRIO_P2PCSECCSR0)

The port 2 package/control symbol error capture command and status register 0 (P2PCSECCSR0), contains the first four bytes of captured packet symbol information or a control character and control symbol. Undefined results occur if this register is written while actual physical layer errors are being detected by the port. Software should check that the P2ECACSR[CVI] bit is set before reading the capture registers to ensure that the error has been properly captured.

Address: C_0000h base + 68Ch offset = C_068Ch

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	C0																															
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

SRIO_P2PCSECCSR0 field descriptions

Field	Description
0–31 C0	Capture 0: Control Character and control symbol or bytes 0 to 3 of packet header.

21.5.48 Port 2 Packet error capture command and status register 1 (SRIO_P2PECCSR1)

The port 2 package error capture command and status register 1 (P2PECCSR1), contains bytes 4 through 7 of the packet header. Undefined results occur if this register is written while actual physical layer errors are being detected by the port. Software should check that the P2ECACSR[CVI] bit is set before reading the capture registers to ensure that the error has been properly captured.

Address: C_0000h base + 690h offset = C_0690h

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	C1																															
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

SRIO_P2PECCSR1 field descriptions

Field	Description
0–31 C1	Capture 1. Bytes 4 to 7 of the packet header

21.5.49 Port 2 Packet error capture command and status register 2 (SRIO_P2PECCSR2)

The port 2 package error capture command and status register 2 (P2PECCSR2), contains bytes 8 through 11 of the packet header. Undefined results occur if this register is written while actual physical layer errors are being detected by the port. Software should check that the P2ECACSR[CVI] bit is set before reading the capture registers to ensure that the error has been properly captured.

Address: C_0000h base + 694h offset = C_0694h

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	C2																															
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

SRIO_P2PECCSR2 field descriptions

Field	Description
0–31 C2	Capture 2. Bytes 8 to 11 of the packet header

21.5.50 Port 2 Packet error capture command and status register 3 (SRIO_P2PECCSR3)

The port 2 package error capture command and status register 3 (P2PECCSR3), contains bytes 12 through 15 of the packet header. Undefined results occur if this register is written while actual physical layer errors are being detected by the port. Software should check that the P2ECACSR[CVI] bit is set before reading the capture registers to ensure that the error has been properly captured.

Address: C_0000h base + 698h offset = C_0698h

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	C3																															
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

SRIO_P2PECCSR3 field descriptions

Field	Description
0–31 C3	Capture 3. Bytes 12 to 15 of the packet header

21.5.51 Port 2 Error rate command and status register (SRIO_P2ERCSR)

The port 2 error rate command and status register (P2ERCSR), is a 32-bit register used with the port 2 error rate threshold register to monitor and control the reporting of transmission errors.

Address: C_0000h base + 6A8h offset = C_06A8h

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	ERB								Reserved						ERR		PER								ERC							
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRIO_P2ERCSR field descriptions

Field	Description
0–7 ERB	These bits provide the error rate bias value. 0x00 Do not decrement the error rate counter 0x01 Decrement every 1 ms (±34%) 0x02 Decrement every 10 ms (±34%) 0x04 Decrement every 100 ms (±34%) 0x08 Decrement every 1 s (±34%) 0x10 Decrement every 10 s (±34%) 0x20 Decrement every 100 s (±34%) 0x40 Decrement every 1000 s (±34%) 0x80 Decrement every 10000 s (±34%) Other values are reserved and cause undefined operation.
8–13 -	This field is reserved. Reserved
14–15 ERR	These bits limit the incrementing of the error rate counter above the failed threshold trigger. 00 Only count 2 errors above 01 Only count 4 errors above 10 Only count 16 error above 11 Do not limit incrementing the error rate count Note that the Error Rate Counter never increments above 0xFF, even if the combination of the settings of ERR and the failed threshold trigger might imply that it might.

Table continues on the next page...

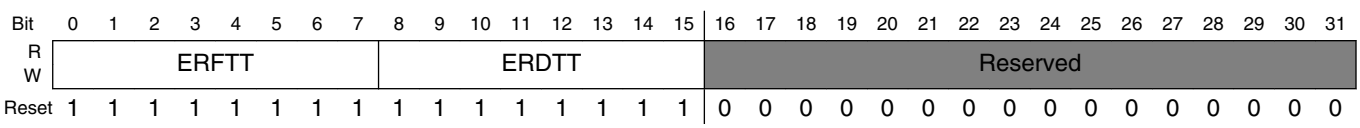
SRIO_P2ERCSR field descriptions (continued)

Field	Description
16–23 PER	Peak error rate. Contains the peak value attained by the error rate counter
24–31 ERC	Error rate counter. These bits maintain a count of the number of transmission errors that have been detected by the port, decremented by the Error Rate Bias mechanism, to create an indication of the link error rate. Software should not attempt to write this field to a value higher than failed threshold trigger plus the number of errors specified in the ERR field (the maximum ERC value).

21.5.52 Port 2 Error rate threshold command and status register (SRIO_P2ERTCSR)

The port 2 error rate threshold command and status register (P2ERTCSR), is a 32-bit register used to control the reporting of the link status to the system host.

Address: C_0000h base + 6ACh offset = C_06ACh



SRIO_P2ERTCSR field descriptions

Field	Description
0–7 ERFTT	Error rate failed threshold trigger. These bits provide the threshold value for reporting an error condition due to a possibly broken link. The P2ESCSR[OFE] bit is not set if ERFTT is written to a value lower than or equal to the P2ERCSR[ERC]. Please see Table 21-267 for more details. 0x00 Disable the Error Rate Failed Threshold Trigger. 0x01 Set the error reporting threshold to 1. 0x02 Set the error reporting threshold to 2. 0xFF Set the error reporting threshold to 255.
8–15 ERDTT	Error rate degraded threshold trigger. These bits provide the threshold value for reporting an error condition due to a degrading link. The P2ESCSR[ODE] bit is not set if ERDTT is written to a value lower than or equal to the P2ERCSR[ERC]. Please see Table 21-267 for more details. 0x00 Disable the Error Rate Degraded Threshold Trigger. 0x01 Set the error reporting threshold to 1.

Table continues on the next page...

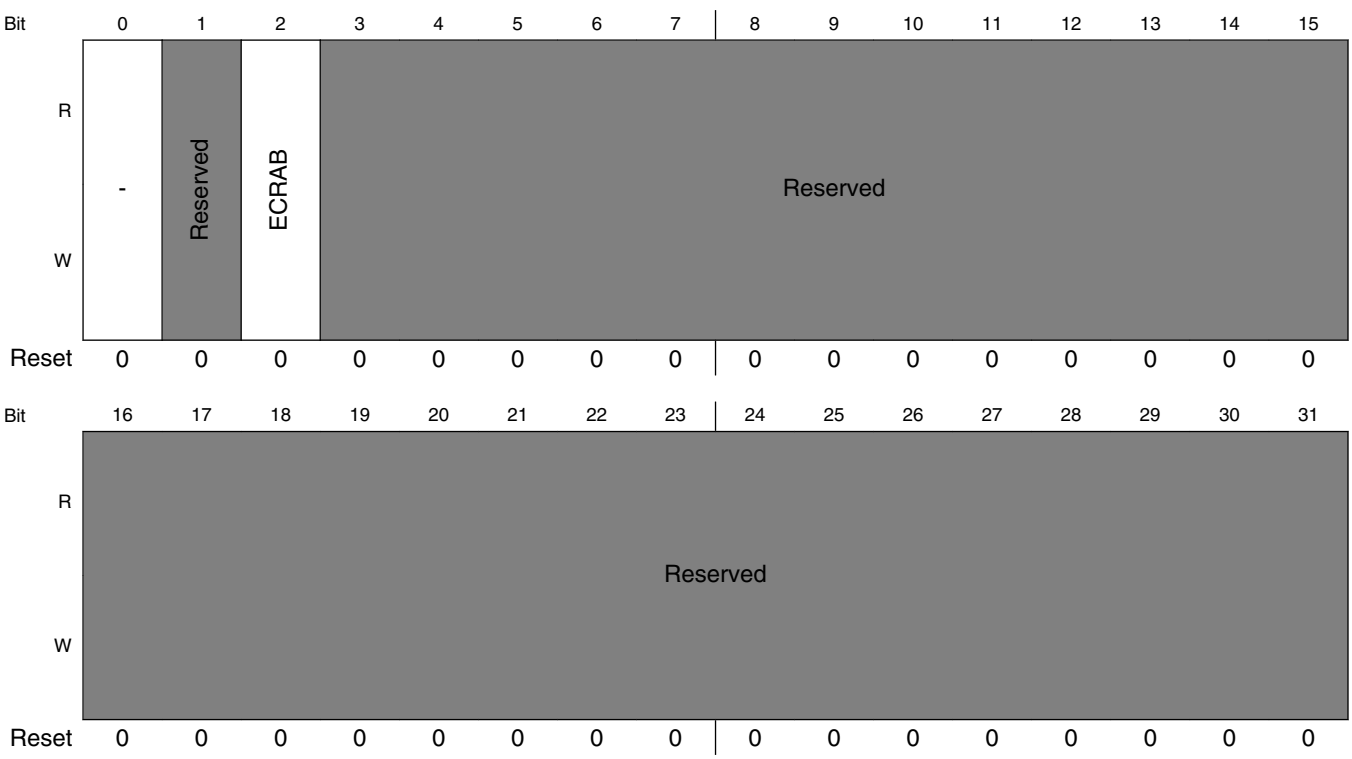
SRIO_P2ERTCSR field descriptions (continued)

Field	Description
	0x02 Set the error reporting threshold to 2. 0xFF Set the error reporting threshold to 255.
16–31 -	This field is reserved. Reserved

21.5.53 Logical layer configuration register (SRIO_LLCR)

The logical layer configuration register (LLCR), contains general port-common logical layer mode enables.

Address: C_0000h base + 1_0004h offset = D_0004h



SRIO_LLCR field descriptions

Field	Description
0 -	This bit is unused. It is readable and writable.
1 -	This field is reserved. Reserved

Table continues on the next page...

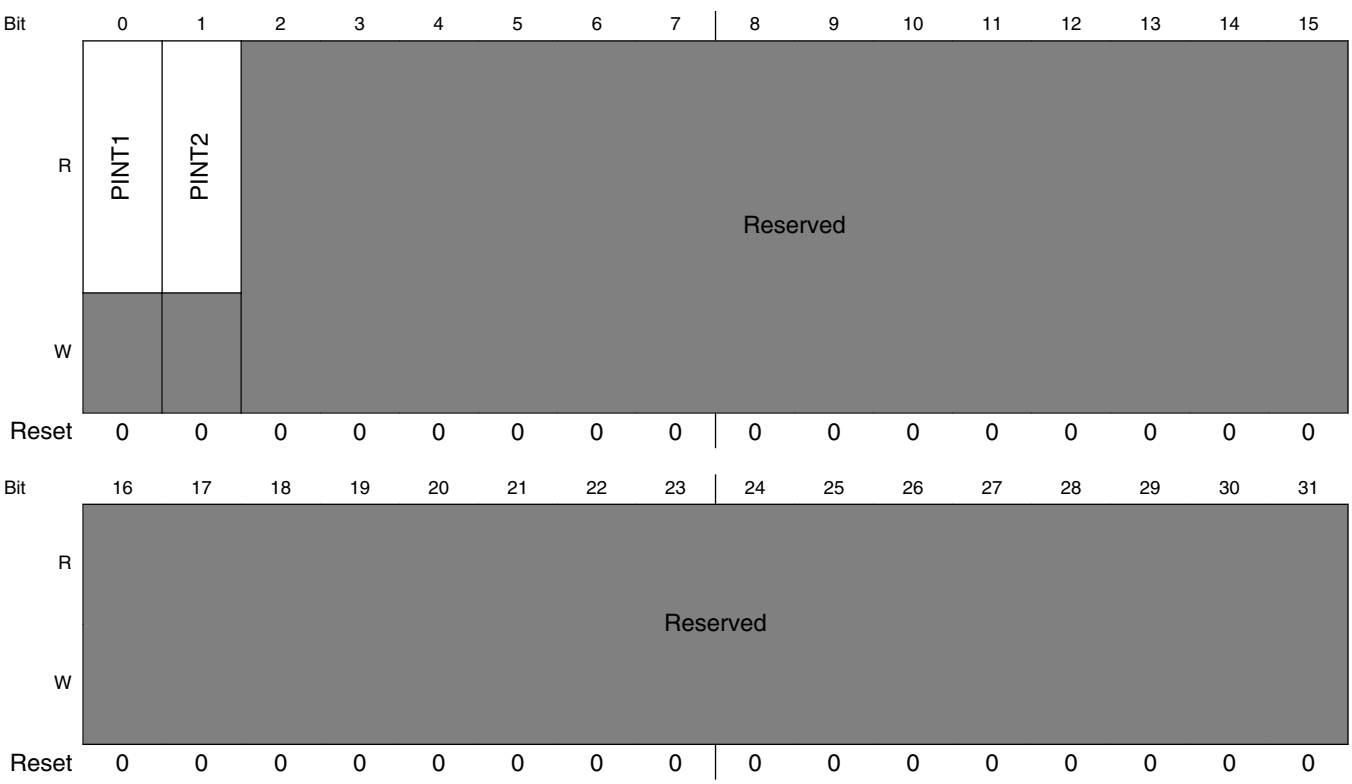
SRIO_LLCR field descriptions (continued)

Field	Description
2 ECRAB	External configuration register access block. When set, all maintenance requests and accesses that hit LCSBA1CSR are blocked; reads return all 0's, and writes are ignored (both return done response). When clear, any external RapidIO device can access registers.
3–31 -	This field is reserved. Reserved

21.5.54 Error / port-write interrupt status register (SRIO_EPWISR)

The error/port-write interrupt status register (EPWISR), contains status bits of the interrupts that have been generated by any port for physical or logical/transport layer errors. Because errors from all ports are reported to the core with one interrupt signal, this register provides the core with quick access to where the error occurred. This register is read only.

Address: C_0000h base + 1_0010h offset = D_0010h



SRIO_EPWISR field descriptions

Field	Description
0 PINT1	A physical or logical/transport error interrupt was generated for port 1.
1 PINT2	A physical or logical/transport error interrupt was generated for port 2.
2–31 -	This field is reserved. Reserved.

21.5.55 Logical retry error threshold configuration register (SRIO_LRETCR)

The logical retry error threshold configuration register (LRETCR), contains the retry error threshold for the logical layer. When the number of consecutive logical retries for a given packet is greater than to this value, an error interrupt is generated. Note that the number of retries must be greater than this value unlike other registers that define a retry threshold.

Address: C_0000h base + 1_0020h offset = D_0020h

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																								RET							
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	1	1	1	1	1	1	1	1

SRIO_LRETCR field descriptions

Field	Description
0–23 -	This field is reserved. Reserved
24–31 RET	Retry error threshold. These bits provide the threshold value for the number of consecutive logical retries (for GSM responses) received for a given packet that causes RAPIDIO ENDPOINT to report an error condition. 0x00 Disable the RET 0x01 Set the error reporting threshold to 1 0xFF Set the error reporting threshold to 255

21.5.56 Physical retry error threshold configuration register (SRIO_PRETCR)

The physical retry error threshold configuration register (PRETCR), contains the retry error threshold for the physical layer. When the number of consecutive ACK-retries is greater than or equal to this value, an error interrupt is generated.

Address: C_0000h base + 1_0080h offset = D_0080h

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																								RET							
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	1	1	1	1	1	1	1	1

SRIO_PRETCR field descriptions

Field	Description
0–23 -	This field is reserved. Reserved
24–31 RET	Retry error threshold. These bits provide the threshold value for the number of consecutive ACK retries received that causes RAPIDIO ENDPOINT to report an error condition. 0x00 Disable the RET 0x01 Set the error reporting threshold to 1 0xFF Set the error reporting threshold to 255

21.5.57 Port 1 Alternate device ID command and status register (SRIO_P1ADIDCSR)

The port 1 alternate device id CSR contains an alternate deviceID. This register should be used on a multi-port device to enable separate deviceIDs for each port. It is intended that this register, should be enabled before the master enabled bit of the PGCCSR is set, such that when it is enabled, all other devices in the RapidIO system (including switches) send packets to and receive packets from the deviceID contained in this register, instead of the deviceID contained in BDIDCSR.

When the alternate deviceID is enabled, the inbound RapidIO endpoint only accepts packets sent with the deviceID contained in P1ADIDCSR or with the deviceID contained in BDIDCSR (except during Accept All mode, during which the inbound RapidIO endpoint accepts packets using the same common transport system). In addition, the

outbound RapidIO endpoint only generates requests using the deviceID contained in P1ADIDCSR; it generates responses with the deviceID given in the original request packet (either from P1ADIDCSR or BDIDCSR).

Address: C_0000h base + 1_0100h offset = D_0100h

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

SRIO_P1ADIDCSR field descriptions

Field	Description
0 ADE	Alternate device ID enable When set, this bit causes the port to use the deviceID specified in this register instead of the deviceid specified in BDIDCSR
1–7 -	This field is reserved. Reserved
8–15 ADID	Alternate device ID for the device in a small transport system
16–31 LADID	Alternate device ID for the device in a large transport system

21.5.58 Port 1 Accept-all configuration register (SRIO_P1AACR)

The port 1 pass-through/accept-all configuration register (P*n*AACR), contains information on pass-through and accept-all mode.

Address: C_0000h base + 1_0120h offset = D_0120h

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

SRIO_P1AACR field descriptions

Field	Description
0–30 -	This field is reserved. Reserved
31 AA	Accept all 1 All packets are accepted without checking the target ID. However, the tt field must be consistent with the common transport system specified by bit 27 of the processing element features CAR. 0 Normal RapidIO acceptance based on target ID.

21.5.59 Port 1 Logical Outbound Packet time-to-live configuration register (SRIO_P1LOPTTLCR)

The port 1 logical outbound packet time-to-live configuration register (P1LOPTTLCR), contains the time-to-live count for all ports on a device. This packet time-to-live counter starts when a packet is ready to be transmitted. If the packet is not successfully transmitted before the timer expires, the packet is discarded. Successfully transmitted means that a packet accept was received for the packet on the RIO interface. If the packet requires a response, an internal error response is returned after the response time-out occurs (PRTOCCSR). The packet time-to-live counter prevents the local processor from being stalled when packets cannot be successfully transmitted (acknowledged with an accept by the link partner at the physical level). The value of this register should always be larger than the link time-out value (PLTOCCSR). By default, this time-out value is disabled (all zeros). The timer resolution is specified between 178-298ns, which correlates to a maximum time-out value of approximately 3-5s.

When the packet time-to-live counter expires, PnPCR[OBDEN] is automatically set. PnPCR[OBDEN] must be cleared by software.

Address: C_0000h base + 1_0124h offset = D_0124h

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	TV																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

SRIO_P1LOPTTLCR field descriptions

Field	Description
0–23 TV	Time-out value. Setting to all zeros disables the time-to-live time-out timer. This value is loaded each time the time-to-live time-out timer starts.
24–31 -	This field is reserved. Reserved

21.5.60 Port 1 Implementation error command and status register (SRIO_P1IECSR)

The port 1 implementation error command and status register (PnIECSR), contains status bits that are asserted whenever an implementation-defined error occurs.

Address: C_0000h base + 1_0130h offset = D_0130h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RETE	Reserved														
W	w1c															
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															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

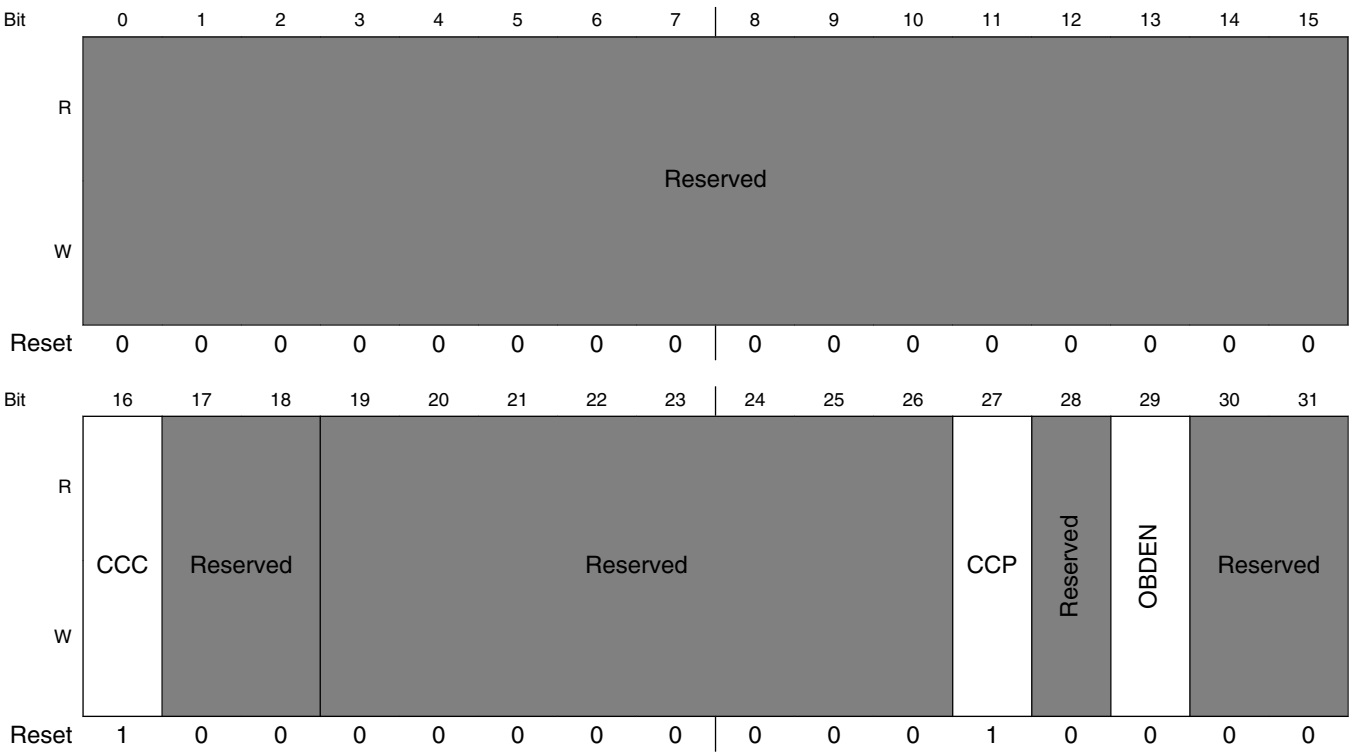
SRIO_P1IECSR field descriptions

Field	Description
0 RETE	Retry error threshold exceeded. This bit is asserted when the number of consecutive retries has reached retry error threshold in the retry error threshold register. This bit is cleared by writing a 1 to it. This bit sets again if another retry is received and the number of consecutive retries continues to exceed the retry error threshold.
1–31 -	This field is reserved. Reserved

21.5.61 Port 1 Physical configuration register (SRIO_P1PCR)

The port 1 physical configuration register (PnPCR), contains general physical layer protocol and link mode enables.

Address: C_0000h base + 1_0140h offset = D_0140h



SRIO_P1PCR field descriptions

Field	Description
0–15 -	This field is reserved. Reserved
16 CCC	CRC checking enable-control symbol. When set, CRC is checked on received control symbols. When cleared, no CRC is checked on received control symbols.
17–18 -	This field is reserved. Reserved
19–26 -	This field is reserved. Reserved
27 CCP	CRC checking enable-packet. When set, CRC is checked on received packets. When cleared, no CRC is checked on received packets
28 -	This field is reserved. Reserved

Table continues on the next page...

SRIO_P1PCR field descriptions (continued)

Field	Description
29 OBDEN	Output buffer drain enable. When set, the output drains packets from the outbound buffer and does not send them out. This intentionally causes the inbound to time-out (when a response on a drained request was expected) and send an error response to the internal platform. A packet time-to-live time-out causes this bit to be set. (See Port 1 Logical Outbound Packet time-to-live configuration register (SRIO_P1LOPTTLCR).")
30–31 -	This field is reserved. Reserved

21.5.62 Port 1 Serial link command and status register (SRIO_P1SLCSR)

The port 1 serial link command and status register (PnSLCSR), contains status of the of the serial physical link.

Address: C_0000h base + 1_0158h offset = D_0158h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	LS0	LS1	LS2	LS3	Reserved				LA	Reserved						
W	w1c	w1c	w1c	w1c					w1c							
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															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRIO_P1SLCSR field descriptions

Field	Description
0 LS0	Lane sync achieved for lane 0. Write with 1 to clear
1 LS1	Lane sync achieved for lane 1. Write with 1 to clear
2 LS2	Lane sync achieved for lane 2. Write with 1 to clear
3 LS3	Lane sync achieved for lane 3. Write with 1 to clear.
4–7 -	This field is reserved. Reserved

Table continues on the next page...

SRIO_P1SLCSR field descriptions (continued)

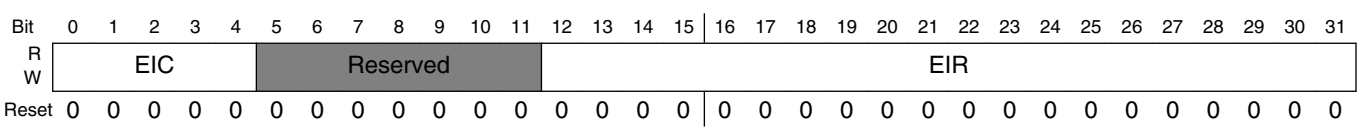
Field	Description
8 LA	Lane alignment achieved. Write with 1 to clear.
9–31 -	This field is reserved. Reserved

21.5.63 Port 1 Serial link error injection configuration register (SRIO_P1SLEICR)

The port 1 serial link error injection configuration register (PnSLEICR), is used to control the injection of bit errors into the transmit bit stream.

The P1SLEICR register is used to generate pseudo-random errors into the outbound serial RapidIO data stream. If the EIC field is any of the allowable non-zero values (as shown in the table above), then error injection is enabled for one lane or all four lanes, as selected by the EIC value. When enabled, at pseudo-random intervals, an error is injected by inverting a single bit in the outgoing data stream. This occurs only in the lane(s) that have error injection enabled. The range of the pseudo-random value (delay between injected errors) is controlled by the EIR field, as described above. That is, the value of EIR, multiplied by 32, determines the maximum number of character times between injected errors.

Address: C_0000h base + 1_0160h offset = D_0160h



SRIO_P1SLEICR field descriptions

Field	Description
0–4 EIC	Error injection control. Enables and controls serial link error injection as follows: 00000 Error injection is disabled. 10000 Error injection, lane 0 only 01000 Error injection, lane 1 only 00100 Error injection, lane 2 only 00010 Error injection, lane 3 only 11110 Error injection, all 4 lanes simultaneously 11111 Error injection, randomly distributed over all 4 lanes All other values are reserved.

Table continues on the next page...

SRIO_P1SLEICR field descriptions (continued)

Field	Description
5–11 -	This field is reserved. Reserved
12–31 EIR	Error injection range. The value of EIR x 32 determines the maximum value of the pseudo-random delay between errors. For example, a value of 0x1 would indicate a maximum delay of 32 character times. Value within this register should be right-justified.

21.5.64 Port 1 Arbitration 0 Tx Configuration Register (SRIO_P1A0TxCR)

The port 1 arbitration 0 Tx configuration register (PnA0TxCR), is used to control the arbitration of packets within the serial RapidIO controller's transmit logic (Tx), specifically at arbitration point '0'. Arbitration," for details on this register's use.

Address: C_0000h base + 1_0164h offset = D_0164h

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																						SPSA_TH_MSGREQ[0:9]									
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	1	1	1	1	1	1	1	1

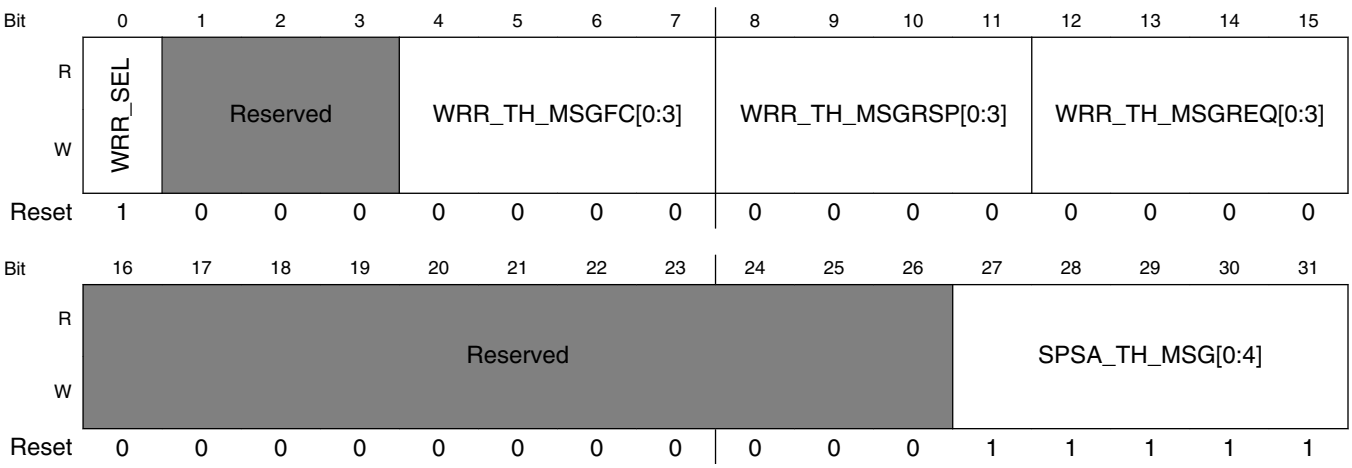
SRIO_P1A0TxCR field descriptions

Field	Description
0–21 -	This field is reserved. Reserved
22–31 SPSA_TH_ MSGREQ[0:9]	Strict priority with starvation avoidance (arbitration point '0' Tx) threshold (count) for message unit request packets. Enables and controls the arbitration as follows ... 0000000000 Disabled, Strict priority with starvation avoidance disabled. A low priority message unit request packet will never promote to highest priority. Note: Use at own risk; this may cause deadlocks. 0000000001 Count of 1, Strict priority with starvation avoidance enabled. A low priority message unit request packet may lose strict priority arbitration 1 consecutive time before it promotes to highest priority (starvation avoidance). 1111111111 Count of 1023: Strict priority with starvation avoidance enabled. A low priority message unit request packet may lose strict priority arbitration 1023 consecutive times before it promotes to highest priority (starvation avoidance).

21.5.65 Port 1 Arbitration 1 Tx Configuration Register (SRIO_P1A1TxCR)

The port 1 arbitration 1 Tx configuration register (PnA1TxCR), is used to control the arbitration of packets within the serial RapidIO controller's transmit logic (Tx), specifically at arbitration point '1'. See Arbitration," for details on this register's use.

Address: C_0000h base + 1_0168h offset = D_0168h



SRIO_P1A1TxCR field descriptions

Field	Description
0 WRR_SEL	Weighted-round-robin select. Indicates the arbitration type to use at arbitration point '1' Tx ... 0 Strict priority with starvation avoidance: 'SPSA_TH_MSG[4:0]' is valid. 1 Weigthed-round-robin: 'WRR_TH_MSGFC[3:0]', 'WRR_TH_MSGRSP[0:3]', and 'WRR_TH_MSGREQ[3:0]' are valid.
1–3 -	This field is reserved. Reserved
4–7 WRR_TH_ MSGFC[0:3]	Weighted-round-robin (arbitration point '1' Tx) threshold (weight) for message unit flow control packets. Enables and controls the arbitration as follows ... 0000 Weight of 1: 1 consecutive message unit flow control packet before considering other packets. FFFF Weigth of 16: 16 consecutive message unit flow control packets before considering other packets.
8–11 WRR_TH_ MSGRSP[0:3]	Weighted-round-robin (arbitration point '1' Tx) threshold (weight) for message unit response packets. Enables and controls the arbitration as follows ... 0000 Weight of 1: 1 consecutive message unit response packet before considering other packets. FFFF Weigth of 16: 16 consecutive message unit response packets before considering other packets.

Table continues on the next page...

SRIO_P1A1TxCR field descriptions (continued)

Field	Description
12–15 WRR_TH_ MSGREQ[0:3]	Weighted-round-robin (arbitration point '1' Tx) threshold (weight) for message unit request packets. Enables and controls the arbitration as follows ... 0000 Weight of 1: 1 consecutive message unit request packet before considering other packets. FFFF Weigth of 16: 16 consecutive message unit request packets before considering other packets.
16–26 -	This field is reserved. Reserved
27–31 SPSA_TH_ MSG[0:4]	Strict priority with starvation avoidance (arbitration point '1' Tx) threshold (count) for message unit packets. Enables and controls the arbitration as follows ... 00000 Disabled: Strict priority with starvation avoidance disabled. A low priority message unit packet will never promote to highest priority. Note: Use at own risk; this may cause deadlocks. 00001 Count of 1: Strict priority with starvation avoidance enabled. A low priority message unit packet may lose strict priority arbitration 1 consecutive time before it promotes to highest priority (starvation avoidance). 11111 Count of 31: Strict priority with starvation avoidance enabled. A low priority message unit packet may lose strict priority arbitration 31 consecutive times before it promotes to highest priority (starvation avoidance).

21.5.66 Port 1 Arbitration 2 Tx Configuration Register (SRIO_P1A2TxCR)

The port 1 arbitration 2 Tx configuration register (PnA2TxCR), is used to control the arbitration of packets within the serial RapidIO controller's transmit logic (Tx), specifically at arbitration point '2'.

See Arbitration,"for details on this register's use

.

Address: C_0000h base + 1_016Ch offset = D_016Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	WRR_SEL	Reserved			WRR_TH_MSG[0:3]				Reserved				WRR_TH_SYS[0:3]			
W																
Reset	1	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											SPSA_TH_SRIO[0:4]				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1

SRIO_P1A2TxCR field descriptions

Field	Description
0 WRR_SEL	Weighted-round-robin select. Indicates the arbitration type to use at arbitration point '2' Tx ... 0 Strict priority with starvation avoidance: 'SPSA_TH_SRIO[4:0]' is valid. 1 Weigthed-round-robin: 'WRR_TH_MSG[3:0]' and 'WRR_TH_SYS[3:0]' are valid.
1–3 -	This field is reserved. Reserved
4–7 WRR_TH_ MSG[0:3]	Weighted-round-robin (arbitration point '2' Tx) threshold (weight) for message unit packets. Enables and controls the arbitration as follows: 0000 Weight of 1: 1 consecutive message unit packet before considering other packets. 1111 Weigth of 16: 16 consecutive message unit packets before considering other packets.
8–11 -	This field is reserved. Reserved
12–15 WRR_TH_ SYS[0:3]	Weighted-round-robin (arbitration point '2' Tx) threshold (weight) for system packets. Enables and controls the arbitration as follows: 0000 Weight of 1: 1 consecutive system packet before considering other packets. 1111 Weigth of 16: 16 consecutive system packets before considering other packets.
16–26 -	This field is reserved. Reserved
27–31 SPSA_TH_ SRIO[0:4]	Strict priority with starvation avoidance (arbitration point '2' Tx) threshold (count) for SRIO packets. Enables and controls the arbitration as follows ... 00000 Disabled: Strict priority with starvation avoidance disabled. A low priority SRIO packet will never promote to highest priority. Note: Use at own risk; this may cause deadlocks. 01111 Count of 1: Strict priority with starvation avoidance enabled. A low priority SRIO packet may lose strict priority arbitration 1 consecutive time before it promotes to highest priority (starvation avoidance). 11111 Count of 31: Strict priority with starvation avoidance enabled. A low priority SRIO packet may lose strict priority arbitration 31 consecutive times before it promotes to highest priority (starvation avoidance).

21.5.67 Port 1 Message Request Tx Buffer Allocation Configuration Register 0 (SRIO_P1MReqTxBACR0)

The port 1 message request Tx buffer allocation configuration register 0 (PnMReqTxBACR0), is used to control the buffer allocation of message unit request packets within the serial RapidIO controller's transmit logic (Tx).

NOTE

There are only 15 buffers that hold message unit request packets. The total value of the fields within this register, as well as Port 1 Message Request Tx Buffer Allocation Configuration

Register 2 (SRIO_P1MReqTxBACR2) must not exceed 15 (undefined behavior will result). See Buffer Allocation, for details on this register's use.

Address: C_0000h base + 1_0170h offset = D_0170h

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				DedAG0[0:3]				Reserved				DedAG1[0:3]				Reserved				DedAG2[0:3]				Reserved				DedAG3[0:3]			
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

SRIO_P1MReqTxBACR0 field descriptions

Field	Description
0–3 -	This field is reserved. Reserved
4–7 DedAG0[0:3]	Dedicated arbitration group 0. Controls the buffer allocation of arbitration group 0 message unit request packets as follows: 0x0 No dedicated buffers allocated: No dedicated buffers allocated for arbitration group 0 message unit request packets. Note: Use at own risk; this may cause deadlocks. 0x1 1 dedicated buffer allocated: 1 dedicated buffer allocated for arbitration group 0 message unit request packets. 0xF 15 dedicated buffers allocated: 15 dedicated buffers allocated for arbitration group 0 message unit request packets.
8–11 -	This field is reserved. Reserved
12–15 DedAG1[0:3]	Dedicated arbitration group 1. Controls the buffer allocation of arbitration group 1 message unit request packets as follows: 0x0 No dedicated buffers allocated: No dedicated buffers allocated for arbitration group 1 message unit request packets. Note: Use at own risk; this may cause deadlocks. 0x1 1 dedicated buffer allocated: 1 dedicated buffer allocated for arbitration group 1 message unit request packets. 0xF 15 dedicated buffers allocated: 15 dedicated buffers allocated for arbitration group 1 message unit request packets.
16–19 -	This field is reserved. Reserved
20–23 DedAG2[0:3]	Dedicated arbitration group 2. Controls the buffer allocation of arbitration group 2 message unit request packets as follows: 0x0 No dedicated buffers allocated: No dedicated buffers allocated for arbitration group 2 message unit request packets. Note: Use at own risk; this may cause deadlocks. 0x1 1 dedicated buffer allocated: 1 dedicated buffer allocated for arbitration group 2 message unit request packets. 0xF 15 dedicated buffers allocated: 15 dedicated buffers allocated for arbitration group 2 message unit request packets.
24–27 -	This field is reserved. Reserved
28–31 DedAG3[0:3]	Dedicated arbitration group 3. Controls the buffer allocation of arbitration group 3 message unit request packets as follows:

Table continues on the next page...

SRIO_P1MReqTxBACR0 field descriptions (continued)

Field	Description
	0x0 No dedicated buffers allocated: No dedicated buffers allocated for arbitration group 3 message unit request packets. Note: Use at own risk; this may cause deadlocks. 0x1 1 dedicated buffer allocated: 1 dedicated buffer allocated for arbitration group 3 message unit request packets. 0xF 15 dedicated buffers allocated: 15 dedicated buffers allocated for arbitration group 3 message unit request packets.

21.5.68 Port 1 Message Request Tx Buffer Allocation Configuration Register 2 (SRIO_P1MReqTxBACR2)

The port 1 message request Tx buffer allocation configuration register 2 (PnMReqTxBACR2), is used to control the buffer allocation of message unit request packets within the serial RapidIO controller's transmit logic (Tx).

NOTE

There are only 15 buffers that hold message unit request packets. The total value of the fields within this register, as well asPort 1 Message Request Tx Buffer Allocation Configuration Register 0 (SRIO_P1MReqTxBACR0) and , must not exceed 15 (undefined behavior will result). See Buffer Allocation," for details on this register's use.

Address: C_0000h base + 1_0178h offset = D_0178h

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				GenAG[0:3]				Reserved																							
Reset	0	0	0	0	0	1	1	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

SRIO_P1MReqTxBACR2 field descriptions

Field	Description
0–3 -	This field is reserved. Reserved
4–7 GenAG[0:3]	Generic arbitration group. Controls the buffer allocation of any arbitration group message unit request packets as follows ... 0000 No generic buffers allocated: No generic buffers allocated for any arbitration group message unit request packets. Note: Use at own risk; this may cause deadlocks. 0001 1 generic buffer allocated: 1 generic buffer allocated for any arbitration group message unit request packets. 000F 15 generic buffers allocated: 15 generic buffers allocated for any arbitration group message unit request packets.
8–31 -	This field is reserved. Reserved

21.5.69 Port 1 Message Response / Flow Control Tx Buffer Allocation Configuration Register (SRIO_P1MRspFcTxBACR)

The port 1 message response/flow control Tx buffer allocation configuration register (PnMRspFcTxBACR), is used to control the buffer allocation of message unit response and flow control packets within the serial RapidIO controller's transmit logic (Tx).

NOTE

There are only 16 buffers that hold message unit response and flow control packets. The total value of the fields within this register, must not exceed 16 (undefined behavior will result). See Buffer Allocation," for details on this register's use.

Address: C_0000h base + 1_017Ch offset = D_017Ch

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			DedRsp[0:4]					Reserved			DedFc[0:4]					Reserved			GenRspFc[0:4]				Reserved							
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0

SRIO_P1MRspFcTxBACR field descriptions

Field	Description
0–2 -	This field is reserved. Reserved
3–7 DedRsp[0:4]	Dedicated response. Controls the buffer allocation of message unit response packets as follows ... 0x00 No dedicated buffers allocated: No dedicated buffers allocated for message unit response packets. Note: Use at own risk; this may cause deadlocks. 0x01 1 dedicated buffer allocated: 1 dedicated buffer allocated for message unit response packets. 0x10 16 dedicated buffers allocated: 16 dedicated buffers allocated for message unit response packets.
8–10 -	This field is reserved. Reserved
11–15 DedFc[0:4]	Dedicated flow control. Controls the buffer allocation of message unit flow control packets as follows ... 0x00 No dedicated buffers allocated: No dedicated buffers allocated for message unit flow control packets. Note: Use at own risk; this may cause deadlocks. 0x01 1 dedicated buffer allocated: 1 dedicated buffer allocated for message unit flow control packets. 0x10 16 dedicated buffers allocated: 16 dedicated buffers allocated for message unit flow control packets.
16–18 -	This field is reserved. Reserved
19–23 GenRspFc[0:4]	Generic response and flow control. Controls the buffer allocation of message unit response and flow control packets as follows ...

Table continues on the next page...

SRIO_P1MRspFcTxBACR field descriptions (continued)

Field	Description
	0x00 No generic buffers allocated: No generic buffers allocated for message unit response and flow control packets. Note: Use at own risk; this may cause deadlocks. 0x01 1 generic buffer allocated: 1 generic buffer allocated for message unit response and flow control packets. 0x10 16 generic buffers allocated: 16 generic buffers allocated for message unit response and flow control packets.
24–31 -	This field is reserved. Reserved

21.5.70 Port 2 Alternate device ID command and status register (SRIO_P2ADIDCSR)

The port 2 alternate device id CSR contains an alternate deviceID. This register should be used on a multi-port device to enable separate deviceIDs for each port. It is intended that this register, should be enabled before the master enabled bit of the PGCCSR is set, such that when it is enabled, all other devices in the RapidIO system (including switches) send packets to and receive packets from the deviceID contained in this register, instead of the deviceID contained in BDIDCSR.

When the alternate deviceID is enabled, the inbound RapidIO endpoint only accepts packets sent with the deviceID contained in P2ADIDCSR or with the deviceID contained in BDIDCSR (except during Accept All mode, during which the inbound RapidIO endpoint accepts packets using the same common transport system). In addition, the outbound RapidIO endpoint only generates requests using the deviceID contained in P2ADIDCSR; it generates responses with the deviceID given in the original request packet (either from P2ADIDCSR or BDIDCSR).

Address: C_0000h base + 1_0180h offset = D_0180h

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

SRIO_P2ADIDCSR field descriptions

Field	Description
0 ADE	Alternate device ID enable

Table continues on the next page...

SRIO_P2ADIDCSR field descriptions (continued)

Field	Description
	When set, this bit causes the port to use the deviceID specified in this register instead of the deviceid specified in BDIDCSR
1–7 -	This field is reserved. Reserved
8–15 ADID	Alternate device ID for the device in a small transport system
16–31 LADID	Alternate device ID for the device in a large transport system

21.5.71 Port 2 Accept-all configuration register (SRIO_P2AACR)

The port 2 pass-through/accept-all configuration register (P*n*AACR), contains information on pass-through and accept-all mode.

Address: C_0000h base + 1_01A0h offset = D_01A0h

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

SRIO_P2AACR field descriptions

Field	Description
0–30 -	This field is reserved. Reserved
31 AA	Accept all 1 All packets are accepted without checking the target ID. However, the tt field must be consistent with the common transport system specified by bit 27 of the processing element features CAR. 0 Normal RapidIO acceptance based on target ID.

21.5.72 Port 2 Logical Outbound Packet time-to-live configuration register (SRIO_P2LOPTTLCR)

The port 2 logical outbound packet time-to-live configuration register (P2LOPTTLCR), contains the time-to-live count for all ports on a device. This packet time-to-live counter starts when a packet is ready to be transmitted. If the packet is not successfully transmitted before the timer expires, the packet is discarded. Successfully transmitted means that a packet accept was received for the packet on the RIO interface. If the packet requires a response, an internal error response is returned after the response time-out occurs (PRTOCCSR). The packet time-to-live counter prevents the local processor from being stalled when packets cannot be successfully transmitted (acknowledged with an accept by the link partner at the physical level). The value of this register should always be larger than the link time-out value (PLTOCCSR). By default, this time-out value is disabled (all zeros). The timer resolution is specified between 178-298ns, which correlates to a maximum time-out value of approximately 3-5s.

When the packet time-to-live counter expires, PnPCR[OBDEN] is automatically set. PnPCR[OBDEN] must be cleared by software.

Address: C_0000h base + 1_01A4h offset = D_01A4h

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	TV																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

SRIO_P2LOPTTLCR field descriptions

Field	Description
0–23 TV	Time-out value. Setting to all zeros disables the time-to-live time-out timer. This value is loaded each time the time-to-live time-out timer starts.
24–31 -	This field is reserved. Reserved

21.5.73 Port 2 Implementation error command and status register (SRIO_P2IECSR)

The port 2 implementation error command and status register (PnIECSR), contains status bits that are asserted whenever an implementation-defined error occurs.

Address: C_0000h base + 1_01B0h offset = D_01B0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	RETE	Reserved														
W	w1c															
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															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

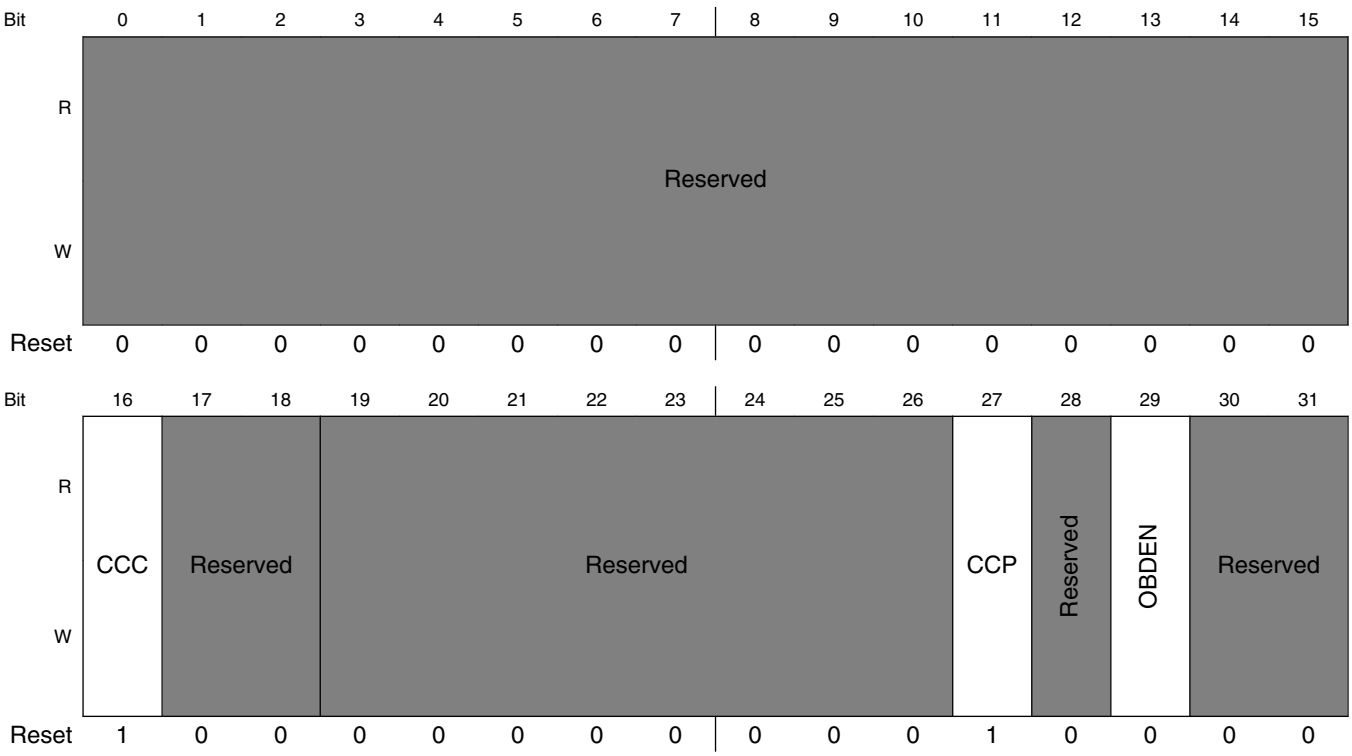
SRIO_P2IECSR field descriptions

Field	Description
0 RETE	Retry error threshold exceeded. This bit is asserted when the number of consecutive retries has reached retry error threshold in the retry error threshold register. This bit is cleared by writing a 1 to it. This bit sets again if another retry is received and the number of consecutive retries continues to exceed the retry error threshold.
1–31 -	This field is reserved. Reserved

21.5.74 Port 2 Physical configuration register (SRIO_P2PCR)

The port 2 physical configuration register (PnPCR), contains general physical layer protocol and link mode enables.

Address: C_0000h base + 1_01C0h offset = D_01C0h



SRIO_P2PCR field descriptions

Field	Description
0–15 -	This field is reserved. Reserved
16 CCC	CRC checking enable-control symbol. When set, CRC is checked on received control symbols. When cleared, no CRC is checked on received control symbols.
17–18 -	This field is reserved. Reserved
19–26 -	This field is reserved. Reserved
27 CCP	CRC checking enable-packet. When set, CRC is checked on received packets. When cleared, no CRC is checked on received packets
28 -	This field is reserved. Reserved

Table continues on the next page...

SRIO_P2PCR field descriptions (continued)

Field	Description
29 OBDEN	Output buffer drain enable. When set, the output drains packets from the outbound buffer and does not send them out. This intentionally causes the inbound to time-out (when a response on a drained request was expected) and send an error response to the internal platform. A packet time-to-live time-out causes this bit to be set. See Port 2 Logical Outbound Packet time-to-live configuration register (SRIO_P2LOPTTLCR).")
30–31 -	This field is reserved. Reserved

21.5.75 Port 2 Serial link command and status register (SRIO_P2SLCSR)

The port 2 serial link command and status register (PnSLCSR), contains status of the of the serial physical link.

Address: C_0000h base + 1_01D8h offset = D_01D8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	LS0	LS1	LS2	LS3	Reserved				LA	Reserved						
W	w1c	w1c	w1c	w1c					w1c							
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															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRIO_P2SLCSR field descriptions

Field	Description
0 LS0	Lane sync achieved for lane 0. Write with 1 to clear
1 LS1	Lane sync achieved for lane 1. Write with 1 to clear
2 LS2	Lane sync achieved for lane 2. Write with 1 to clear
3 LS3	Lane sync achieved for lane 3. Write with 1 to clear.
4–7 -	This field is reserved. Reserved

Table continues on the next page...

SRIO_P2SLCSR field descriptions (continued)

Field	Description
8 LA	Lane alignment achieved. Write with 1 to clear.
9–31 -	This field is reserved. Reserved

21.5.76 Port 2 Serial link error injection configuration register (SRIO_P2SLEICR)

The port 2 serial link error injection configuration register (PnSLEICR), is used to control the injection of bit errors into the transmit bit stream.

The P2SLEICR register is used to generate pseudo-random errors into the outbound serial RapidIO data stream. If the EIC field is any of the allowable non-zero values (as shown in the table above), then error injection is enabled for one lane or all four lanes, as selected by the EIC value. When enabled, at pseudo-random intervals, an error is injected by inverting a single bit in the outgoing data stream. This occurs only in the lane(s) that have error injection enabled. The range of the pseudo-random value (delay between injected errors) is controlled by the EIR field, as described above. That is, the value of EIR, multiplied by 32, determines the maximum number of character times between injected errors.

Address: C_0000h base + 1_01E0h offset = D_01E0h

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	EIC					Reserved							EIR																			
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

SRIO_P2SLEICR field descriptions

Field	Description
0–4 EIC	Error injection control. Enables and controls serial link error injection as follows: 00000 Error injection is disabled. 10000 Error injection, lane 0 only 01000 Error injection, lane 1 only 00100 Error injection, lane 2 only 00010 Error injection, lane 3 only 11110 Error injection, all 4 lanes simultaneously 11111 Error injection, randomly distributed over all 4 lanes All other values are reserved.

Table continues on the next page...

SRIO_P2SLEICR field descriptions (continued)

Field	Description
5–11 -	This field is reserved. Reserved
12–31 EIR	Error injection range. The value of EIR x 32 determines the maximum value of the pseudo-random delay between errors. For example, a value of 0x1 would indicate a maximum delay of 32 character times. Value within this register should be right-justified.

21.5.77 Port 2 Arbitration 0 Tx Configuration Register (SRIO_P2A0TxCR)

The port 2 arbitration 0 Tx configuration register (PnA0TxCR), is used to control the arbitration of packets within the serial RapidIO controller's transmit logic (Tx), specifically at arbitration point '0'. See Arbitration for details on this register's use.

Address: C_0000h base + 1_01E4h offset = D_01E4h

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																						SPSA_TH_MSGREQ[0:9]									
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	1	1	1	1	1	1	1	1

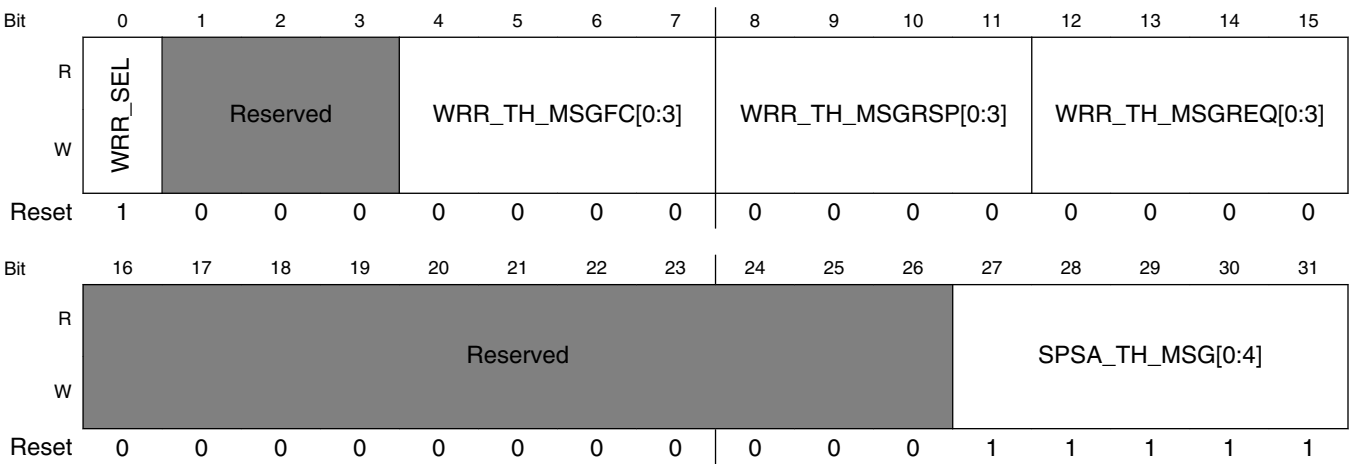
SRIO_P2A0TxCR field descriptions

Field	Description
0–21 -	This field is reserved. Reserved
22–31 SPSA_TH_ MSGREQ[0:9]	Strict priority with starvation avoidance (arbitration point '0' Tx) threshold (count) for message unit request packets. Enables and controls the arbitration as follows ... 0000000000 Disabled, Strict priority with starvation avoidance disabled. A low priority message unit request packet will never promote to highest priority. Note: Use at own risk; this may cause deadlocks. 0000000001 Count of 1, Strict priority with starvation avoidance enabled. A low priority message unit request packet may lose strict priority arbitration 1 consecutive time before it promotes to highest priority (starvation avoidance). 1111111111 Count of 1023: Strict priority with starvation avoidance enabled. A low priority message unit request packet may lose strict priority arbitration 1023 consecutive times before it promotes to highest priority (starvation avoidance).

21.5.78 Port 2 Arbitration 1 Tx Configuration Register (SRIO_P2A1TxCR)

The port 2 arbitration 1 Tx configuration register (PnA1TxCR), is used to control the arbitration of packets within the serial RapidIO controller's transmit logic (Tx), specifically at arbitration point '1'. See Arbitration for details on this register's use.

Address: C_0000h base + 1_01E8h offset = D_01E8h



SRIO_P2A1TxCR field descriptions

Field	Description
0 WRR_SEL	Weighted-round-robin select. Indicates the arbitration type to use at arbitration point '1' Tx ... 0 Strict priority with starvation avoidance: 'SPSA_TH_MSG[4:0]' is valid. 1 Weigthed-round-robin: 'WRR_TH_MSGFC[3:0]', 'WRR_TH_MSGRSP[0:3]', and 'WRR_TH_MSGREQ[3:0]' are valid.
1–3 -	This field is reserved. Reserved
4–7 WRR_TH_ MSGFC[0:3]	Weighted-round-robin (arbitration point '1' Tx) threshold (weight) for message unit flow control packets. Enables and controls the arbitration as follows ... 0000 Weight of 1: 1 consecutive message unit flow control packet before considering other packets. FFFF Weigth of 16: 16 consecutive message unit flow control packets before considering other packets.
8–11 WRR_TH_ MSGRSP[0:3]	Weighted-round-robin (arbitration point '1' Tx) threshold (weight) for message unit response packets. Enables and controls the arbitration as follows ... 0000 Weight of 1: 1 consecutive message unit response packet before considering other packets. FFFF Weigth of 16: 16 consecutive message unit response packets before considering other packets.

Table continues on the next page...

SRIO_P2A1TxCR field descriptions (continued)

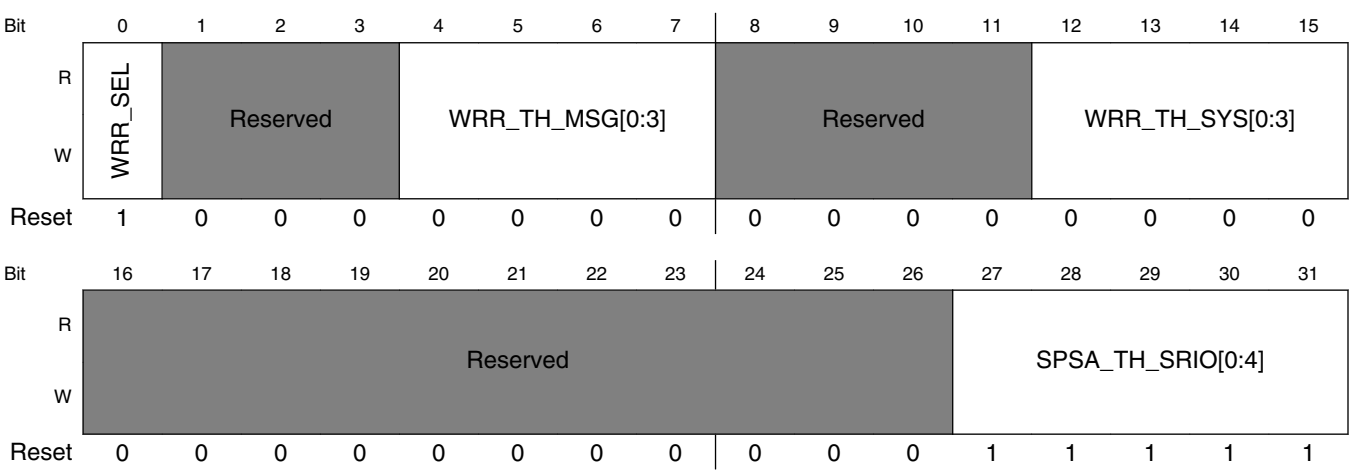
Field	Description
12–15 WRR_TH_ MSGREQ[0:3]	Weighted-round-robin (arbitration point '1' Tx) threshold (weight) for message unit request packets. Enables and controls the arbitration as follows ... 0000 Weight of 1: 1 consecutive message unit request packet before considering other packets. FFFF Weigth of 16: 16 consecutive message unit request packets before considering other packets.
16–26 -	This field is reserved. Reserved
27–31 SPSA_TH_ MSG[0:4]	Strict priority with starvation avoidance (arbitration point '1' Tx) threshold (count) for message unit packets. Enables and controls the arbitration as follows ... 00000 Disabled: Strict priority with starvation avoidance disabled. A low priority message unit packet will never promote to highest priority. Note: Use at own risk; this may cause deadlocks. 00001 Count of 1: Strict priority with starvation avoidance enabled. A low priority message unit packet may lose strict priority arbitration 1 consecutive time before it promotes to highest priority (starvation avoidance). 11111 Count of 31: Strict priority with starvation avoidance enabled. A low priority message unit packet may lose strict priority arbitration 31 consecutive times before it promotes to highest priority (starvation avoidance).

21.5.79 Port 2 Arbitration 2 Tx Configuration Register (SRIO_P2A2TxCR)

The port 2 arbitration 2 Tx configuration register (PnA2TxCR), is used to control the arbitration of packets within the serial RapidIO controller's transmit logic (Tx), specifically at arbitration point '2'.

See Arbitration,"for details on this register's use

Address: C_0000h base + 1_01ECh offset = D_01ECh



SRIO_P2A2TxCR field descriptions

Field	Description
0 WRR_SEL	Weighted-round-robin select. Indicates the arbitration type to use at arbitration point '2' Tx ... 0 Strict priority with starvation avoidance: 'SPSA_TH_SRIO[4:0]' is valid. 1 Weigthed-round-robin: 'WRR_TH_MSG[3:0]' and 'WRR_TH_SYS[3:0]' are valid.
1–3 -	This field is reserved. Reserved
4–7 WRR_TH_ MSG[0:3]	Weighted-round-robin (arbitration point '2' Tx) threshold (weight) for message unit packets. Enables and controls the arbitration as follows: 0000 Weight of 1: 1 consecutive message unit packet before considering other packets. 1111 Weigth of 16: 16 consecutive message unit packets before considering other packets.
8–11 -	This field is reserved. Reserved
12–15 WRR_TH_ SYS[0:3]	Weighted-round-robin (arbitration point '2' Tx) threshold (weight) for system packets. Enables and controls the arbitration as follows: 0000 Weight of 1: 1 consecutive system packet before considering other packets. 1111 Weigth of 16: 16 consecutive system packets before considering other packets.
16–26 -	This field is reserved. Reserved
27–31 SPSA_TH_ SRIO[0:4]	Strict priority with starvation avoidance (arbitration point '2' Tx) threshold (count) for SRIO packets. Enables and controls the arbitration as follows ... 00000 Disabled: Strict priority with starvation avoidance disabled. A low priority SRIO packet will never promote to highest priority. Note: Use at own risk; this may cause deadlocks. 01111 Count of 1: Strict priority with starvation avoidance enabled. A low priority SRIO packet may lose strict priority arbitration 1 consecutive time before it promotes to highest priority (starvation avoidance). 11111 Count of 31: Strict priority with starvation avoidance enabled. A low priority SRIO packet may lose strict priority arbitration 31 consecutive times before it promotes to highest priority (starvation avoidance).

21.5.80 Port 2 Message Request Tx Buffer Allocation Configuration Register 0 (SRIO_P2MReqTxBACR0)

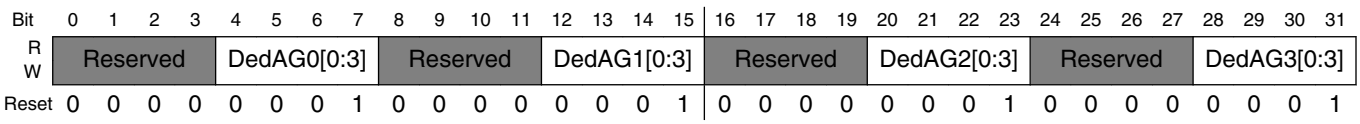
The port 2 message request Tx buffer allocation configuration register 0 (PnMReqTxBACR0), is used to control the buffer allocation of message unit request packets within the serial RapidIO controller's transmit logic (Tx).

NOTE

There are only 15 buffers that hold message unit request packets. The total value of the fields within this register, as well as Port 2 Message Request Tx Buffer Allocation Configuration

Register 2 (SRIO_P2MReqTxBACR2) must not exceed 15 (undefined behavior will result). See Buffer Allocation, for details on this register's use.

Address: C_0000h base + 1_01F0h offset = D_01F0h



SRIO_P2MReqTxBACR0 field descriptions

Field	Description
0–3 -	This field is reserved. Reserved
4–7 DedAG0[0:3]	Dedicated arbitration group 0. Controls the buffer allocation of arbitration group 0 message unit request packets as follows: 0x0 No dedicated buffers allocated: No dedicated buffers allocated for arbitration group 0 message unit request packets. Note: Use at own risk; this may cause deadlocks. 0x1 1 dedicated buffer allocated: 1 dedicated buffer allocated for arbitration group 0 message unit request packets. 0xF 15 dedicated buffers allocated: 15 dedicated buffers allocated for arbitration group 0 message unit request packets.
8–11 -	This field is reserved. Reserved
12–15 DedAG1[0:3]	Dedicated arbitration group 1. Controls the buffer allocation of arbitration group 1 message unit request packets as follows: 0x0 No dedicated buffers allocated: No dedicated buffers allocated for arbitration group 1 message unit request packets. Note: Use at own risk; this may cause deadlocks. 0x1 1 dedicated buffer allocated: 1 dedicated buffer allocated for arbitration group 1 message unit request packets. 0xF 15 dedicated buffers allocated: 15 dedicated buffers allocated for arbitration group 1 message unit request packets.
16–19 -	This field is reserved. Reserved
20–23 DedAG2[0:3]	Dedicated arbitration group 2. Controls the buffer allocation of arbitration group 2 message unit request packets as follows: 0x0 No dedicated buffers allocated: No dedicated buffers allocated for arbitration group 2 message unit request packets. Note: Use at own risk; this may cause deadlocks. 0x1 1 dedicated buffer allocated: 1 dedicated buffer allocated for arbitration group 2 message unit request packets. 0xF 15 dedicated buffers allocated: 15 dedicated buffers allocated for arbitration group 2 message unit request packets.
24–27 -	This field is reserved. Reserved
28–31 DedAG3[0:3]	Dedicated arbitration group 3. Controls the buffer allocation of arbitration group 3 message unit request packets as follows:

Table continues on the next page...

SRIO_P2MReqTxBACR0 field descriptions (continued)

Field	Description
	0x0 No dedicated buffers allocated: No dedicated buffers allocated for arbitration group 3 message unit request packets. Note: Use at own risk; this may cause deadlocks. 0x1 1 dedicated buffer allocated: 1 dedicated buffer allocated for arbitration group 3 message unit request packets. 0xF 15 dedicated buffers allocated: 15 dedicated buffers allocated for arbitration group 3 message unit request packets.

21.5.81 Port 2 Message Request Tx Buffer Allocation Configuration Register 2 (SRIO_P2MReqTxBACR2)

The port 2 message request Tx buffer allocation configuration register 2 (PnMReqTxBACR2), is used to control the buffer allocation of message unit request packets within the serial RapidIO controller's transmit logic (Tx).

NOTE

There are only 15 buffers that hold message unit request packets. The total value of the fields within this register, as well as Port 2 Message Request Tx Buffer Allocation Configuration Register 0 (SRIO_P2MReqTxBACR0) and must not exceed 15 (undefined behavior will result). See Buffer Allocation for details on this register's use.

Address: C_0000h base + 1_01F8h offset = D_01F8h

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				GenAG[0:3]				Reserved																							
Reset	0	0	0	0	0	1	1	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

SRIO_P2MReqTxBACR2 field descriptions

Field	Description
0–3 -	This field is reserved. Reserved
4–7 GenAG[0:3]	Generic arbitration group. Controls the buffer allocation of any arbitration group message unit request packets as follows ... 0000 No generic buffers allocated: No generic buffers allocated for any arbitration group message unit request packets. Note: Use at own risk; this may cause deadlocks. 0001 1 generic buffer allocated: 1 generic buffer allocated for any arbitration group message unit request packets. 000F 15 generic buffers allocated: 15 generic buffers allocated for any arbitration group message unit request packets.
8–31 -	This field is reserved. Reserved

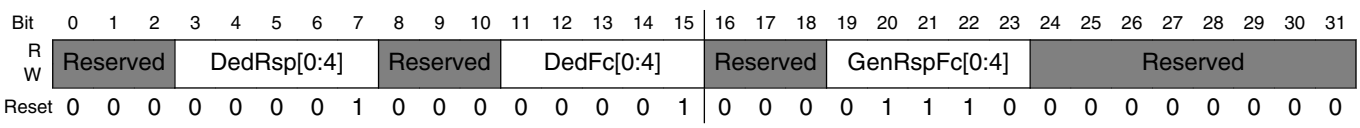
21.5.82 Port 2 Message Response / Flow Control Tx Buffer Allocation Configuration Register (SRIO_P2MRspFcTxBACR)

The port 2 message response/flow control Tx buffer allocation configuration register (PnMRspFcTxBACR), is used to control the buffer allocation of message unit response and flow control packets within the serial RapidIO controller's transmit logic (Tx).

NOTE

There are only 16 buffers that hold message unit response and flow control packets. The total value of the fields within this register, must not exceed 16 (undefined behavior will result). See Buffer Allocation," for details on this register's use.

Address: C_0000h base + 1_01FCh offset = D_01FCh



SRIO_P2MRspFcTxBACR field descriptions

Field	Description
0–2 -	This field is reserved. Reserved
3–7 DedRsp[0:4]	Dedicated response. Controls the buffer allocation of message unit response packets as follows ... 0x00 No dedicated buffers allocated: No dedicated buffers allocated for message unit response packets. Note: Use at own risk; this may cause deadlocks. 0x01 1 dedicated buffer allocated: 1 dedicated buffer allocated for message unit response packets. 0x10 16 dedicated buffers allocated: 16 dedicated buffers allocated for message unit response packets.
8–10 -	This field is reserved. Reserved
11–15 DedFc[0:4]	Dedicated flow control. Controls the buffer allocation of message unit flow control packets as follows ... 0x00 No dedicated buffers allocated: No dedicated buffers allocated for message unit flow control packets. Note: Use at own risk; this may cause deadlocks. 0x01 1 dedicated buffer allocated: 1 dedicated buffer allocated for message unit flow control packets. 0x10 16 dedicated buffers allocated: 16 dedicated buffers allocated for message unit flow control packets.
16–18 -	This field is reserved. Reserved
19–23 GenRspFc[0:4]	Generic response and flow control. Controls the buffer allocation of message unit response and flow control packets as follows ...

Table continues on the next page...

SRIO_P2MRspFcTxBACR field descriptions (continued)

Field	Description
	0x00 No generic buffers allocated: No generic buffers allocated for message unit response and flow control packets. Note: Use at own risk; this may cause deadlocks. 0x01 1 generic buffer allocated: 1 generic buffer allocated for message unit response and flow control packets. 0x10 16 generic buffers allocated: 16 generic buffers allocated for message unit response and flow control packets.
24–31 -	This field is reserved. Reserved

21.5.83 IP Block Revision Register 1 (SRIO_IPBRR1)

IP block revision register 1 is used to track changes and revisions of the RapidIO endpoint.

Address: C_0000h base + 1_0BF8h offset = D_0BF8h

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	IPID																IPMJ								IPMN							
W																																
Reset	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1

SRIO_IPBRR1 field descriptions

Field	Description
0–15 IPID	IP block ID = 0x01C0
16–23 IPMJ	Major revision of the IP block = 0x02
24–31 IPMN	Minor revision of the IP block = 0x01

21.5.84 IP Block Revision Register 2 (SRIO_IPBRR2)

IP block revision register 2 is used to track changes and revisions of the RapidIO endpoint.

Address: C_0000h base + 1_0BFCh offset = D_0BFCh

Bit	0–7	8–15	16–23	24–31
R	Reserved	IPINT	Reserved	IPCFG
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

SRIO_IPBRR2 field descriptions

Field	Description
0–7 -	This field is reserved. Reserved
8–15 IPINT	IP block Integration options = 0x0
16–23 -	This field is reserved. Reserved
24–31 IPCFG	IP block Configuration options = 0x0

21.5.85 Port 1 RapidIO outbound window translation address register n (SRIO_P1ROWTAR0)

Address: C_0000h base + 1_0C00h offset = D_0C00h

Bit	0–1	2–7	8–9	10–15
R W	LTGTID	TREXAD		Reserved
Reset	1 1	1 1 1 1 1 1	1 0	0 0 0 0 0 0

Bit	16–23	24–31
R W	Reserved	
Reset	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0

SRIO_P1ROWTAR0 field descriptions

Field	Description
0–1 LTGTID	LTGTID correspond to bits 6-7 of the target ID for a large transport system. This field is valid only if bit 27 of PEFCAR is set.

Table continues on the next page...

SRIO_P1ROWTAR0 field descriptions (continued)

Field	Description
	Bits 0-5 of the target ID are specified in the window's port *n*RapidIO outbound window translation extended address register.
2–9 TREXAD	Translation extended address. TREXAD[0-7] correspond to the target ID for a small transport system or the least significant byte (bits 8-15) of the target ID for a large transport system.
10–31 -	This field is reserved. Reserved

21.5.86 Port 1 RapidIO outbound window translation extended address register 0 (SRIO_P1ROWTEAR0)

The port 1 RapidIO outbound window translation extended address registers contain bits 0-5 of the target ID for a common transport large system.

Address: C_0000h base + 1_0C04h offset = D_0C04h

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																										LTGTID					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1

SRIO_P1ROWTEAR0 field descriptions

Field	Description
0–25 -	This field is reserved. Reserved
26–31 LTGTID	LTGTID correspond to bits 0-5 of the target ID for a large transport system. This field is valid only if bit 27 of PEFCAR is set. Bits 6-7 of the target ID are specified in the window's port *n*RapidIO outbound window translation address register.

21.5.87 Port 1 RapidIO outbound window attributes register 0 (SRIO_P1ROWAR0)

The port 1 RapidIO outbound window attributes register 0, defines the window size to translate and other attributes for the translations. 1T is the largest window size allowed. For a segmented window, these attributes are used for segment 0. The PCI_Window bit applies for all segments.

Address: C_0000h base + 1_0C10h offset = D_0C10h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	EN	Reserved		CRF	TFLOWLV		PCI	Reserved					RDTYP			
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

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

SRIO_P1ROWAR0 field descriptions

Field	Description
0 EN	This field enables this address translation window. It is set to 1 and is read only for default window 0.
1–2 -	This field is reserved. Reserved
3 CRF	Critical Request Flow. 0 For packets that are destined for the same output port of a swich, the flow associated with this packet shall, if this feature is implemented, not receive precedence over other packets at the same flow level 1 For packets that are destined for the same output port of a swich, the flow associated with this packet shall, if this feature is implemented, receive precedence over other packets at the same flow level but have this bit as a 0.
4–5 TFLOWLV	Transaction flow level. This field must be set to 00 if the PCI bit is set. When bridging from a PCI type interface, set this field to 00 and set the PCI field to 1. When bridging from a RapidIO, this field must always be greater than or equal to the priority of the packet received from the originating RapidIO port or a deadlock can occur. 00 Lowest priority transaction request flow 01 Next highest priority transaction request flow 10 Highest priority level transaction request flow 11 Reserved
6 PCI	PCI_Window. This window follows PCI ordering rules as defined in the RapidIO Inter-operability specification

Table continues on the next page...

SRIO_P1ROWAR0 field descriptions (continued)

Field	Description
	The TFLOWLV field must be set to 00 if this bit is set causing reads to have RapidIO priority 0 and writes to have RapidIO priority 1.
7–11 -	This field is reserved. Reserved
12–15 RDTYP	Transaction type to run on RapidIO interface if access is a read. 0000 Reserved 0001 Reserved 0010 IO_READ_HOME 0011 Reserved 0100 NREAD 0101 Reserved 0110 Reserved 0111 MAINTENANCE read 1000 Reserved 1101 ATOMIC decrement 1110 ATOMIC set 1100 Enhanced read, don't snoop local processor 1111 Enhanced read, snoop local processor
16–19 WRTYP	Transaction type to run on RapidIO interface if access was a write. Write-requiring-response sent from the internal platform must generate a write-requiring-response to RapidIO. Therefore, if an internal platform write-requiring-response request hits a window with WRTYP = SWRITE or NWRITE, the RapidIO endpoint generates a NWRITE_R instead. 0000 Reserved 0001 FLUSH 0010 Reserved 0011 SWRITE 0100 NWRITE 0101 NWRITE_R 0110 Reserved 0111 MAINTENANCE write 1000 Reserved 1111 Reserved 1100 Enhanced Write, don't snoop local processor 1111 Enhanced Write, snoop local processor
20–25 -	This field is reserved. Reserved
26–31 SIZE	Outbound translation window size N which is the encoded $2^{(N+1)}$ bytes window size. The smallest window size is 4 Kbytes. 000000 Reserved 001011 4-Kbyte window size 001100 8-Kbyte window size 011111 4-Gbyte window size 100000 8-Gbyte window size 100001 16-Gbyte window size 100010 32-Gbyte window size

Table continues on the next page...

SRIO_P1ROWAR0 field descriptions (continued)

Field	Description
	100011 64-Gbyte window size 100100 128-Gbyte window size 100101 256-Gbyte window size 100110 512-Gbyte window size 100111 1-Tbyte window size 101000 Reserved 111111 Reserved

21.5.88 Port 1 RapidIO outbound window translation address register n (SRIO_P1ROWTAR*n*)

The port 1 RapidIO outbound window translation address registers select the starting addresses in the external address space for window hits within the outbound translation windows. The new translated address is created by concatenating the transaction offset to this translation address.

Address: C_0000h base + 1_0C20h offset + (32d × i), where i=0d to 7d

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

SRIO_P1ROWTAR*n* field descriptions

Field	Description
0–1 LTGTID	LTGTID correspond to bits 6-7 of the target ID for a large transport system. This field is valid only if bit 27 of PEFCAR is set. Bits 0-5 of the target ID are specified in the window's port *n*RapidIO outbound window translation extended address register.
2–11 TREXAD	Translation extended address. TREXAD[0-7] correspond to the target ID for a small transport system or the least significant byte (bits 8-15) of the target ID for a large transport system. TREXAD[8-9] corresponds to bits [0-1] of a 34-bit RapidIO translation address. For maintenance transactions , TREXAD[8-9] is reserved.
12–31 TRAD	Translation address. System address which represents the starting point of the outbound translated address. The translation address must be aligned based on the size field. This corresponds to bits [2-21] of the 34-bit RapidIO translation address.

Table continues on the next page...

SRIO_P1ROWTAR*n* field descriptions (continued)

Field	Description
	For maintenance transactions, the hop count is formed from TRAD[0-7], and the upper 12 bits of the maintenance offset is formed from TRAD[8-19]; the rest of the maintenance offset is formed from the untranslated address. .

21.5.89 Port 1 RapidIO outbound window translation extended address register n (SRIO_P1ROWTEAR*n*)

The port 1 RapidIO outbound window translation extended address registers contain bits 0-5 of the target ID for a common transport large system.

Address: C_0000h base + 1_0C24h offset + (32d × 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	Reserved																										LTGTID					
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

SRIO_P1ROWTEAR*n* field descriptions

Field	Description
0–25 -	This field is reserved. Reserved
26–31 LTGTID	LTGTID correspond to bits 0-5 of the target ID for a large transport system. This field is valid only if bit 27 of PEFCAR is set. Bits 6-7 of the target ID are specified in the window's port *n*RapidIO outbound window translation address register.

21.5.90 Port 1 RapidIO outbound window base address register n (SRIO_P1ROWBAR*n*)

The port 1 RapidIO outbound window base address registers select the base address for the windows which are translated to an alternate system address space. Addresses for outbound transactions are compared to these windows. If such a transaction does not fall within one of these spaces the transaction is forwarded through the default register set. For transactions that cross more than one window, please see Window Boundary Crossing Errors-Outbound."

Address: C_0000h base + 1_0C28h offset + (32d × 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	Reserved				BEXTADD								BADD																			
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

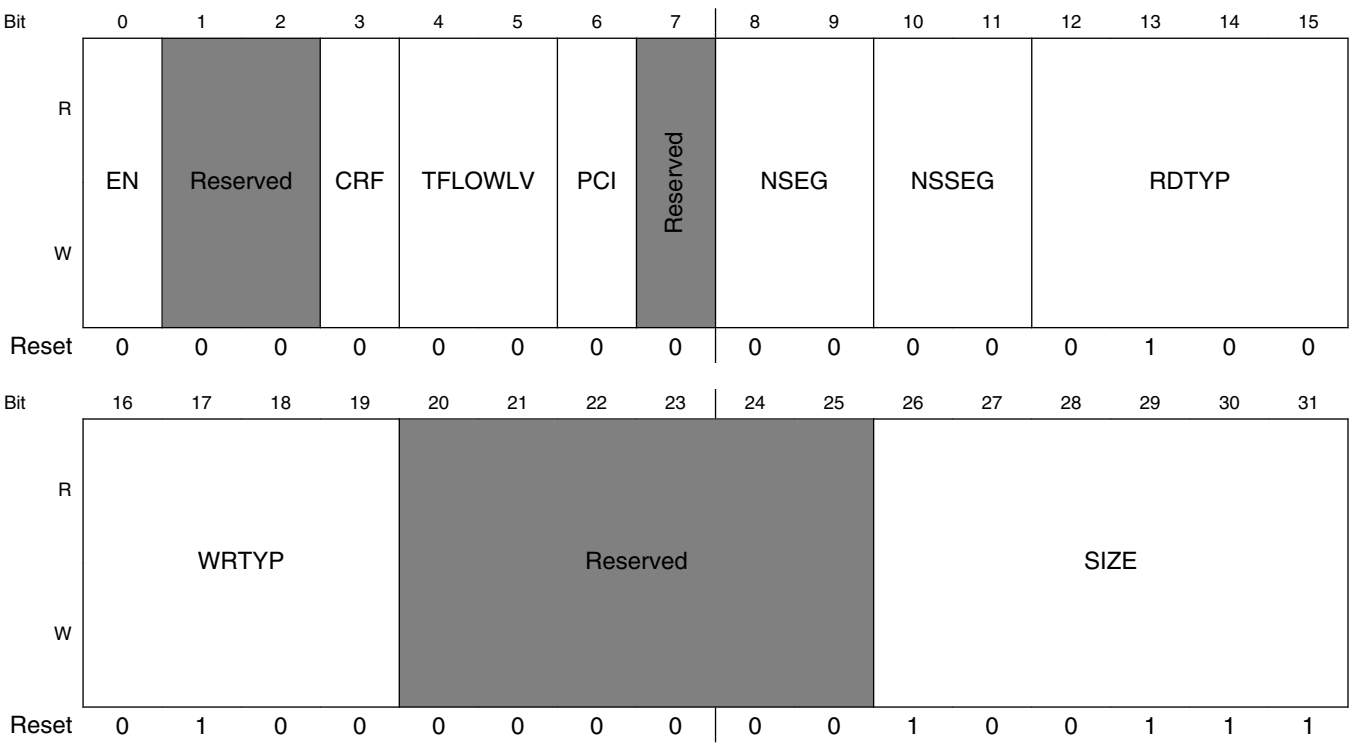
SRIO_P1ROWBAR*n* field descriptions

Field	Description
0–3 -	This field is reserved. Reserved
4–11 BEXTADD	Window base extended address. Corresponds to bits [0-7] of the 40-bit internal platform base address.
12–31 BADD	Window base address. Source address which is the starting point for the outbound translation window. The window must be aligned based on the size selected in the window size bits. This corresponds to bits[8-27] of the 40-bit internal platform base address.

21.5.91 Port 1 RapidIO outbound window attributes register n (SRIO_P1ROWAR*n*)

The port 1 RapidIO outbound window attributes registers, define the window sizes to translate and other attributes for the translations. 1T is the largest window size allowed. For a segmented window, these attributes are used for segment 0. The PCI_Window bit applies for all segments.

Address: C_0000h base + 1_0C30h offset + (32d × i), where i=0d to 7d



SRIO_P1ROWAR*n* field descriptions

Field	Description
0 EN	This field enables this address translation window. It is set to 1 and is read only for default window 0.
1–2 -	This field is reserved. Reserved
3 CRF	Critical Request Flow. 0 For packets that are destined for the same output port of a swich, the flow associated with this packet shall, if this feature is implemented, not receive precedence over other packets at the same flow level 1 For packets that are destined for the same output port of a swich, the flow associated with this packet shall, if this feature is implemented, receive precedence over other packets at the same flow level but have this bit as a 0.

Table continues on the next page...

SRIO_P1ROWAR*n* field descriptions (continued)

Field	Description
4–5 TFLOWLV	Transaction flow level. This field must be set to 00 if the PCI bit is set. When bridging from a PCI type interface, set this field to 00 and set the PCI field to 1. When bridging from a RapidIO, this field must always be greater than or equal to the priority of the packet received from the originating RapidIO port or a deadlock can occur. 00 Lowest priority transaction request flow 01 Next highest priority transaction request flow 10 Highest priority level transaction request flow 11 Reserved
6 PCI	PCI_Window. This window follows PCI ordering rules as defined in the RapidIO Inter-operability specification The TFLOWLV field must be set to 00 if this bit is set causing reads to have RapidIO priority 0 and writes to have RapidIO priority 1.
7 -	This field is reserved. Reserved
8–9 NSEG	Number of segments for this window. 00 One segment (normal window) 01 Two segments (half size aliasing window) 10 Four segments (quarter size aliasing window) 11 Reserved
10–11 NSSEG	Number of subsegments for this segment. Note that this field is valid only when ROWARn[NSEG] contains a non-zero value (1 or 2). 00 One target deviceID for this segment 01 Two target deviceIDs for this segment 10 Four target deviceIDs for this segment 11 Eight target deviceIDs for this segment
12–15 RDTYP	Transaction type to run on RapidIO interface if access is a read. 0000 Reserved 0001 Reserved 0010 IO_READ_HOME 0011 Reserved 0100 NREAD 0101 Reserved 0110 Reserved 0111 MAINTENANCE read 1000 Reserved 1100 ATOMIC increment 1101 ATOMIC decrement 1110 ATOMIC set 1111 ATOMIC clear 1100 Enhanced read, don’t snoop local processor 1111 Enhanced read, snoop local processor

Table continues on the next page...

SRIO_P1ROWAR*n* field descriptions (continued)

Field	Description
16–19 WRTYP	Transaction type to run on RapidIO interface if access was a write. Write-requiring-response sent from the internal platform must generate a write-requiring-response to RapidIO. Therefore, if an internal platform write-requiring-response request hits a window with WRTYP = SWRITE or NWRITE, the RapidIO endpoint generates a NWRITE_R instead. 0000 Reserved 0001 FLUSH 0010 Reserved 0011 SWRITE 0100 NWRITE 0101 NWRITE_R 0110 Reserved 0111 MAINTENANCE write 1000 Reserved 1100 Enhanced Write, don't snoop local processor 1111 Enhanced Write, snoop local processor
20–25 -	This field is reserved. Reserved
26–31 SIZE	Outbound translation window size N which is the encoded $2^{(N+1)}$ bytes window size. The smallest window size is 4 Kbytes. 000000 Reserved 001011 4-Kbyte window size 001100 8-Kbyte window size 011111 4-Gbyte window size 100000 8-Gbyte window size 100001 16-Gbyte window size 100010 32-Gbyte window size 100011 64-Gbyte window size 100100 128-Gbyte window size 100101 256-Gbyte window size 100110 512-Gbyte window size 100111 1-Tbyte window size 101000 Reserved 111111 Reserved

21.5.92 Port 1 RapidIO outbound window segment 1 register n (SRIO_P1ROWS1R*n*)

The port 1 RapidIO outbound window segment registers define the attributes and target device ID that are to be used for a transaction that hits in that segment rather than the primary attributes and target deviceID. There is one of these registers for each segment (except segment 0).

Address: C_0000h base + 1_0C34h offset + (32d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved			CRF	TFLOWLV		Reserved		RDTYP				WRTYP			
W																
Reset	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved								SGTGTDID1					SGTGTDID2	SGTGTDID3	SGTGTDID4
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRIO_P1ROWS1R*n* field descriptions

Field	Description
0–2 -	This field is reserved. Reserved
3 CRF	Critical Request Flow. 0 For packets that are destined for the same output port of a swich, the flow associated with this packet shall, if this feature is implemented, not receive precedence over other packets at the same flow level. 1 For packets that are destined for the same output port of a swich, the flow associated with this packet shall, if this feature is implemented, receive precedence over other packets at the same flow level but have this bit as a 0.
4–5 TFLOWLV	Transaction flow level. This field must be set to 00 if the PCI_Window bit is set 00 Lowest priority transaction request flow 01 Next highest priority transaction request flow 10 Highest priority level transaction request flow 11 Reserved
6–7 -	This field is reserved. Reserved

Table continues on the next page...

SRIO_P1ROWS1R*n* field descriptions (continued)

Field	Description
8–11 RDTYP	Transaction type to run on RapidIO interface if access was a read. 0000 Reserved 0001 Reserved 0010 IO_READ_HOME 0011 Reserved 0100 NREAD 0101 Reserved 0110 Reserved 0111 MAINTENANCE read 1000 Reserved 1100 ATOMIC increment 1101 ATOMIC decrement 1110 ATOMIC set 1111 ATOMIC clear
12–15 WRTYP	Transaction type to run on RapidIO interface if access was a write. Writes-requiring-response sent from the internal platform must generate a write-requiring-response to RapidIO. Therefore, if an internal platform write-requiring-response request hits a window with WRTYP = SWRITE or NWRITE, the RapidIO endpoint generates a NWRITE_R instead. 0000 Reserved 0001 FLUSH 0010 Reserved 0011 SWRITE 0100 NWRITE 0101 NWRITE_R 0110 Reserved 0111 MAINTENANCE write 1000 Reserved 1110 Reserved 1111 Reserved
16–23 -	This field is reserved. Reserved
24–28 SGTGTDID1	Bits 0-4 (or bits 8-12 if large transport system) of the target device ID for this segment.
29 SGTGTDID2	Bit 5 (or bit 13 if large transport system) of the target deviceID; this bit is reserved if 8 target subsegments are selected.
30 SGTGTDID3	Bit 6 (or bit 14 if large transport system) of the target deviceID; this bit is reserved if 8 or 4 target subsegments are selected.
31 SGTGTDID4	Bit 7 (or bit 15 if large transport system) of the target deviceID; this bit is reserved if 8, 4, or 2 target subsegments are selected.

21.5.93 Port 1 RapidIO outbound window segment 2 register n (SRIO_P1ROWS2R*n*)

The port 1 RapidIO outbound window segment registers define the attributes and target device ID that are to be used for a transaction that hits in that segment rather than the primary attributes and target deviceID. There is one of these registers for each segment (except segment 0).

Address: C_0000h base + 1_0C38h offset + (32d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved			CRF	TFLOWLV		Reserved		RDTYP				WRTYP			
W																
Reset	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved								SGTGTDID1					SGTGTDID2	SGTGTDID3	SGTGTDID4
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRIO_P1ROWS2R*n* field descriptions

Field	Description
0–2 -	This field is reserved. Reserved
3 CRF	Critical Request Flow. 0 For packets that are destined for the same output port of a swich, the flow associated with this packet shall, if this feature is implemented, not receive precedence over other packets at the same flow level. 1 For packets that are destined for the same output port of a swich, the flow associated with this packet shall, if this feature is implemented, receive precedence over other packets at the same flow level but have this bit as a 0.
4–5 TFLOWLV	Transaction flow level. This field must be set to 00 if the PCI_Window bit is set 00 Lowest priority transaction request flow 01 Next highest priority transaction request flow 10 Highest priority level transaction request flow 11 Reserved
6–7 -	This field is reserved. Reserved

Table continues on the next page...

SRIO_P1ROWS2R*n* field descriptions (continued)

Field	Description
8–11 RDTYP	Transaction type to run on RapidIO interface if access was a read. 0000 Reserved 0001 Reserved 0010 IO_READ_HOME 0011 Reserved 0100 NREAD 0101 Reserved 0110 Reserved 0111 MAINTENANCE read 1000 Reserved 1100 ATOMIC increment 1101 ATOMIC decrement 1110 ATOMIC set 1111 ATOMIC clear
12–15 WRTYP	Transaction type to run on RapidIO interface if access was a write. Writes-requiring-response sent from the internal platform must generate a write-requiring-response to RapidIO. Therefore, if an internal platform write-requiring-response request hits a window with WRTYP = SWRITE or NWRITE, the RapidIO endpoint generates a NWRITE_R instead. 0000 Reserved 0001 FLUSH 0010 Reserved 0011 SWRITE 0100 NWRITE 0101 NWRITE_R 0110 Reserved 0111 MAINTENANCE write 1000 Reserved 1110 Reserved 1111 Reserved
16–23 -	This field is reserved. Reserved
24–28 SGTGTDID1	Bits 0-4 (or bits 8-12 if large transport system) of the target device ID for this segment.
29 SGTGTDID2	Bit 5 (or bit 13 if large transport system) of the target deviceID; this bit is reserved if 8 target subsegments are selected.
30 SGTGTDID3	Bit 6 (or bit 14 if large transport system) of the target deviceID; this bit is reserved if 8 or 4 target subsegments are selected.
31 SGTGTDID4	Bit 7 (or bit 15 if large transport system) of the target deviceID; this bit is reserved if 8, 4, or 2 target subsegments are selected.

21.5.94 Port 1 RapidIO outbound window segment 3 register n (SRIO_P1ROWS3R*n*)

The port 1 RapidIO outbound window segment registers define the attributes and target device ID that are to be used for a transaction that hits in that segment rather than the primary attributes and target deviceID. There is one of these registers for each segment (except segment 0).

Address: C_0000h base + 1_0C3Ch offset + (32d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Reserved			CRF	TFLOWLV		Reserved		RDTYP				WRTYP			
Reset	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Reserved								SGTGTDID1					SGTGTDID2	SGTGTDID3	SGTGTDID4
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRIO_P1ROWS3R*n* field descriptions

Field	Description
0–2 -	This field is reserved. Reserved
3 CRF	Critical Request Flow. 0 For packets that are destined for the same output port of a swich, the flow associated with this packet shall, if this feature is implemented, not receive precedence over other packets at the same flow level. 1 For packets that are destined for the same output port of a swich, the flow associated with this packet shall, if this feature is implemented, receive precedence over other packets at the same flow level but have this bit as a 0.
4–5 TFLOWLV	Transaction flow level. This field must be set to 00 if the PCI_Window bit is set 00 Lowest priority transaction request flow 01 Next highest priority transaction request flow 10 Highest priority level transaction request flow 11 Reserved
6–7 -	This field is reserved. Reserved

Table continues on the next page...

SRIO_P1ROWS3R*n* field descriptions (continued)

Field	Description
8–11 RDTYP	Transaction type to run on RapidIO interface if access was a read. 0000 Reserved 0001 Reserved 0010 IO_READ_HOME 0011 Reserved 0100 NREAD 0101 Reserved 0110 Reserved 0111 MAINTENANCE read 1000 Reserved 1100 ATOMIC increment 1101 ATOMIC decrement 1110 ATOMIC set 1111 ATOMIC clear
12–15 WRTYP	Transaction type to run on RapidIO interface if access was a write. Writes-requiring-response sent from the internal platform must generate a write-requiring-response to RapidIO. Therefore, if an internal platform write-requiring-response request hits a window with WRTYP = SWRITE or NWRITE, the RapidIO endpoint generates a NWRITE_R instead. 0000 Reserved 0001 FLUSH 0010 Reserved 0011 SWRITE 0100 NWRITE 0101 NWRITE_R 0110 Reserved 0111 MAINTENANCE write 1000 Reserved 1110 Reserved 1111 Reserved
16–23 -	This field is reserved. Reserved
24–28 SGTGTDID1	Bits 0-4 (or bits 8-12 if large transport system) of the target device ID for this segment.
29 SGTGTDID2	Bit 5 (or bit 13 if large transport system) of the target deviceID; this bit is reserved if 8 target subsegments are selected.
30 SGTGTDID3	Bit 6 (or bit 14 if large transport system) of the target deviceID; this bit is reserved if 8 or 4 target subsegments are selected.
31 SGTGTDID4	Bit 7 (or bit 15 if large transport system) of the target deviceID; this bit is reserved if 8, 4, or 2 target subsegments are selected.

21.5.95 Port 1 RapidIO Inbound window translation address register n (SRIO_P1RIWTAR*n*)

The port 1 RapidIO inbound window translation address registers point to the starting addresses in local address space for window hits within the inbound translation windows. The new translated address is created by concatenating the transaction offset to this translation address.

Address: C_0000h base + 1_0D60h offset + (32d × i), where i=0d to 3d

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				TREXAD								TRAD																			
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

SRIO_P1RIWTAR*n* field descriptions

Field	Description
0–3 -	This field is reserved. Reserved
4–11 TREXAD	Translation extended address. Corresponds to bits 0-7 of the 40-bit internal platform translation address. TREXAD[6-7] are reserved for default window 0.
12–31 TRAD	Translation address. Target address which indicates the starting point of the inbound translated address. The translation address must be aligned based on the size field. This corresponds to bits 8-27 of the 40-bit internal platform translation address. TRAD is reserved for default window 0.

21.5.96 Port 1 RapidIO Inbound window base address register n (SRIO_P1RIWBAR*n*)

The port 1 RapidIO inbound window *n* base address registers select the base address for the windows which are translated to an alternate target address space. Addresses for inbound transactions are compared to these windows. If such a transaction does not fall within one of these spaces, then the transaction is forwarded to the interior of the chip using the default window. For transactions that cross more than one window, please see Window Boundary Crossing Errors-Inbound. Note that the LCSBA1CSR register Local configuration space base address 1 command and status register (SRIO_LCSBA1CSR) has priority over all ATMU windows if both are configured for the same address space.

Address: C_0000h base + 1_0D68h offset + (32d × i), where i=0d to 3d

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

SRIO_P1RIWBAR*n* field descriptions

Field	Description
0–9 -	This field is reserved. Reserved
10–11 BEXAD	Base extended address. BEXAD represents bits 0-1 of the 34-bit RapidIO address.
12–31 BADD	Base address. System address which is the starting point for the inbound translation window. The window must be aligned based on the size selected in the window size bits. This corresponds to bits 2-21 of the 34-bit RapidIO base address.

21.5.97 Port 1 RapidIO inbound window attributes register n (SRIO_P1RIWAR*n*)

The port 1 RapidIO inbound window attributes registers define the window sizes to translate and other attributes for the translations. 16 Gbytes is the largest window size allowed. The RDTYP and WRTYP fields are used for attributes on the request, but do not actually change the transaction type of the transaction. In other words, PnRIWARn does not modify whether or not the request requires a response or if the request was atomic; this type of attribute remains unchanged on the request as it is translated through the ATMU.

Address: C_0000h base + 1_0D70h offset + (32d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	EN	PW	Reserved						TGINT				RDTYP			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

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

SRIO_P1RIWAR*n* field descriptions

Field	Description
0 EN	This bit enables this address translation. This field is set to 1 and read-only for default window 0.
1 PW	Protected Window. This bit indicates that this window is protected. Writes requiring response and reads to this window generate an error response. Writes not requiring response are silently discarded.
2–7 -	This field is reserved. Reserved
8–11 TGINT	Target interface. If this field is set to anything other than local address space, the attributes for the transaction must be assigned in a corresponding outbound window at the target. 0001 PCI Express 1 0011 PCI Express 2 0010 PCI Express 3 0101 PCI Express 4 0001 PCI Express 1 1000 Serial RapidIO port 1 1001 Serial RapidIO port 2
12–15 RDTYP	Transaction type to run on the I/O interface if access is a read. 0000 Reserved ... 0100 Read

Table continues on the next page...

SRIO_P1RIWAR*n* field descriptions (continued)

Field	Description
	0101 Reserved ... 1111 Reserved Transaction type to run on local memory if access is a read. 0000 Reserved ... 0100 Read, don't snoop local processor 0101 Read, snoop local processor 0110 Reserved 0111 Reserved ... 1100 Enhanced read, don't snoop local processor 1101 Enhanced read, snoop local processor ... 1111 Reserved
16–19 WRTYP	Transaction type to run on I/O interface if access is a write. 0000 Reserved ... 0100 Write 0101 Reserved ... 1111 Reserved Transaction type to run on local memory if access is a write. 0000 Reserved ... 0011 Reserved 0100 Write, don't snoop local processor 0101 Write, snoop local processor 0110 Reserved ... 1100 Write, don't snoop local processor 1101 Write, snoop local processor ... 1111 Reserved
20–25 -	This field is reserved. Reserved
26–31 SIZE	Inbound translation window size N which is the encoded 2^(N+1) bytes window size. The smallest window size is 4 Kbytes.

Table continues on the next page...

SRIO_P1RIWAR*n* field descriptions (continued)

Field	Description
	This field is read only for default window 0. 000000 Reserved 001011 4-Kbyte window size 001100 8-Kbyte window size 011111 4-Gbyte window size 100000 8-Gbyte window size 100001 16-Gbyte window size 100010 Reserved 111111 Reserved

21.5.98 Port 1 RapidIO inbound window translation address register 0 (SRIO_P1RIWTAR0)

The port 1 RapidIO inbound window translation address registers point to the starting addresses in local address space for window hits within the inbound translation windows. The new translated address is created by concatenating the transaction offset to this translation address.

Address: C_0000h base + 1_0DE0h offset = D_0DE0h

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				TREXAD						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

SRIO_P1RIWTAR0 field descriptions

Field	Description
0–3 -	This field is reserved. Reserved
4–9 TREXAD	Translation extended address. Corresponds to bits 0-5 of the 40-bit internal platform translation address.
10–31 -	This field is reserved. Reserved

21.5.99 Port 1 RapidIO inbound window attributes register 0 (SRIO_P1RIWAR0)

The port 1 RapidIO inbound window attributes registers define the window sizes to translate and other attributes for the translations. 16 Gbytes is the largest window size allowed. The RDTYP and WRTYP fields are used for attributes on the request, but do not actually change the transaction type of the transaction. In other words, PnRIWARn does not modify whether or not the request requires a response or if the request was atomic; this type of attribute remains unchanged on the request as it is translated through the ATMU.

Address: C_0000h base + 1_0DF0h offset = D_0DF0h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EN	PW	Reserved						TGINT				RDTYP			
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

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

SRIO_P1RIWAR0 field descriptions

Field	Description
0 EN	This bit enables this address translation. This field is set to 1 and read-only for default window 0.
1 PW	Protected Window. This bit indicates that this window is protected. Writes requiring response and reads to this window generate an error response. Writes not requiring response are silently discarded.
2–7 -	This field is reserved. Reserved
8–11 TGINT	Target interface. If this field is set to anything other than local address space, the attributes for the transaction must be assigned in a corresponding outbound window at the target. 0001 PCI Express 1 0011 PCI Express 2 0010 PCI Express 3 0101 PCI Express 4 0001 PCI Express 1 1000 Serial RapidIO port 1 1001 Serial RapidIO port 2
12–15 RDTYP	Transaction type to run on the I/O interface if access is a read.

Table continues on the next page...

SRIO_P1RIWAR0 field descriptions (continued)

Field	Description
	0000 Reserved ... 0100 Read 0101 Reserved ... 1111 Reserved Transaction type to run on local memory if access is a read. 0000 Reserved ... 0100 Read, don't snoop local processor 0101 Read, snoop local processor 0110 Reserved 0111 Reserved ... 1100 Enhanced read, don't snoop local processor 1101 Enhanced read, snoop local processor ... 1111 Reserved
16–19 WRTYP	Transaction type to run on I/O interface if access is a write. 0000 Reserved ... 0100 Write 0101 Reserved ... 1111 Reserved Transaction type to run on local memory if access is a write. 0000 Reserved ... 0011 Reserved 0100 Write, don't snoop local processor 0101 Write, snoop local processor 0110 Reserved ... 1100 Write, don't snoop local processor 1101 Write, snoop local processor ... 1111 Reserved

Table continues on the next page...

SRIO_P1RIWAR0 field descriptions (continued)

Field	Description
20–25 -	This field is reserved. Reserved
26–31 SIZE	Inbound translation window size N which is the encoded 2^(N+1) bytes window size. The smallest window size is 4 Kbytes. This field is read only for default window 0. 000000 Reserved 001011 4-Kbyte window size 001100 8-Kbyte window size 011111 4-Gbyte window size 100000 8-Gbyte window size 100001 16-Gbyte window size 100010 Reserved 111111 Reserved

21.5.100 Port 2 RapidIO outbound window translation address register n (SRIO_P2ROWTAR0)

Address: C_0000h base + 1_0E00h offset = D_0E00h

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

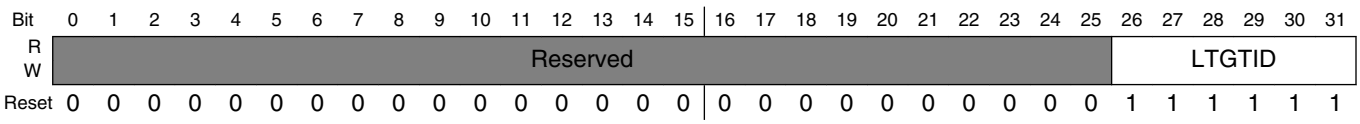
SRIO_P2ROWTAR0 field descriptions

Field	Description
0–1 LTGTID	LTGTID correspond to bits 6-7 of the target ID for a large transport system. This field is valid only if bit 27 of PEFCAR is set. Bits 0-5 of the target ID are specified in the window's port *n*RapidIO outbound window translation extended address register.
2–9 TREXAD	Translation extended address. TREXAD[0-7] correspond to the target ID for a small transport system or the least significant byte (bits 8-15) of the target ID for a large transport system.
10–31 -	This field is reserved. Reserved

21.5.101 Port 2 RapidIO outbound window translation extended address register 0 (SRIO_P2ROWTEAR0)

The port 2 RapidIO outbound window translation extended address registers contain bits 0-5 of the target ID for a common transport large system.

Address: C_0000h base + 1_0E04h offset = D_0E04h



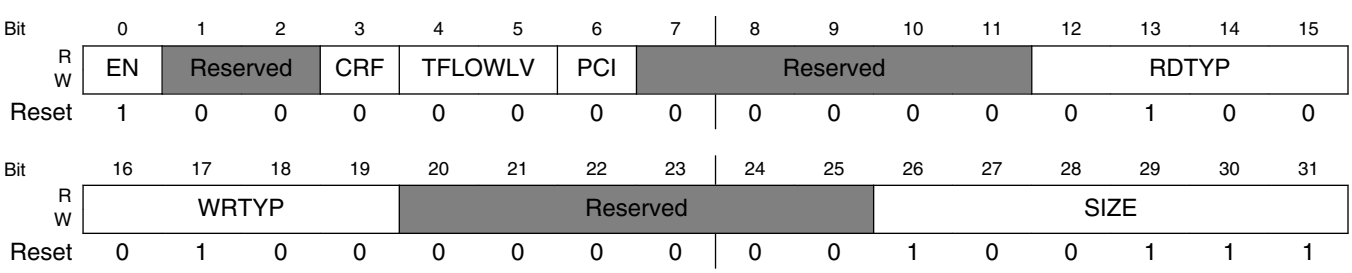
SRIO_P2ROWTEAR0 field descriptions

Field	Description
0–25 -	This field is reserved. Reserved
26–31 LTGTID	LTGTID correspond to bits 0-5 of the target ID for a large transport system. This field is valid only if bit 27 of PEFCAR is set. Bits 6-7 of the target ID are specified in the window's port *n*RapidIO outbound window translation address register.

21.5.102 Port 2 RapidIO outbound window attributes register 0 (SRIO_P2ROWAR0)

The port 2 RapidIO outbound window attributes register 0, defines the window size to translate and other attributes for the translations. 1T is the largest window size allowed. For a segmented window, these attributes are used for segment 0. The PCI_Window bit applies for all segments.

Address: C_0000h base + 1_0E10h offset = D_0E10h



SRIO_P2ROWAR0 field descriptions

Field	Description
0 EN	This field enables this address translation window. It is set to 1 and is read only for default window 0.
1–2 -	This field is reserved. Reserved
3 CRF	Critical Request Flow. 0 For packets that are destined for the same output port of a swich, the flow associated with this packet shall, if this feature is implemented, not receive precedence over other packets at the same flow level 1 For packets that are destined for the same output port of a swich, the flow associated with this packet shall, if this feature is implemented, receive precedence over other packets at the same flow level but have this bit as a 0.
4–5 TFLOWLV	Transaction flow level. This field must be set to 00 if the PCI bit is set. When bridging from a PCI type interface, set this field to 00 and set the PCI field to 1. When bridging from a RapidIO, this field must always be greater than or equal to the priority of the packet received from the originating RapidIO port or a deadlock can occur. 00 Lowest priority transaction request flow 01 Next highest priority transaction request flow 10 Highest priority level transaction request flow 11 Reserved
6 PCI	PCI_Window. This window follows PCI ordering rules as defined in the RapidIO Inter-operability specification The TFLOWLV field must be set to 00 if this bit is set causing reads to have RapidIO priority 0 and writes to have RapidIO priority 1.
7–11 -	This field is reserved. Reserved
12–15 RDTYP	Transaction type to run on RapidIO interface if access is a read. 0000 Reserved 0001 Reserved 0010 IO_READ_HOME 0011 Reserved 0100 NREAD 0101 Reserved 0110 Reserved 0111 MAINTENANCE read 1000 Reserved 1100 ATOMIC increment 1101 ATOMIC decrement 1110 ATOMIC set 1111 ATOMIC clear
16–19 WRTYP	Transaction type to run on RapidIO interface if access was a write. Write-requiring-response sent from the internal platform must generate a write-requiring-response to RapidIO. Therefore, if an internal platform write-requiring-response request hits a window with WRTYP = SWRITE or NWRITE, the RapidIO endpoint generates a NWRITE_R instead.

Table continues on the next page...

SRIO_P2ROWAR0 field descriptions (continued)

Field	Description
	0000 Reserved 0001 FLUSH 0010 Reserved 0011 SWRITE 0100 NWRITE 0101 NWRITE_R 0110 Reserved 0111 MAINTENANCE write 1000 Reserved 1111 Reserved
20–25 -	This field is reserved. Reserved
26–31 SIZE	Outbound translation window size N which is the encoded $2^{(N+1)}$ bytes window size. The smallest window size is 4 Kbytes. 000000 Reserved 001011 4-Kbyte window size 001100 8-Kbyte window size 011111 4-Gbyte window size 100000 8-Gbyte window size 100001 16-Gbyte window size 100010 32-Gbyte window size 100011 64-Gbyte window size 100100 128-Gbyte window size 100101 256-Gbyte window size 100110 512-Gbyte window size 100111 1-Tbyte window size 101000 Reserved 111111 Reserved

21.5.103 Port 2 RapidIO outbound window translation address register n (SRIO_P2ROWTAR*n*)

The port 2 RapidIO outbound window translation address registers select the starting addresses in the external address space for window hits within the outbound translation windows. The new translated address is created by concatenating the transaction offset to this translation address.

Address: C_0000h base + 1_0E20h offset + (32d × i), where i=0d to 7d

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

SRIO_P2ROWTAR*n* field descriptions

Field	Description
0–1 LTGTID	LTGTID correspond to bits 6-7 of the target ID for a large transport system. This field is valid only if bit 27 of PEFCAR is set. Bits 0-5 of the target ID are specified in the window's port *n*RapidIO outbound window translation extended address register.
2–11 TREXAD	Translation extended address. TREXAD[0-7] correspond to the target ID for a small transport system or the least significant byte (bits 8-15) of the target ID for a large transport system. TREXAD[8-9] corresponds to bits [0-1] of a 34-bit RapidIO translation address. For maintenance transactions , TREXAD[8-9] is reserved.
12–31 TRAD	Translation address. System address which represents the starting point of the outbound translated address. The translation address must be aligned based on the size field. This corresponds to bits [2-21] of the 34-bit RapidIO translation address. For maintenance transactions, the hop count is formed from TRAD[0-7], and the upper 12 bits of the maintenance offset is formed from TRAD[8-19]; the rest of the maintenance offset is formed from the untranslated address. .

21.5.104 Port 2 RapidIO outbound window translation extended address register n (SRIO_P2ROWTEAR*n*)

The port 2 RapidIO outbound window translation extended address registers contain bits 0-5 of the target ID for a common transport large system.

Address: C_0000h base + 1_0E24h offset + (32d × 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	Reserved																										LTGTID					
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

SRIO_P2ROWTEAR*n* field descriptions

Field	Description
0–25 -	This field is reserved. Reserved
26–31 LTGTID	LTGTID correspond to bits 0-5 of the target ID for a large transport system. This field is valid only if bit 27 of PEFCAR is set. Bits 6-7 of the target ID are specified in the window's port *n*RapidIO outbound window translation address register.

21.5.105 Port 2 RapidIO outbound window base address register n (SRIO_P2ROWBAR*n*)

The port 2 RapidIO outbound window base address registers select the base address for the windows which are translated to an alternate system address space. Addresses for outbound transactions are compared to these windows. If such a transaction does not fall within one of these spaces the transaction is forwarded through the default register set. For transactions that cross more than one window, please see Window Boundary Crossing Errors-Outbound."

Address: C_0000h base + 1_0E28h offset + (32d × 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	Reserved				BEXTADD								BADD																			
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

SRIO_P2ROWBAR*n* field descriptions

Field	Description
0–3 -	This field is reserved. Reserved
4–11 BEXTADD	Window base extended address. Corresponds to bits [0-7] of the 40-bit internal platform base address.
12–31 BADD	Window base address. Source address which is the starting point for the outbound translation window. The window must be aligned based on the size selected in the window size bits. This corresponds to bits[8-27] of the 40-bit internal platform base address.

21.5.106 Port 2 RapidIO outbound window attributes register n (SRIO_P2ROWAR*n*)

The port 2 RapidIO outbound window attributes registers, define the window sizes to translate and other attributes for the translations. 1T is the largest window size allowed. For a segmented window, these attributes are used for segment 0. The PCI_Window bit applies for all segments.

Address: C_0000h base + 1_0E30h offset + (32d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	EN	Reserved	CRF	TFLOWLV			PCI	Reserved	NSEG		NSSEG		RDTYP			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

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

SRIO_P2ROWAR*n* field descriptions

Field	Description
0 EN	This field enables this address translation window. It is set to 1 and is read only for default window 0.
1–2 -	This field is reserved. Reserved
3 CRF	Critical Request Flow. 0 For packets that are destined for the same output port of a swich, the flow associated with this packet shall, if this feature is implemented, not receive precedence over other packets at the same flow level 1 For packets that are destined for the same output port of a swich, the flow associated with this packet shall, if this feature is implemented, receive precedence over other packets at the same flow level but have this bit as a 0.
4–5 TFLOWLV	Transaction flow level. This field must be set to 00 if the PCI bit is set. When bridging from a PCI type interface, set this field to 00 and set the PCI field to 1. When bridging from a RapidIO, this field must always be greater than or equal to the priority of the packet received from the originating RapidIO port or a deadlock can occur. 00 Lowest priority transaction request flow 01 Next highest priority transaction request flow 10 Highest priority level transaction request flow 11 Reserved
6 PCI	PCI_Window. This window follows PCI ordering rules as defined in the RapidIO Inter-operability specification The TFLOWLV field must be set to 00 if this bit is set causing reads to have RapidIO priority 0 and writes to have RapidIO priority 1.
7 -	This field is reserved. Reserved
8–9 NSEG	Number of segments for this window. 00 One segment (normal window) 01 Two segments (half size aliasing window) 10 Four segments (quarter size aliasing window) 11 Reserved
10–11 NSSEG	Number of subsegments for this segment. Note that this field is valid only when ROWARn[NSEG] contains a non-zero value (1 or 2). 00 One target deviceID for this segment 01 Two target deviceIDs for this segment 10 Four target deviceIDs for this segment 11 Eight target deviceIDs for this segment
12–15 RDTYP	Transaction type to run on RapidIO interface if access is a read. 0000 Reserved 0001 Reserved 0010 IO_READ_HOME 0011 Reserved 0100 NREAD

Table continues on the next page...

SRIO_P2ROWAR*n* field descriptions (continued)

Field	Description
	0101 Reserved 0110 Reserved 0111 MAINTENANCE read 1000 Reserved 1100 ATOMIC increment 1101 ATOMIC decrement 1110 ATOMIC set 1111 ATOMIC clear
16–19 WRTYP	Transaction type to run on RapidIO interface if access was a write. Write-requiring-response sent from the internal platform must generate a write-requiring-response to RapidIO. Therefore, if an internal platform write-requiring-response request hits a window with WRTYP = SWRITE or NWRITE, the RapidIO endpoint generates a NWRITE_R instead. 0000 Reserved 0001 FLUSH 0010 Reserved 0011 SWRITE 0100 NWRITE 0101 NWRITE_R 0110 Reserved 0111 MAINTENANCE write 1000 Reserved 1111 Reserved
20–25 -	This field is reserved. Reserved
26–31 SIZE	Outbound translation window size N which is the encoded $2^{(N+1)}$ bytes window size. The smallest window size is 4 Kbytes. 000000 Reserved 001011 4-Kbyte window size 001100 8-Kbyte window size 011111 4-Gbyte window size 100000 8-Gbyte window size 100001 16-Gbyte window size 100010 32-Gbyte window size 100011 64-Gbyte window size 100100 128-Gbyte window size 100101 256-Gbyte window size 100110 512-Gbyte window size 100111 1-Tbyte window size 101000 Reserved 111111 Reserved

21.5.107 Port 2 RapidIO outbound window segment 1 register n (SRIO_P2ROWS1R*n*)

The port 2 RapidIO outbound window segment registers define the attributes and target device ID that are to be used for a transaction that hits in that segment rather than the primary attributes and target deviceID. There is one of these registers for each segment (except segment 0).

Address: C_0000h base + 1_0E34h offset + (32d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved			CRF	TFLOWLV		Reserved		RDTYP				WRTYP			
W																
Reset	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved								SGTGTDID1					SGTGTDID2	SGTGTDID3	SGTGTDID4
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRIO_P2ROWS1R*n* field descriptions

Field	Description
0–2 -	This field is reserved. Reserved
3 CRF	Critical Request Flow. 0 For packets that are destined for the same output port of a swich, the flow associated with this packet shall, if this feature is implemented, not receive precedence over other packets at the same flow level. 1 For packets that are destined for the same output port of a swich, the flow associated with this packet shall, if this feature is implemented, receive precedence over other packets at the same flow level but have this bit as a 0.
4–5 TFLOWLV	Transaction flow level. This field must be set to 00 if the PCI_Window bit is set 00 Lowest priority transaction request flow 01 Next highest priority transaction request flow 10 Highest priority level transaction request flow 11 Reserved
6–7 -	This field is reserved. Reserved

Table continues on the next page...

SRIO_P2ROWS1R*n* field descriptions (continued)

Field	Description
8–11 RDTYP	Transaction type to run on RapidIO interface if access was a read. 0000 Reserved 0001 Reserved 0010 IO_READ_HOME 0011 Reserved 0100 NREAD 0101 Reserved 0110 Reserved 0111 MAINTENANCE read 1000 Reserved 1100 ATOMIC increment 1101 ATOMIC decrement 1110 ATOMIC set 1111 ATOMIC clear
12–15 WRTYP	Transaction type to run on RapidIO interface if access was a write. Writes-requiring-response sent from the internal platform must generate a write-requiring-response to RapidIO. Therefore, if an internal platform write-requiring-response request hits a window with WRTYP = SWRITE or NWRITE, the RapidIO endpoint generates a NWRITE_R instead. 0000 Reserved 0001 FLUSH 0010 Reserved 0011 SWRITE 0100 NWRITE 0101 NWRITE_R 0110 Reserved 0111 MAINTENANCE write 1000 Reserved 1110 Reserved 1111 Reserved
16–23 -	This field is reserved. Reserved
24–28 SGTGTDID1	Bits 0-4 (or bits 8-12 if large transport system) of the target device ID for this segment.
29 SGTGTDID2	Bit 5 (or bit 13 if large transport system) of the target deviceID; this bit is reserved if 8 target subsegments are selected.
30 SGTGTDID3	Bit 6 (or bit 14 if large transport system) of the target deviceID; this bit is reserved if 8 or 4 target subsegments are selected.
31 SGTGTDID4	Bit 7 (or bit 15 if large transport system) of the target deviceID; this bit is reserved if 8, 4, or 2 target subsegments are selected.

21.5.108 Port 2 RapidIO outbound window segment 2 register n (SRIO_P2ROWS2R*n*)

The port 2 RapidIO outbound window segment registers define the attributes and target device ID that are to be used for a transaction that hits in that segment rather than the primary attributes and target deviceID. There is one of these registers for each segment (except segment 0).

Address: C_0000h base + 1_0E38h offset + (32d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved			CRF	TFLOWLV		Reserved		RDTYP				WRTYP			
W																
Reset	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved								SGTGTDID1					SGTGTDID2	SGTGTDID3	SGTGTDID4
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRIO_P2ROWS2R*n* field descriptions

Field	Description
0–2 -	This field is reserved. Reserved
3 CRF	Critical Request Flow. 0 For packets that are destined for the same output port of a swich, the flow associated with this packet shall, if this feature is implemented, not receive precedence over other packets at the same flow level. 1 For packets that are destined for the same output port of a swich, the flow associated with this packet shall, if this feature is implemented, receive precedence over other packets at the same flow level but have this bit as a 0.
4–5 TFLOWLV	Transaction flow level. This field must be set to 00 if the PCI_Window bit is set 00 Lowest priority transaction request flow 01 Next highest priority transaction request flow 10 Highest priority level transaction request flow 11 Reserved
6–7 -	This field is reserved. Reserved

Table continues on the next page...

SRIO_P2ROWS2R*n* field descriptions (continued)

Field	Description
8–11 RDTYP	Transaction type to run on RapidIO interface if access was a read. 0000 Reserved 0001 Reserved 0010 IO_READ_HOME 0011 Reserved 0100 NREAD 0101 Reserved 0110 Reserved 0111 MAINTENANCE read 1000 Reserved 1100 ATOMIC increment 1101 ATOMIC decrement 1110 ATOMIC set 1111 ATOMIC clear
12–15 WRTYP	Transaction type to run on RapidIO interface if access was a write. Writes-requiring-response sent from the internal platform must generate a write-requiring-response to RapidIO. Therefore, if an internal platform write-requiring-response request hits a window with WRTYP = SWRITE or NWRITE, the RapidIO endpoint generates a NWRITE_R instead. 0000 Reserved 0001 FLUSH 0010 Reserved 0011 SWRITE 0100 NWRITE 0101 NWRITE_R 0110 Reserved 0111 MAINTENANCE write 1000 Reserved 1110 Reserved 1111 Reserved
16–23 -	This field is reserved. Reserved
24–28 SGTGTDID1	Bits 0-4 (or bits 8-12 if large transport system) of the target device ID for this segment.
29 SGTGTDID2	Bit 5 (or bit 13 if large transport system) of the target deviceID; this bit is reserved if 8 target subsegments are selected.
30 SGTGTDID3	Bit 6 (or bit 14 if large transport system) of the target deviceID; this bit is reserved if 8 or 4 target subsegments are selected.
31 SGTGTDID4	Bit 7 (or bit 15 if large transport system) of the target deviceID; this bit is reserved if 8, 4, or 2 target subsegments are selected.

21.5.109 Port 2 RapidIO outbound window segment 3 register n (SRIO_P2ROWS3R*n*)

The port 2 RapidIO outbound window segment registers define the attributes and target device ID that are to be used for a transaction that hits in that segment rather than the primary attributes and target deviceID. There is one of these registers for each segment (except segment 0).

Address: C_0000h base + 1_0E3Ch offset + (32d × i), where i=0d to 7d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	Reserved			CRF	TFLOWLV		Reserved		RDTYP				WRTYP			
Reset	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	Reserved								SGTGTDID1					SGTGTDID2	SGTGTDID3	SGTGTDID4
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRIO_P2ROWS3R*n* field descriptions

Field	Description
0–2 -	This field is reserved. Reserved
3 CRF	Critical Request Flow. 0 For packets that are destined for the same output port of a swich, the flow associated with this packet shall, if this feature is implemented, not receive precedence over other packets at the same flow level. 1 For packets that are destined for the same output port of a swich, the flow associated with this packet shall, if this feature is implemented, receive precedence over other packets at the same flow level but have this bit as a 0.
4–5 TFLOWLV	Transaction flow level. This field must be set to 00 if the PCI_Window bit is set 00 Lowest priority transaction request flow 01 Next highest priority transaction request flow 10 Highest priority level transaction request flow 11 Reserved
6–7 -	This field is reserved. Reserved

Table continues on the next page...

SRIO_P2ROWS3R*n* field descriptions (continued)

Field	Description
8–11 RDTYP	Transaction type to run on RapidIO interface if access was a read. 0000 Reserved 0001 Reserved 0010 IO_READ_HOME 0011 Reserved 0100 NREAD 0101 Reserved 0110 Reserved 0111 MAINTENANCE read 1000 Reserved 1100 ATOMIC increment 1101 ATOMIC decrement 1110 ATOMIC set 1111 ATOMIC clear
12–15 WRTYP	Transaction type to run on RapidIO interface if access was a write. Writes-requiring-response sent from the internal platform must generate a write-requiring-response to RapidIO. Therefore, if an internal platform write-requiring-response request hits a window with WRTYP = SWRITE or NWRITE, the RapidIO endpoint generates a NWRITE_R instead. 0000 Reserved 0001 FLUSH 0010 Reserved 0011 SWRITE 0100 NWRITE 0101 NWRITE_R 0110 Reserved 0111 MAINTENANCE write 1000 Reserved 1110 Reserved 1111 Reserved
16–23 -	This field is reserved. Reserved
24–28 SGTGTDID1	Bits 0-4 (or bits 8-12 if large transport system) of the target device ID for this segment.
29 SGTGTDID2	Bit 5 (or bit 13 if large transport system) of the target deviceID; this bit is reserved if 8 target subsegments are selected.
30 SGTGTDID3	Bit 6 (or bit 14 if large transport system) of the target deviceID; this bit is reserved if 8 or 4 target subsegments are selected.
31 SGTGTDID4	Bit 7 (or bit 15 if large transport system) of the target deviceID; this bit is reserved if 8, 4, or 2 target subsegments are selected.

21.5.110 Port 2 RapidIO Inbound window translation address register n (SRIO_P2RIWTAR*n*)

The port 2 RapidIO inbound window translation address registers point to the starting addresses in local address space for window hits within the inbound translation windows. The new translated address is created by concatenating the transaction offset to this translation address.

Address: C_0000h base + 1_0F60h offset + (32d × i), where i=0d to 3d

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				TREXAD								TRAD																			
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

SRIO_P2RIWTAR*n* field descriptions

Field	Description
0–3 -	This field is reserved. Reserved
4–11 TREXAD	Translation extended address. Corresponds to bits 0-7 of the 40-bit internal platform translation address. TREXAD[6-7] are reserved for default window 0.
12–31 TRAD	Translation address. Target address which indicates the starting point of the inbound translated address. The translation address must be aligned based on the size field. This corresponds to bits 8-27 of the 40-bit internal platform translation address. TRAD is reserved for default window 0.

21.5.111 Port 2 RapidIO Inbound window base address register n (SRIO_P2RIWBAR*n*)

The port 2 RapidIO inbound window *n* base address registers select the base address for the windows which are translated to an alternate target address space. Addresses for inbound transactions are compared to these windows. If such a transaction does not fall within one of these spaces, then the transaction is forwarded to the interior of the chip using the default window. For transactions that cross more than one window, please see Window Boundary Crossing Errors-Inbound. Note that the LCSBA1CSR register (See Local configuration space base address 1 command and status register (SRIO_LCSBA1CSR)) has priority over all ATMU windows if both are configured for the same address space.

Address: C_0000h base + 1_0F68h offset + (32d × i), where i=0d to 3d

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

SRIO_P2RIWBAR*n* field descriptions

Field	Description
0–9 -	This field is reserved. Reserved
10–11 BEXAD	Base extended address. BEXAD represents bits 0-1 of the 34-bit RapidIO address.
12–31 BADD	Base address. System address which is the starting point for the inbound translation window. The window must be aligned based on the size selected in the window size bits. This corresponds to bits 2-21 of the 34-bit RapidIO base address.

21.5.112 Port 2 RapidIO inbound window attributes register n (SRIO_P2RIWAR*n*)

The port 2 RapidIO inbound window attributes registers define the window sizes to translate and other attributes for the translations. 16 Gbytes is the largest window size allowed. The RDTYP and WRTYP fields are used for attributes on the request, but do not actually change the transaction type of the transaction. In other words, PnRIWARn does not modify whether or not the request requires a response or if the request was atomic; this type of attribute remains unchanged on the request as it is translated through the ATMU.

Address: C_0000h base + 1_0F70h offset + (32d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	EN	PW	Reserved						TGINT				RDTYP			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

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

SRIO_P2RIWAR*n* field descriptions

Field	Description
0 EN	This bit enables this address translation. This field is set to 1 and read-only for default window 0.
1 PW	Protected Window. This bit indicates that this window is protected. Writes requiring response and reads to this window generate an error response. Writes not requiring response are silently discarded.
2–7 -	This field is reserved. Reserved
8–11 TGINT	Target interface. If this field is set to anything other than local address space, the attributes for the transaction must be assigned in a corresponding outbound window at the target. 0001 PCI Express 1 0011 PCI Express 2 0010 PCI Express 3 0101 PCI Express 4 0001 PCI Express 1 1000 Serial RapidIO port 1 1001 Serial RapidIO port 2
12–15 RDTYP	Transaction type to run on the I/O interface if access is a read. 0000 Reserved ... 0100 Read

Table continues on the next page...

SRIO_P2RIWAR*n* field descriptions (continued)

Field	Description
	0101 Reserved ... 1111 Reserved Transaction type to run on local memory if access is a read. 0000 Reserved ... 0100 Read, don't snoop local processor 0101 Read, snoop local processor 0110 Reserved 0111 Reserved ... 1100 Enhanced read, don't snoop local processor 1101 Enhanced read, snoop local processor ... 1111 Reserved
16–19 WRTYP	Transaction type to run on I/O interface if access is a write. 0000 Reserved ... 0100 Write 0101 Reserved ... 1111 Reserved Transaction type to run on local memory if access is a write. 0000 Reserved ... 0011 Reserved 0100 Write, don't snoop local processor 0101 Write, snoop local processor 0110 Reserved ... 1100 Write, don't snoop local processor 1101 Write, snoop local processor ... 1111 Reserved
20–25 -	This field is reserved. Reserved
26–31 SIZE	Inbound translation window size N which is the encoded 2^(N+1) bytes window size. The smallest window size is 4 Kbytes.

Table continues on the next page...

SRIO_P2RIWAR*n* field descriptions (continued)

Field	Description
	This field is read only for default window 0. 000000 Reserved 001011 4-Kbyte window size 001100 8-Kbyte window size 011111 4-Gbyte window size 100000 8-Gbyte window size 100001 16-Gbyte window size 100010 Reserved 111111 Reserved

21.5.113 Port 2 RapidIO inbound window translation address register 0 (SRIO_P2RIWTAR0)

The port 2 RapidIO inbound window translation address registers point to the starting addresses in local address space for window hits within the inbound translation windows. The new translated address is created by concatenating the transaction offset to this translation address.

Address: C_0000h base + 1_0FE0h offset = D_0FE0h

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				TREXAD						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

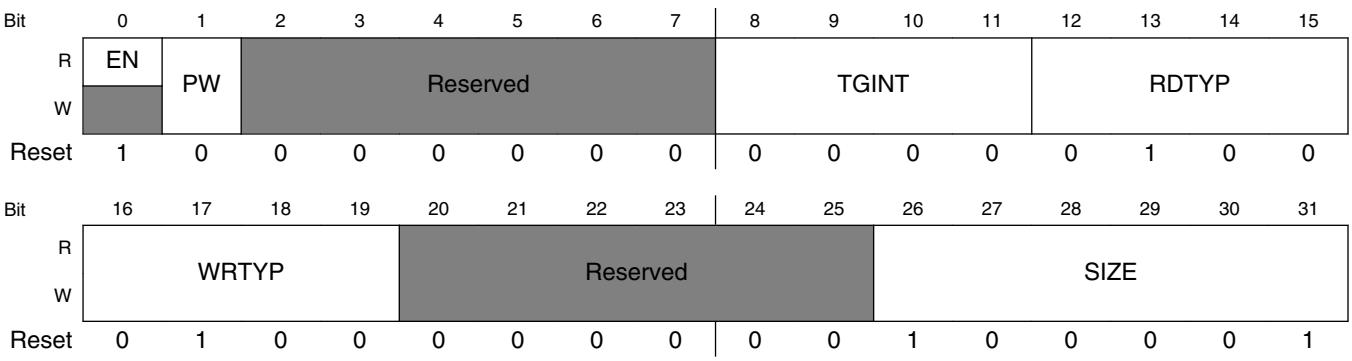
SRIO_P2RIWTAR0 field descriptions

Field	Description
0–3 -	This field is reserved. Reserved
4–9 TREXAD	Translation extended address. Corresponds to bits 0-5 of the 40-bit internal platform translation address.
10–31 -	This field is reserved. Reserved

21.5.114 Port 2 RapidIO inbound window attributes register 0 (SRIO_P2RIWAR0)

The port 2 RapidIO inbound window attributes registers define the window sizes to translate and other attributes for the translations. 16 Gbytes is the largest window size allowed. The RDTYP and WRTYP fields are used for attributes on the request, but do not actually change the transaction type of the transaction. In other words, PnRIWARn does not modify whether or not the request requires a response or if the request was atomic; this type of attribute remains unchanged on the request as it is translated through the ATMU.

Address: C_0000h base + 1_0FF0h offset = D_0FF0h



SRIO_P2RIWAR0 field descriptions

Field	Description
0 EN	This bit enables this address translation. This field is set to 1 and read-only for default window 0.
1 PW	Protected Window. This bit indicates that this window is protected. Writes requiring response and reads to this window generate an error response. Writes not requiring response are silently discarded.
2–7 -	This field is reserved. Reserved
8–11 TGINT	Target interface. If this field is set to anything other than local address space, the attributes for the transaction must be assigned in a corresponding outbound window at the target. 0001 PCI Express 1 0011 PCI Express 2 0010 PCI Express 3 0101 PCI Express 4 0001 PCI Express 1 1000 Serial RapidIO port 1 1001 Serial RapidIO port 2
12–15 RDTYP	Transaction type to run on the I/O interface if access is a read.

Table continues on the next page...

SRIO_P2RIWAR0 field descriptions (continued)

Field	Description
	0000 Reserved ... 0100 Read 0101 Reserved ... 1111 Reserved Transaction type to run on local memory if access is a read. 0000 Reserved ... 0100 Read, don't snoop local processor 0101 Read, snoop local processor 0110 Reserved 0111 Reserved ... 1100 Enhanced read, don't snoop local processor 1101 Enhanced read, snoop local processor ... 1111 Reserved
16–19 WRTYP	Transaction type to run on I/O interface if access is a write. 0000 Reserved ... 0100 Write 0101 Reserved ... 1111 Reserved Transaction type to run on local memory if access is a write. 0000 Reserved ... 0011 Reserved 0100 Write, don't snoop local processor 0101 Write, snoop local processor 0110 Reserved ... 1100 Write, don't snoop local processor 1101 Write, snoop local processor ... 1111 Reserved

Table continues on the next page...

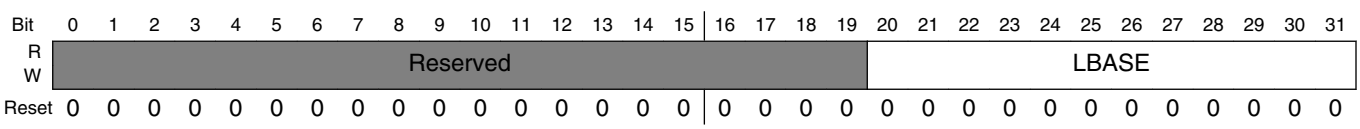
SRIO_P2RIWAR0 field descriptions (continued)

Field	Description
20–25 -	This field is reserved. Reserved
26–31 SIZE	Inbound translation window size N which is the encoded 2^(N+1) bytes window size. The smallest window size is 4 Kbytes. This field is read only for default window 0. 000000 Reserved 001011 4-Kbyte window size 001100 8-Kbyte window size 011111 4-Gbyte window size 100000 8-Gbyte window size 100001 16-Gbyte window size 100010 Reserved 111111 Reserved

21.5.115 Port 1 LIODN Base Register (SRIO_P1LBR)

This register contains the LIODN base used for inbound translation from RapidIO source ID to LIODN.

Address: C_0000h base + 1_3000h offset = D_3000h



SRIO_P1LBR field descriptions

Field	Description
0–19 -	This field is reserved. Reserved
20–31 LBASE	LIODN Base: The LIODN base is added to an LIODN offset to generate an LIODN value.

21.5.116 Port 1 LIODN Address Offset Register (SRIO_P1LAOR)

This register contains the address offset of an entry within the LIODN lookup table. This is used for accesses to and from the LIODN lookup table through SRIOPn_LUDR and SRIOPn_LLDR.

Address: C_0000h base + 1_3020h offset = D_3020h

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																							ADDR						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

SRIO_P1LAOR field descriptions

Field	Description
0–22 -	This field is reserved. Reserved
23–28 ADDR	LIODN Base: The LIODN base is added to an LIODN offset to generate an LIODN value.
29–31 -	This field is reserved. Reserved

21.5.117 Port 1 LIODN upper data Register (SRIO_P1LUDR)

This register contains upper data fields of an entry within the LIODN lookup table. This register is used for accesses to and from the LIODN lookup table. A read to this register will read the upper data fields of the entry within the LIODN lookup table, indicated by SRIOPn_LAOR[ADDR]. A write to this register will write the upper data fields of the entry within the LIODN lookup table, indicated by SRIOPn_LAOR[ADDR]..

Address: C_0000h base + 1_3030h offset = D_3030h

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	SRCID																MASK															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRIO_P1LUDR field descriptions

Field	Description
0–15 SRCID	Source ID: The source ID field of an entry within the LIODN lookup table. This field is qualified with the MASK field of the same LIODN lookup table entry when attempting to match. Note: **NOTE:** All 16 bits of SRCID are compared against the source ID field of a request packet. 8 bits of 0s are pre-pended to the source ID field of a small transport packet before comparison with SRCID.
16–31 MASK	Mask: The mask field of an entry within the LIODN lookup table. This field is qualified with the SRCID field of the same LIODN lookup table entry and a received packet's source ID when attempting to match. It is used to bit-wise disable matches against SRCID and a received packet's source ID. **NOTE:** The reset value is 0xFFFF for entry 0 and 0x0000 for all other entries of the LIODN lookup table.

21.5.118 Port 1 LIODN lower data register (SRIO_P1LLDR)

This register contains the lower data fields of an entry within the LIODN lookup table. This register is used for accesses to and from the LIODN lookup table. A read to this register will read the lower data fields of the entry within the LIODN lookup table, indicated by SRIOPn_LAOR[ADDR]. A write to this register will write the lower data fields of the entry within the LIODN lookup table, indicated by SRIOPn_LAOR[ADDR].

Address: C_0000h base + 1_3034h offset = D_3034h

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

SRIO_P1LLDR field descriptions

Field	Description
0–1 -	This field is reserved. Reserved
2–11 LOFFSET	LIODN Offset: The LIODN offset field of an entry within the LIODN lookup table. If this entry matches, this field is added to SRIOPn_LBR[LBASE] to generate an LIODN value.
12–30 -	This field is reserved. Reserved
31 EN	Enable: The enable field of an entry within the LIODN lookup table. This field identifies whether the entry participates in matching.

Table continues on the next page...

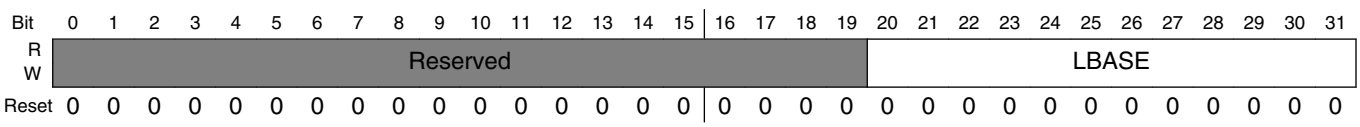
SRIO_P1LLDR field descriptions (continued)

Field	Description
	NOTE: The reset value is 0x1 for entry 0 and 0x0 for all other entries of the LIODN lookup table. 0 This entry does not participate in matching. 1 This entry participates in matching.

21.5.119 Port 2 LIODN Base Register (SRIO_P2LBR)

This register contains the LIODN base used for inbound translation from RapidIO source ID to LIODN.

Address: C_0000h base + 1_3200h offset = D_3200h



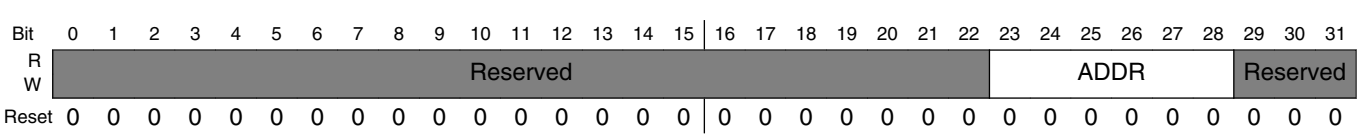
SRIO_P2LBR field descriptions

Field	Description
0–19 -	This field is reserved. Reserved
20–31 LBASE	LIODN Base: The LIODN base is added to an LIODN offset to generate an LIODN value.

21.5.120 Port 2 LIODN Address Offset Register (SRIO_P2LAOR)

This register contains the address offset of an entry within the LIODN lookup table. This is used for accesses to and from the LIODN lookup table through SRIOPn_LUDR and SRIOPn_LLDR.

Address: C_0000h base + 1_3220h offset = D_3220h



SRIO_P2LAOR field descriptions

Field	Description
0–22 -	This field is reserved. Reserved
23–28 ADDR	LIODN Base: The LIODN base is added to an LIODN offset to generate an LIODN value.
29–31 -	This field is reserved. Reserved

21.5.121 Port 2 LIODN upper data Register (SRIO_P2LUDR)

This register contains upper data fields of an entry within the LIODN lookup table. This register is used for accesses to and from the LIODN lookup table. A read to this register will read the upper data fields of the entry within the LIODN lookup table, indicated by SRIOPn_LAOR[ADDR]. A write to this register will write the upper data fields of the entry within the LIODN lookup table, indicated by SRIOPn_LAOR[ADDR]..

Address: C_0000h base + 1_3230h offset = D_3230h

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	SRCID																MASK															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRIO_P2LUDR field descriptions

Field	Description
0–15 SRCID	Source ID: The source ID field of an entry within the LIODN lookup table. This field is qualified with the MASK field of the same LIODN lookup table entry when attempting to match. Note: **NOTE:** All 16 bits of SRCID are compared against the source ID field of a request packet. 8 bits of 0s are pre-pended to the source ID field of a small transport packet before comparison with SRCID.
16–31 MASK	Mask: The mask field of an entry within the LIODN lookup table. This field is qualified with the SRCID field of the same LIODN lookup table entry and a received packet's source ID when attempting to match. It is used to bit-wise disable matches against SRCID and a received packet's source ID. **NOTE:** The reset value is 0xFFFF for entry 0 and 0x0000 for all other entries of the LIODN lookup table.

21.5.122 Port 2 LIODN lower data Register (SRIO_P2LLDR)

This register contains the lower data fields of an entry within the LIODN lookup table. This register is used for accesses to and from the LIODN lookup table. A read to this register will read the lower data fields of the entry within the LIODN lookup table, indicated by SRIOPn_LAOR[ADDR]. A write to this register will write the lower data fields of the entry within the LIODN lookup table, indicated by SRIOPn_LAOR[ADDR].

Address: C_0000h base + 1_3234h offset = D_3234h

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

SRIO_P2LLDR field descriptions

Field	Description
0–1 -	This field is reserved. Reserved
2–11 LOFFSET	LIODN Offset: The LIODN offset field of an entry within the LIODN lookup table. If this entry matches, this field is added to SRIOPn_LBR[LBASE] to generate an LIODN value.
12–30 -	This field is reserved. Reserved
31 EN	Enable: The enable field of an entry within the LIODN lookup table. This field identifies whether the entry participates in matching. **NOTE:** The reset value is 0x1 for entry 0 and 0x0 for all other entries of the LIODN lookup table. 0 This entry does not participate in matching. 1 This entry participates in matching.

21.6 RapidIO Functional Description

21.6.1 RapidIO Implementation Space-ATMU Registers

ATMU registers are used for outbound and inbound transactions. Their purpose is to translate RapidIO packets to internal platform packets on inbound and to translate internal platform packets to RapidIO packets on outbound. ATMU window misses use the window 0 register set by default, and overlapping window hits results in the use of the lowest number window register set hit. For both inbound and outbound translation, the smallest window size is 4 Kbytes and the largest window size is 16G for inbound translation and 1T for outbound translation. The default window register set causes no translation of the transaction address for inbound transactions since the RapidIO address space has 34 bits and the internal platform address space has 40 bits. For outbound transactions, the default window maps each of the 64 16G chunks to the RapidIO 16-Gbyte address space. The inbound and outbound translation windows must be aligned based on the granularity selected by the size fields. The packet device ID fields are not used in the inbound translation process, only the address field.

The RapidIO endpoint implementation allows up to a 34-bit (0:33) RapidIO address and a 40-bit (0:39) internal platform address. As is the case with all registers, an external processor writing the ATMU registers should not assume that the write has completed until a response is received. A user must guarantee that the address tenures have stopped being initiated from those masters before starting to modify any ATMU settings and all traffic must be haulted during any reconfiguration of ATMU registers.

Note that when booting from serial RapidIO, outbound ATMU window 0 must be used.

21.6.1.1 Segmented Outbound Window Description

All outbound windows have the capability to have 2 or 4 segments, all of which are equal in size, numbered 0 and 1, or 0 through 3, respectively . Each segment assigns attributes and the target deviceID for an outbound transaction. All segments of a window translate to the same translation address in the target.

Additionally, each segment can be set up with 2, 4, or 8 subsegments, all of which are equal in size. These subsegments allow a single segment to target a number of numerically adjacent target device IDs, and, again, they all translate to the same translation address in the targets. For example, a segment with 8 subsegments can be configured to generate a transaction with the same set of attributes to target deviceIDs 0, 1, 2, 3, 4, 5, 6, or 7, depending on which subsegment is addressed.

Note that subsegments are only supported when multiple segments are chosen.

This allows a window to be configured so that aliases can be created to the same offset within the target device so that a single window can be used to generate different transaction types. Without segmented windows, achieving the equivalent behavior would require multiple windows. Figure 21-260 shows an example of this capability. A window is defined to be 4kB in size, and is defined to have 4 segments and no subsegments. Each segment is assigned to target deviceID 0x05, and each segment is given a different write transaction type attribute - segment 0 is assigned NWRITE, segment 1 is assigned SWRITE, segment 2 is assigned NWRITE_R, and segment 3 is assigned FLUSH. Since all of the segments are assigned to target the same device, by writing to the same offset in each segment, a different write transaction can be generated to the target to the same offset in the target.

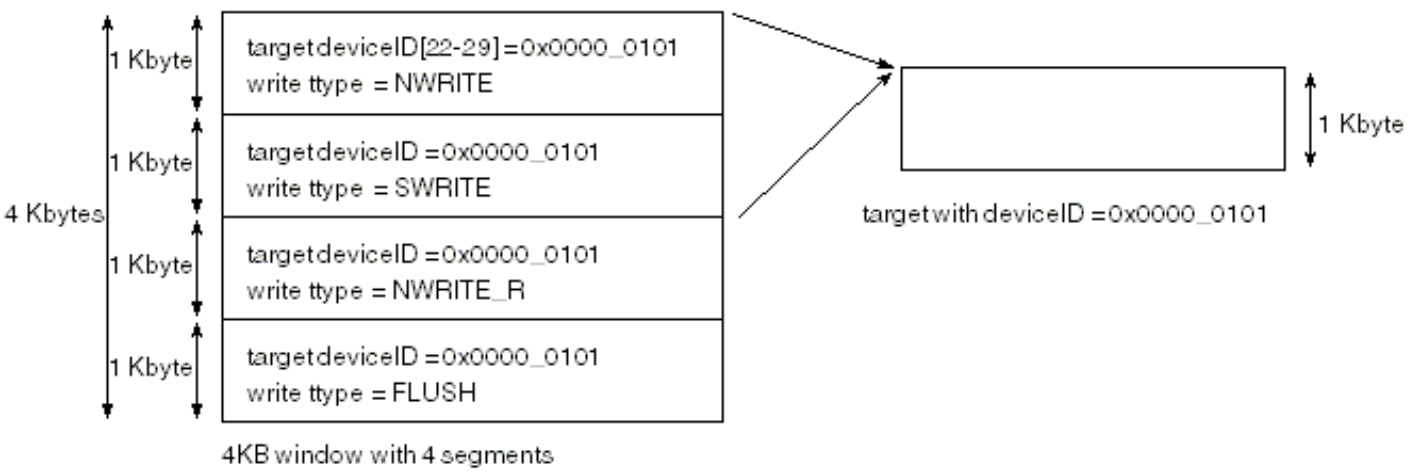


Figure 21-260. Example of Attribute Aliasing

So, writing to offset 0x0 in segment 0 is translated (as defined in the translation address registers) and generates a NWRITE transaction to offset 0x0 in a 1KB window in the target with deviceID = 0x05. A write to offset 0x0 in segment 2 is also translated, to the same offset in the target device as the write to segment 0, but this time a NWRITE_R transaction is generated.

Another use is that the same window can be used to target multiple devices with the same translation offset. Without segmented, (and subsegmented) windows, achieving the equivalent behavior would require multiple windows. Figure 21-261 shows an example of this multi-targeting. For example, a 4kB window is set up with 2 segments of 2 subsegments. Each segment is assigned a write ttype of NWRITE, but each segment and subsegment has a different target deviceID. Segments 0 and 1 are assigned target deviceIDs 4 and 5, and 8 and 9, respectively.

Figure 21-261. Example of Multi-Targeting

In this example, a write to offset 0x0 in segment 0 is translated as defined, and a NWRITE transaction is generated targeted to deviceID 4. A corresponding write to segment 1 to offset 0x400 is also translated but also using the assigned deviceID instead of the translation address bits [22-29]. The generated NWRITE transaction has the same target device offset as the write to segment 0, but is instead targeted to deviceID 9.

Combinations of aliasing and multi-targeting are also possible for a window.

21.6.2 RapidIO Transaction Summary

The RapidIO endpoint on this device supports all RapidIO I/O transactions, all RapidIO message passing transactions, and a few RapidIO GSM transactions.

This table summarizes the RapidIO I/O transactions that are supported by this RapidIO endpoint.

Table 21-259. RapidIO I/O Transactions

IO Transaction	ftype	ttype	Status	Description
NREAD	0010	0100	NA	Read
ATOMIC inc		1100		Read and post-increment with response
ATOMIC dec		1101		Read and post-decrement with response
ATOMIC set		1110		Read and set to all-1s with response

Table continues on the next page...

Table 21-259. RapidIO I/O Transactions (continued)

IO Transaction	ftype	ttype	Status	Description
ATOMIC clr		1111		Read and set to all-0s with response
NWRITE	0101	0100		Write with no response
NWRITE_R		0101		Write with response
SWRITE	0110	N/A		Streaming-Write
MAINT read	1000	0000		Maintenance read
MAINT write		0001		Maintenance write
MAINT read response		0010	0000	Done maintenance read response
			0111	Error response
MAINT write response		0011	0000	Done maintenance write response
			0111	Error response
MAINT port-write		0100	NA	Maintenance port-write1
RESPONSE without data	1101	0000	0000	I/O done response
			0111	I/O error response
RESPONSE with data		1000	0000	I/O done response with data

1). Limited to inbound RapidIO packets only

This table summarizes the RapidIO message passing transactions that are supported by this RapidIO endpoint.

Table 21-260. RapidIO Message Passing Transactions

MSG Transaction	ftype	ttype	status	Description
DOORBELL	1010	NA	NA	Doorbell
MESSAGE	1011			Message
RESPONSE without data	1101	0000	0000	Doorbell done response
			0011	Doorbell retry response
			0111	Doorbell error response
		0001	0000	Message done response
			0011	Message retry response
			0111	Message error response

This table summarizes the RapidIO data streaming (type 9) transactions that are supported by this RapidIO endpoint.

Table 21-261. RapidIO Data Streaming Transactions

Data Streaming Header Type	ftype	start	end	xh	Description
START	1001	1	0	0	Start segment packet.
SINGLE	1001	1	1	0	Single segment packet.

Table continues on the next page...

Table 21-261. RapidIO Data Streaming Transactions (continued)

Data Streaming Header Type	ftype	start	end	xh	Description
CONTINUATION	1001	0	0	0	Continuation segment packet.
END	1001	0	1	0	End segment packet.
FLOW CONTROL	1001	-	-	1	Flow Control packet.

This table summarizes the RapidIO GSM transactions that are supported by this RapidIO endpoint.

Table 21-262. RapidIO GSM Transactions

GSM Transaction	ftype	ttype	status	Description
IO_READ_HOME	0010	0010	NA	I/O Read Home1
FLUSH w/data	0101	0001		GSM Flush with data1
RESPONSE without data	1101	0000	0000	GSM done response
			0011	GSM retry response
			0101	GSM done intervention response
			0111	GSM error response
RESPONSE with data		1000	0000	GSM done response
			0001	GSM data only response

1). Limited to RapidIO packet generation only

21.6.3 RapidIO Packet Format Summary

This table summarizes the small transport field packet formats for supported RapidIO transaction types for LP-Serial operation. Details of packet fields are located in the RapidIO Interconnect Specification, Revision 2.0.1.

Transaction ID (TID) source bits and sizes are defined as: rsv[0:2], doorbell type bit [3], source TID[4:7].

Note that this RapidIO endpoint limits configuration read and write requests to 32-bit data accesses.

Table 21-263. RapidIO Small Transport Field Packet Format

Transaction	Bits																
	32								32					16			64
	5	2	1	2	2	4	8	8	4	4	8	8	8	13	1	2	
NREAD, ATOMIC (inc/dec/inc/dec), IO_READ_HOME	ack ID	rsv=0	crf	prio	tt	ftype	dest ID	src ID	ttype	rdsize	src TID	addr			wd ptr	xam bs	NA
NWRITE_R, FLUSH w/data	ack ID	rsv=0	crf	prio	tt	ftype	dest ID	src ID	ttype	wrsize	src TID	addr			wd ptr	xam bs	dword 0 -> dword n
NWRITE	ack ID	rsv=0	crf	prio	tt	ftype	dest ID	src ID	ttype	wrsize	don't care	addr			wd ptr	xam bs	dword 0 -> dword n
SWRITE	ack ID	rsv=0	crf	prio	tt	ftype	dest ID	src ID	addr(29), rsv(1)=0, xambs(2)					dword 0 -> dword n			
MAINT read	ack ID	rsv=0	crf	prio	tt	ftype	dest ID	src ID	ttype	rdsize	src TID	hop cnt	cfg offset		wd ptr	rsv=0	NA
MAINT write	ack ID	rsv=0	crf	prio	tt	ftype	dest ID	src ID	ttype	wrsize	src TID	hop cnt	cfg offset		wd ptr	rsv=0	dword 0 (32-bit)
MAINT port-write	ack ID	rsv=0	crf	prio	tt	ftype	dest ID	src ID	ttype	wrsize	rsv=0	hop cnt	rsv=0		wd ptr	rsv=0	dword 0 -> dword n
MAINT response without data	ack ID	rsv=0	crf	prio	tt	ftype	dest ID	src ID	ttype	status	tar TID	hop cnt	rsv=0				NA
MAINT response with data	ack ID	rsv=0	crf	prio	tt	ftype	dest ID	src ID	ttype	status	tar TID	hop cnt	rsv=0				dword 0 (32-bit)
RESPONSE without data	ack ID	rsv=0	crf	prio	tt	ftype	dest ID	src ID	ttype	status	tar TID2	NA					
RESPONSE without data for message	ack ID	rsv=0	crf	prio	tt	ftype	dest ID	src ID	ttype	status	letter(2), mbox(2), msgseg(4)	NA					
RESPONSE with data	ack ID	rsv=0	crf	prio	tt	ftype	dest ID	src ID	ttype	status	tar TID	dword 0 -> dword n					
DOORBELL	ack ID	rsv=0	crf	prio	tt	ftype	destID	src ID	rsv=0		src TID[1]	Info-msb	Info-lsb	NA			
MESSAGE	ack ID	rsv=0	crf	prio	tt	ftype	dest ID	src ID	msglen(4), ssize(4), letter(2), mbox(2), msgseg(4)			dword 0 -> dword n					
DATA STREAMING (start)	ack ID	rsv=0	crf	prio	tt	ftype	dest ID	src ID	cos		s(1),e(1), rsv(3), xh(1)=0, rsv(2)	streamID		hword 0 -> hword n			
DATA STREAMING (single)	ack ID	rsv=0	crf	prio	tt	ftype	dest ID	src ID	cos		s(1),e(1), rsv(3), xh(1)=0, o(1),p(1)	streamID		hword 0 -> hword n			

Table continues on the next page...

Table 21-263. RapidIO Small Transport Field Packet Format (continued)

Transaction	Bits																
	32								32					16			64
	5	2	1	2	2	4	8	8	4	4	8	8	8	13	1	2	
DATA STREAMING (continuation)	ack ID	rsv =0	crf	prio	tt	ftype	dest ID	src ID	cos		s(1),e(1), rsv(3), xh(1)=0, rsv(2)	hword 0 -> hword n					
DATA STREAMING (end)	ack ID	rsv =0	crf	prio	tt	ftype	dest ID	src ID	cos		s(1),e(1), rsv(3), xh(1)=0, o(1),p(1)	length		hword 0 -> hword n			
DATA STREAMING (Flow Control)	ack ID	rsv =0	crf	prio	tt	ftype	dest ID	src ID	cos		rsv(2), xtype(3), xh(1)=1, rsv(2)	length		hword 0 -> hword n			

1. src TID[3] of [0:7] bus must be set to '1' for doorbell accesses.2 tar TID[3] of [0:7] bus must be set to '1' for doorbell responses.

The large transport field packet formats extends the destination and source IDs to 16-bits each.

21.6.4 RapidIO Control Symbol Summary.

This table summarizes the 1x/4x LP-Serial control symbols and their format. Refer to the *RapidIO Interconnect Specification, Revision 1.2, Part IV: Physical Layer 1x/4x LP-Serial Specifications*, *Chapter4, PCS and PMA Layers,* for 8B/10B data and special (/PD/, /SC/, idle, sync, skip, align) characters. The 32-bit LP-Serial control symbol is comprised of the eight bit special character and the 24-bit control symbol format.

Table 21-264. 1x/4x LP-Serial Control Symbol Format

Bits						Description
24						
stype0	param0	param1	stype1	cmd	CRC	
3	5	5	3	3	5	
000	pkt_ackID	buf_stat	-		crc	Packet accepted
001	pkt_ackID	buf_stat	-		crc	Packet retry
010	pkt_ackID	cause	-		crc	Packet not accepted cause: 00001: Received unexpected ackID on packet

Table continues on the next page...

Table 21-264. 1x/4x LP-Serial Control Symbol Format (continued)

Bits						Description
24						
stype0	param0	param1	stype1	cmd	CRC	
3	5	5	3	3	5	
						00010: Received a control symbol with bad CRC 00011: Non-maintenance packet reception is stopped 00100: Received packet with bad CRC 00101: Received invalid character or a valid but illegal character 11111: General error
100	ackID_stat	buf_stat	-		crc	Status ackID_stat: 00000: Expecting ackID 0 00001: Expecting ackID 1 00010: Expecting ackID 2 00011: Expecting ackID 3 00100: Expecting ackID 4 00101: Expecting ackID 5 00110: Expecting ackID 6 00111: Expecting ackID 7
110	ackID_stat	port_stat	-		crc	Link-response ackID_stat: 00000: Expecting ackID 0 00001: Expecting ackID 1 00010: Expecting ackID 2 00011: Expecting ackID 3 00100: Expecting ackID 4 00101: Expecting ackID 5 00110: Expecting ackID 6 00111: Expecting ackID 7 port_stat: 00010: Error; unrecoverable 00100: Retry stopped 00101: Error stopped 10000: OK
-			000	000	crc	Start of packet
-			001	000	crc	Stomp
-			010	000	crc	End of packet
-			011	000	crc	Restart from retry

Table continues on the next page...

Table 21-264. 1x/4x LP-Serial Control Symbol Format (continued)

Bits						Description
24						
stype0	param0	param1	stype1	cmd	CRC	
3	5	5	3	3	5	
-			100	cmd	crc	Link request cmd: 011: Reset the receiving device 100: Return input port status; functions as a restart-from-error control symbol under error conditions
-			101	000	crc	Multicast-event
-			111	000	crc	NOP (ignore)

21.6.5 Accessing Configuration Registers Using RapidIO Packets

21.6.5.1 Inbound Maintenance Accesses

There are two suggested methods by which RapidIO transactions can target RapidIO configuration register space in local memory.

The first method is based on RapidIO NREAD and NWRITE_R requests hitting a RapidIO address window defined by the local configuration space base address command and status register (LCSBACSR). See Local configuration space base address 1 command and status register (SRIO_LCSBA1CSR), for details.

If external configuration accesses are disabled (LLCR[ECRAB] = 1), any configuration access through the LCSBACSR window is denied. A 32-bit data payload of all zeros is returned for a non-maintenance configuration read.

The second method is based on a RapidIO MAINT requests. This method allows an external device limited access to local RapidIO configuration register space only. Any maintenance access beyond the first 64 Kbytes (lower 64KB contains RapidIO architecture registers; upper 64KB contains RapidIO implementation registers) of RapidIO configuration register space, is denied. A 32-bit data payload of all zeros is returned if for a read response.

A third method, though not suggested, would use an inbound ATMU window to translate RapidIO NREAD and NWRITE_R requests to configuration accesses. This method does not support the configuration access protection features offered by the LCSBACSR window and RapidIO MAINT requests.

21.6.5.2 Outbound Maintenance Accesses

Outbound internal platform NREAD_R or NWRITE requests can be translated to a RapidIO maintenance request if the internal platform address falls within the bounds of an outbound ATMU window that is setup for generating a maintenance request. The ATMU window specifies the configuration offset, hop count, source and destination ID, critical request flow, and priority for the outbound RapidIO packet.

21.6.6 Interaction with the Message Unit

The RapidIO controller is designed to interact with a message unit, which provides it with a message passing architecture.

There are both inbound (Rx) and outbound (Tx) interfaces with the message unit, in addition to the Rx and Tx interfaces with the system and link.

This figure shows RapidIO controller's interaction with the message unit.

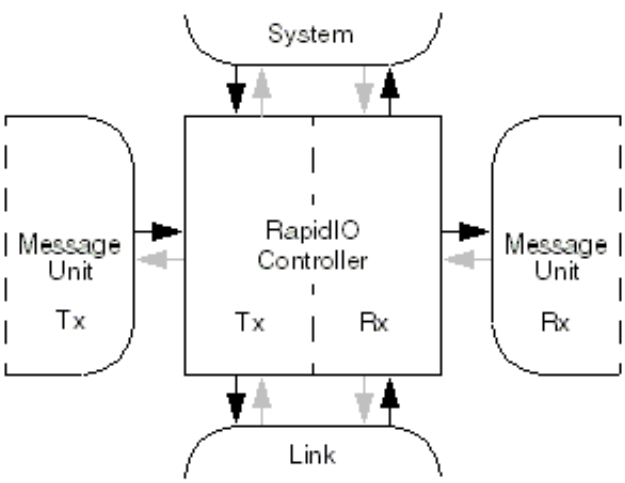


Figure 21-262. Interaction with the Message Unit

21.6.6.1 Inbound (Rx)

The inbound (Rx) interaction with the message unit is used receive message unit packets from the RapidIO link. The RapidIO link can receive message requests, responses, or flow control packets.

The inbound interaction directs incoming packets to either the message unit or the controller based upon various RapidIO fields. Inbound message packets will be buffered into either a request or response queue before being sent out to the message unit. A round robin arbitration scheme is used to select the next transaction out to the message unit.

21.6.6.2 Outbound (Tx)

The outbound (Tx) interaction with the message unit is used to transmit message unit packets to the RapidIO link. The message unit can transmit requests, responses, or flow control packets.

Tx message unit packets are bufferred within the RapidIO controller until they have been accepted on the link (received ack accepted). In addition, Tx message unit packets arbitrate with Tx system packets within the RapidIO controller before being transmitted to the link.

21.6.6.2.1 Buffer Allocation

The RapidIO controller contains buffers for Tx message unit packets.

The buffers are separated into two categories:

- Tx Message Unit Request Packets
- Tx Message Unit Response/Flow Control Packets

This figure shows the buffers for Tx message unit packets within the RapidIO controller.

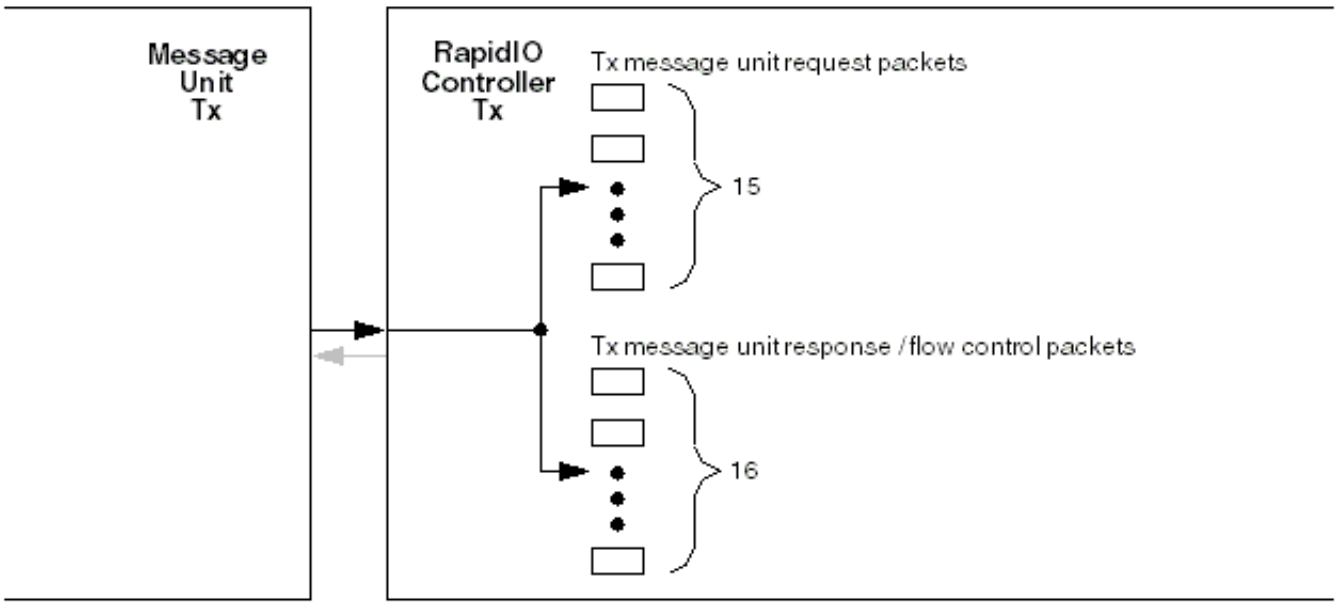


Figure 21-263. Tx Message Unit Packet Buffers

Within each category, there is dedicated and generic buffer allocation. Dedicated buffer allocation holds a specific type of message unit packet (within that category). Generic buffer allocation holds any type of message unit packet (within that category). Each buffer, regardless of its allocation (dedicated or generic), can only hold 1 message unit packet.

21.6.6.2.1.1 Tx Message Unit Request Packets

Tx mesage unit request packets are categorized according to an arbitration group (for example, arbitration group 0). Each arbitration group may have its own dedicated buffer allocation. In addition, there is also generic buffer allocation (holds any of the arbitration groups).

The number of dedicated and generic buffers for Tx message unit request packets is configurable with the following registers:

- Port 1 Message Request Tx Buffer Allocation Configuration Register 0 (SRIO_P1MReqTxBACR0) and Port 2 Message Request Tx Buffer Allocation Configuration Register 0 (SRIO_P2MReqTxBACR0)
- Port 1 Message Request Tx Buffer Allocation Configuration Register 2 (SRIO_P1MReqTxBACR2) and Port 2 Message Request Tx Buffer Allocation Configuration Register 2 (SRIO_P2MReqTxBACR2)

NOTE

There are only 15 buffers that hold message unit request packets. If the total value of the fields within the registers mentioned above exceeds 15 undefined behavior will result.

These registers are overwritten when the maximum arbitration group changes.

The new values support one dedicated buffer for each arbitration group up to the maximum, and the remaining buffers are generic. The total of all buffers (both dedicated and generic) should be 15.

For example, when the maximum arbitration group is set to 0, then DedAG0[3:0] is set to 4'h1 (one dedicated buffer for arbitration group 0) and GenAG[3:0] is set to 4'hE (the remaining 14 buffers are generic).

You can change these registers after the maximum arbitration group changes, and your new values are used. A subsequent change of the maximum arbitration group overwrites the values of these registers again.

21.6.6.2.1.2 Tx Message Unit Response/Flow Control Packets

There are two types of Tx message unit response/flow control packets:

- Tx message unit response packet
- Tx message unit flow control packet

Each type may have its own dedicated buffer allocation. In addition, there is also generic buffer allocation (holds any of the types mentioned above).

The number of dedicated and generic buffers for Tx message unit response/flow control packets is configurable with the following registers:

- Port 1 Message Response / Flow Control Tx Buffer Allocation Configuration Register (SRIO_P1MRspFcTxBACR) and Port 2 Message Response / Flow Control Tx Buffer Allocation Configuration Register (SRIO_P2MRspFcTxBACR)

NOTE

There are only 16 buffers that hold message unit response and flow control packets. The total value of the fields within the register mentioned above, must not exceed 16 (undefined behavior will result).

21.6.6.2.2 Arbitration

The RapidIO controller arbitrates amongst its Tx message unit packet buffers and an arbitration winner from the system packet buffers. Two main types of arbitration are used throughout: strict priority and weighted round robin.

Strict priority arbitration, with starvation avoidance selects a winner based on priority. Starvation avoidance, which may be disabled, elevates lower priority requestors to the highest priority. This can only occur when a starvation avoidance counter has expired. This counter only counts when a requestor has lost. A counter threshold of 0s will disable this feature. Note: disabling starvation avoidance may cause deadlocks.

Weighted round robin arbitration selects a winner based on round robin, but with weights. The weights delay moving to the next requestor until a number, equal to the corresponding weight, of winners has been chosen.

This figure shows the details of Tx message unit packet arbitration.

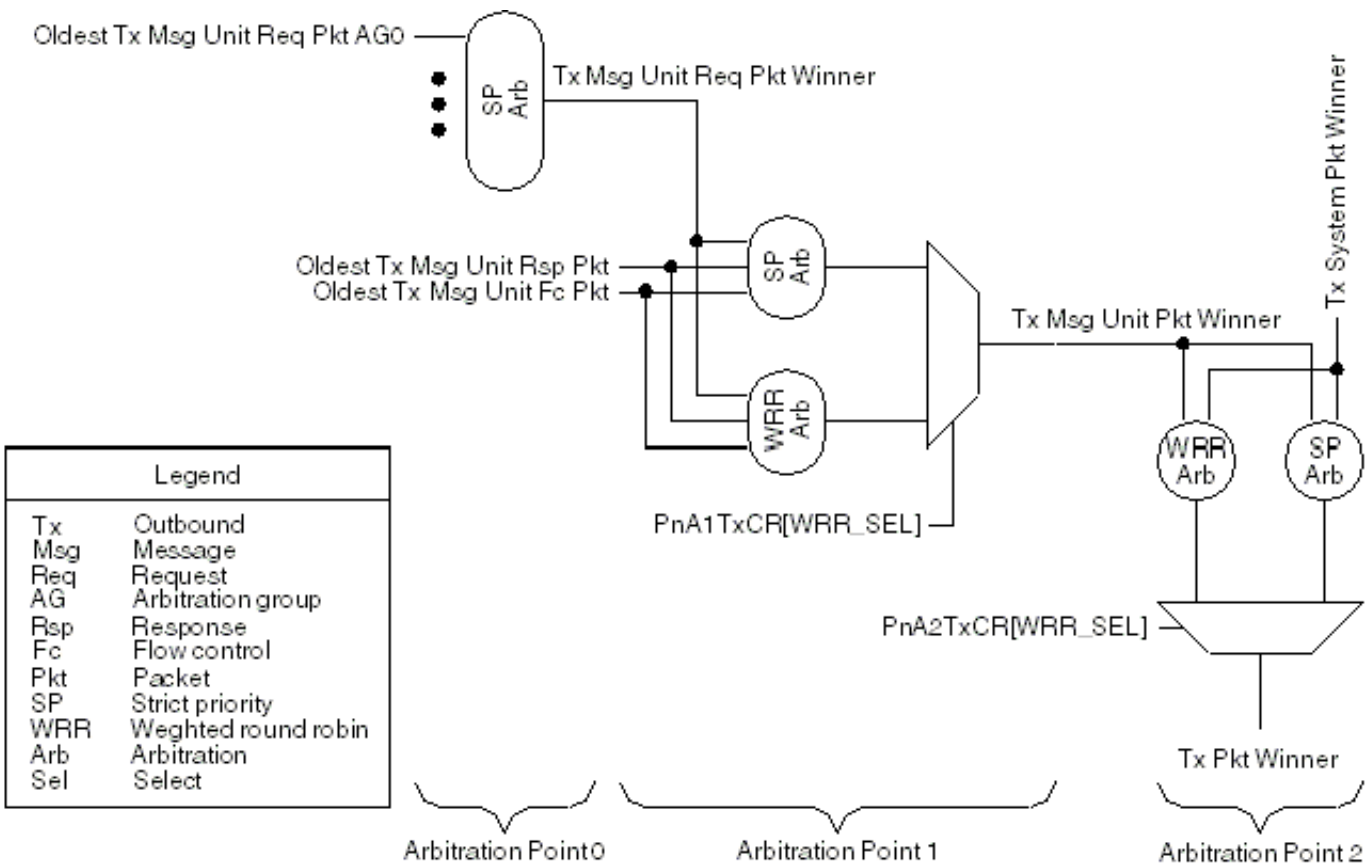


Figure 21-264. Tx Message Unit Packet Arbitration

21.6.6.2.2.1 Arbitration Point 0

Consider the following regarding arbitration point 0:

- It chooses a Tx message unit request packet winner. Only the oldest Tx message unit request packet of each arbitration group is eligible.
- Strict priority arbitration is used to select the winner.
- The priority of Tx message unit request packets is related to their arbitration group, with arbitration group 0 being the highest priority.
- It is configurable with Port 1 Arbitration 0 Tx Configuration Register (SRIO_P1A0TxCR) and Port 2 Arbitration 0 Tx Configuration Register (SRIO_P2A0TxCR).

21.6.6.2.2.2 Arbitration Point 1

Consider the following regarding arbitration point 1:

- It chooses a Tx message packet winner. Only the Tx message unit request packet winner (chosen from Arbitration Point 0), the oldest Tx message unit response packet, and the oldest Tx message unit flow control packet are eligible.
- Two types of arbitration are available:
  - Strict priority-If selected, the weighted round robin arbitration result is ignored. The RapidIO priorities (CRF and 2-bit priority field in the packet's header) of the three requestors (mentioned above) are compared. The packet with the highest RapidIO priority is chosen as the winner. In the case of a tie, round robin arbitration, based on past history, is used to select the winner.
  - Weighted round robin-if selected, the strict priority arbitration result is ignored. Each of the three requestors is considered based on its corresponding weight, and a winner is chosen.
- It is configurable with .

21.6.6.2.2.3 Arbitration Point 2

Arbitration point 2 behaves the same as Arbitration Point 1, except it:

- Chooses the Tx message unit request packet winner from Arbitration Point 1
- Is configurable with

21.6.7 RapidIO Outbound ATMU

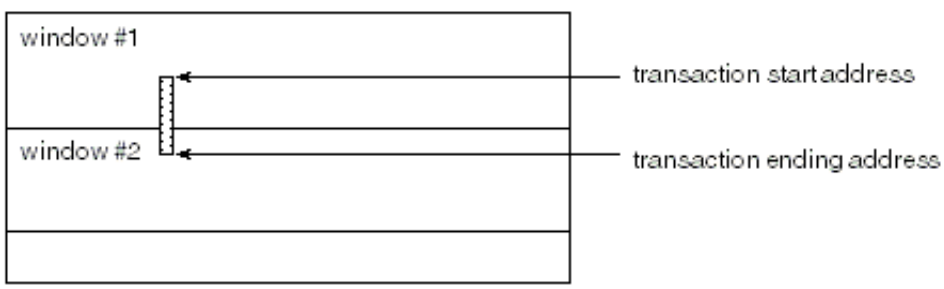
21.6.7.1 Valid Hits to Multiple ATMU Windows

If a request hits multiple ATMU windows, window 1 has the highest priority of the nine outbound ATMU windows (windows 1-8, default). Window 2 is given the next higher priority and is followed by windows 3 through 8. The default window has the lowest priority.

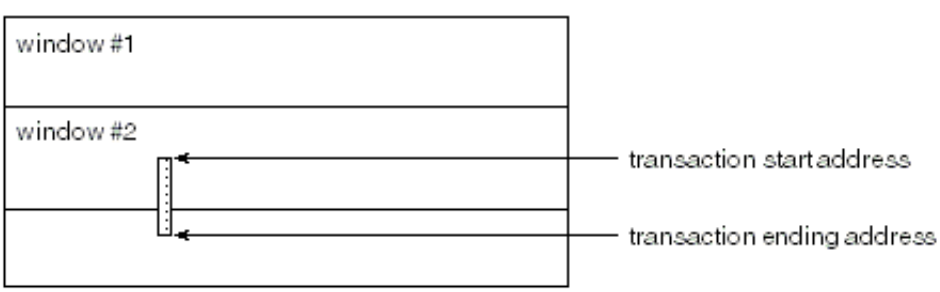
If a request hits (base address match) multiple ATMU windows and the transaction's end address is contained within the boundary of each hit window and does not extend into another ATMU window, the translation window is the highest priority window that is hit.

If a lower priority window is programmed to lie entirely within a higher priority window, then it is possible for a transaction to cross window boundaries. Although not a practical programming application, the RapidIO endpoint handles this as follows:

- If a request hits (base address match) an ATMU window (1-8) and the transaction's end address extends into another ATMU window with lower priority but is still contained within the boundary of the hit window, the translation window is the hit window.



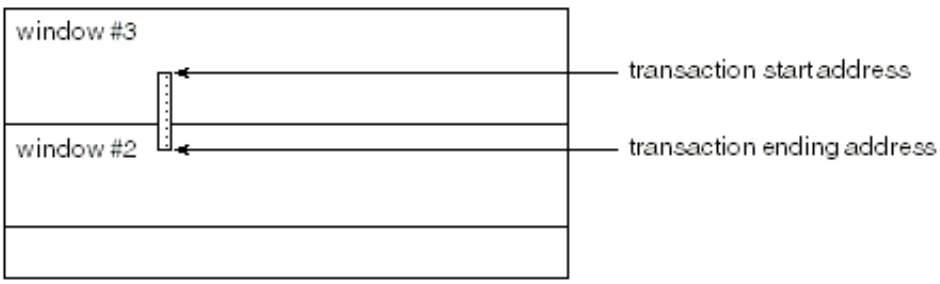
- If a request hits (base address match) multiple ATMU windows (1-8) and the transaction's end address extends beyond the boundary of a lower priority hit window but is still contained within the boundary of a higher priority hit window, the translation window is the highest priority window that is hit.



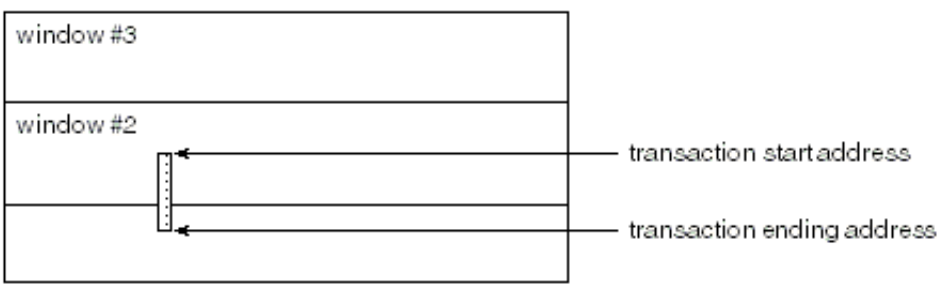
21.6.7.2 Window Boundary Crossing Errors-Outbound

If a higher priority window is programmed to lie entirely within a lower priority window, then it is possible for a transaction to cross window boundaries. The RapidIO endpoint handles this as follows:

- If a request hits (base address match) an ATMU window (1-8, default) and the transaction's end address extends into another ATMU window with higher priority, an ATMU crossed boundary error is generated and logged. The outbound request is discarded.

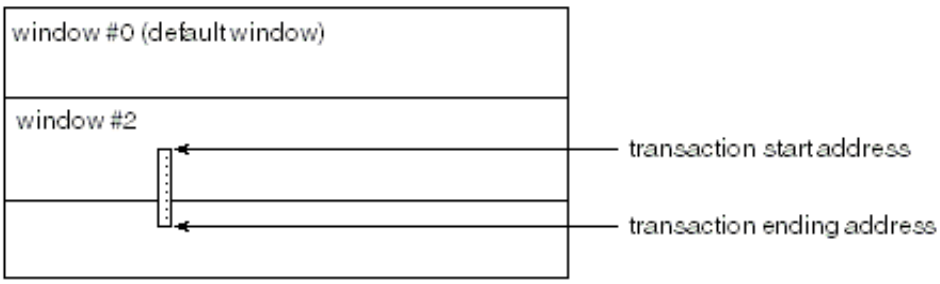


- If a request hits multiple ATMU windows (1-8, default) and transaction's end address extends beyond the boundary of a higher priority hit window, an outbound ATMU crossed boundary error is generated and logged. The outbound request is discarded.



Other window boundary crossing errors are:

- If a request hits (base address match) an ATMU window (1-8) and the transaction's end address exceeds the size of the window, an outbound ATMU crossed boundary error is generated and logged. The outbound request is discarded.



Boundary crossing errors (outbound ATMU boundary crossing, segment boundary crossing, and subsegment boundary crossing) are logged in the LTLEDCSR[OACB] configuration register field.

21.6.8 RapidIO Inbound ATMU

21.6.8.1 Hits to Multiple ATMU Windows

If a request hits multiple ATMU windows, window 1 has the highest priority of the five inbound ATMU windows (windows 1-4, default). Window 2 is given the next higher priority and is followed by windows 3 and 4. The default window has the lowest priority.

If a request hits (base address match) multiple ATMU windows and the transaction's end address is contained within the boundary of each hit window and does not extend into another ATMU window, the translation window is the highest priority window that is hit.

If a lower priority window is programmed to lie entirely within a higher priority window, then it is possible for a transaction to cross window boundaries. Although not a practical programming application, the RapidIO endpoint handles this as follows:

1. If a request hits (base address match) an ATMU window (1-4) and the transaction's end address extends into another ATMU window with lower priority but is still contained within the boundary of the hit window, the translation window is the hit window.
2. If a request hits (base address match) multiple ATMU windows (1-4) and transaction's end address extends beyond the boundary of a lower priority hit window but is still contained within the boundary of a higher priority hit window, the translation window is the highest priority window that is hit.

21.6.8.2 Window Boundary Crossing Errors-Inbound

If a higher priority window is programmed to lie entirely within a lower priority window, then it is possible for a transaction to cross window boundaries. The RapidIO endpoint handles this as follows:

1. If a request hits (base address match) an ATMU window (1-4, default) and the transaction's end address extends into another ATMU window with higher priority, an ATMU crossed boundary error is generated and logged.
2. If a request hits multiple ATMU windows (1-4, default) and transaction's end address extends beyond the boundary of a higher priority hit window, an inbound ATMU crossed boundary error is generated and logged.

Other window boundary crossing errors are:

1. If a request hits (base address match) an ATMU window (1-4) and the transaction's end address exceeds the size of the window, an inbound ATMU crossed boundary error is generated and logged.
2. If a NREAD/NWRITE_R/NWRITE/SWRITE request hits (base address match) an ATMU window (1-4, default) and the transaction's end address extends into the region defined as the local configuration space window, an inbound ATMU crossed boundary error is generated and logged.

A RapidIO error response is generated for RapidIO requests that require a response. RapidIO requests that do not require a response are dropped.

Inbound ATMU boundary crossing errors are logged in the LTLEDCSR[IACB] configuration register.

21.6.9 Inbound LIODN Translation

Inbound LIODN translation generates an internal LIODN value from an inbound request packet’s source ID. Inbound LIODN translation does not occur for inbound message type requests, pass-through packets and inbound responses; they do not generate an internal LIODN value. A base LIODN value, contained in SRIOPn_LBR, is added to a LIODN offset, contained in a matching entry of the LIODN lookup table, to create an LIODN value for the inbound request packet.

21.6.9.1 LIODN Lookup table

The LIODN lookup table is shown below.

Table 21-265. LIODN Lookup table

	Upper		Lower	
	SRCID	MASK	LOFFSET	EN
Entry 0	0x0000	0xFFFF	0x000	0x1
Entry 1	0x0000	0x0000	0x000	0x0
Entry 2	0x0000	0x0000	0x000	0x0
*				
*				

The default state of the LIODN lookup table will ensure a match to entry#0 and is defined as the following:

Entry 0

SRCID == 0x0000 MASK == 0xFFFF LOFFSET == 0x000 EN == 0x1.

Entry 1-63

SRCID == 0x0000 MASK == 0x0000 LOFFSET == 0x000 EN == 0x0.

21.6.9.1.1 Accessing the LIODN Lookup table

Software can program the LIODN lookup table using the following method:

1. Write the desired entry number into the SRIOPn_LAOR[ADDR] field.
2. Write the desired values for the SRCID and MASK fields within the SRIOPn_LUDR register.
3. Write the desired values for the LOFFSET and EN fields within the SRIOPn_LLDR register.
4. Repeat for other entries.

Each write to the SRIOPn_LUDR or SRIOPn_LLDR registers will initiate a write to the LIODN lookup table at the entry designated by SRIOPn_LAOR. Each read from the SRIOPn_LUDR or SRIOPn_LLDR registers will return data from the LIODN lookup table of the entry designated by SRIOPn_LAOR.

21.6.9.1.2 Translation

The following steps are used to determine an LIODN lookup table match:

1. An inbound request packet's source ID is AND'ed with each entry's inverse MASK field.
2. Each entry's SRCID field is AND'ed with the inverse MASK field.

3. The results of items #1 and #2 are AND'ed with the entry's EN field and compared to determine if a match has been detected.
4. A winning match will be determined by a valid match for that entry and no lower entry matches exist as well. For example multiple matches to entries #2, #6 and #24 will yield a winning match of entry #2.
5. The resulting LIODN field is calculated from the following:

   {4'b0000, SRIOPn_LBR[LBASE]} + {6'b000000, winning entry's LOFFSET field}

An inbound request that does not match any entry of the LIODN lookup table will be considered a logical inbound LIODN error and logged in the LTLEDCSR[ILE] error bit, along with any error capture information. Inbound requests requiring responses will return an ERROR response for a LIODN lookup table miss while inbound requests not requiring responses will be dropped. See hardware error tables in Logical Layer RapidIO Errors Detected Logical Layer RapidIO Errors Detected for more details.

21.6.10 Generating Link-Request/Reset-Device

21.6.10.1 LP-Serial Mode

In LP-Serial mode the link partner cannot be reliably reset using the link-request/reset-device control symbols since the input port receiver cannot be disabled independent of the output port driver. The input port driver needs to be disabled to prevent non idle control symbols from being transmitted between the four link-request/reset-device control symbols. For example, if a packet was received on the inbound side after one of the four link-request/reset-device control symbols was sent outbound, the required sequence of four link-request/reset-device symbols would be interrupted with the packet acknowledgement (packet accept, packet retry or packet not accept).

21.6.11 Outbound Drain Mode

- The RapidIO port is placed into Drain mode when one of the following occurs:
  - PnPCR[OBDEN] is set
  - the Failed Threshold has been encountered and the PnCCSR[SPF] and PnCCSR[DPE] are both set
  - the packet time-to-live counter expires causing P*n*PCR[OBDEN] to be set
- When the RapidIO port is placed into Drain mode, the RapidIO port discards all packets in the outgoing data stream. Since the data stream is invalid, the RapidIO port also puts its Outbound port back to normal state. Any received

acknowledgements and link-responses are considered invalid during this period (since the RapidIO port has cleared out all acknowledgement history).
- The RapidIO port's outbound and outstanding ackID shows that all outstanding packets at the time Drain mode was entered were accepted, whether they truly were accepted or not. If the outbound ackID is not acceptable, then software should change it prior to taking the RapidIO port out of the Drain mode. Also, if the link-partner needs to be put back into inbound OK state, then software should send a link-request/input-status. The recommended sequence for recovering from Drain mode is given in Software Assisted Error Recovery Register Support.
- PnPCR[OBDEN] also causes any queued up packet acknowledgements to be discarded if the port is uninitialized (the RapidIO port lets them be transferred if the port is initialized). Drain mode due to failed threshold does not cause any packet acknowledgements to be dropped.
- If a discarded packet in the outgoing data stream requires a logical response, a packet response time-out occurs as long as the packet response timer is enabled (PRTOCCSR is non 0).

21.6.12 Input Port Disable Mode

- The RapidIO port is placed into Input Port Disable mode when PnCCSR[PD] is set.
- When the RapidIO port is placed in Input Port Disable mode, the RapidIO port discards all incoming data stream (obviously). Since the incoming stream is invalid, the RapidIO port also puts its inbound port back to normal state.
- When the RapidIO port is placed in input port disable mode, the RapidIO port also:
  - Ends any packet capture that was in progress.
  - Clears the link-request/reset-device count.
- The RapidIO port's inbound ackID shows that all packets successfully received by the RapidIO port at the time input port disable mode was entered were accepted. If output port disable mode was entered at the same time, some packet acknowledgements may be discarded by the RapidIO port. If the inbound ackID is not acceptable, then software should change it prior to taking the RapidIO port out of Input Port Disable mode.

21.6.13 Software Assisted Error Recovery Register Support

P*n*LMREQCSR is only supported for recovery from drain mode, including hot-swap support. Consistent with this statement, software should only write this register when the port is in Drain mode.

The proper sequence for recovering from Drain mode is as follows:

1. Software ensures that link activity is stopped. This should include the following:
   a. Software polls for Port OK bit to be set.
   b. Software waits longer than the link time-out value.
2. Software generates a link-request/input-status to obtain the link-partner's inbound ackID value.
3. Software changes the RapidIO port's outbound ackID to this value (if necessary).
4. If the link-partner was hot-inserted, software changes the RapidIO port's inbound ackID to zero. (Note that software needs to know when the link-partner was hot-inserted.)

NOTE

If software can guarantee that the link-partner does not attempt to forward any packets to this RapidIO port, then software may write the outbound ackID of the link-partner to match the inbound ackID of this RapidIO port. However, if the link-partner attempts to forward another packet while this write is still outstanding, and the ackIDs already happened to line-up, then the write actually causes the ackIDs to not match, and the link can not be recovered.

5. Software should cause the link-partner to send a link-request/input-status just to ensure that the RapidIO port's inbound port is in Normal operation.
6. Software clears the Failed Encountered bit or the Output Buffer Drain Enable bit; whichever one caused the Drain mode (thus clearing Drain mode).

Software is responsible for timing software generated link-requests. If the response valid bit is not set in some reasonable period of time, the software should write another request in the register.

When software writes P*n*LMREQCSR, software should make sure to successfully read P*n*LMRESPCSR set, otherwise, software may read a "stale" ackID status/link status later.

Note that when the RapidIO port's outbound ackID is written by software using P*n*LASCSR, the inbound ackID is also written. Care must be taken to ensure that the inbound ackID is not written to an incorrect value. For example, P*n*CCSR[PL] could be set to prevent the inbound ackID from changing before P*n*LASCSR is written.

21.6.14 Hot-Swap Support

The two basic hot-insertion approaches described in the RapidIO Error Management Extensions are supported although the second approach is only partially supported. Method one has a host bringing a field replacement unit (FRU) into the system. Method two has the FRU bringing itself into the system. Note that this RapidIO port is most

likely the device being hot-inserted/extracted since typically this device would be connected to a switch but it could be the link partner of a device being hot-inserted/extracted.

21.6.14.1 Method 1

One possible sequence for when the host brings the FRU into the system is given. This RapidIO port can be either the device being hot-inserted/extracted or the link partner of the device being hot-inserted/extracted. The goal of this sequence is to bring the FRU into the system cleanly without generating errors.

21.6.14.1.1 Extraction Sequence (Method 1)

The extraction sequence using method 1 is as follows:

1. The host determines that the FRU has failed by getting a port response time-out when polling the port OK bit (PO bit in port n error and status CSR) of the FRU. Note that there are other ways to determine that the FRU has failed. This is one possible method to detect FRU failure.
2. The host must perform the following steps to the FRU link partner before hot insertion of the FRU occurs.
   a. Set the port lockout bit (PL = 1 in the port *n*control CSR) preventing packet reception and transmission
   b. Prevent any new packets from arriving to the FRU link partner RapidIO port by all other RapidIO devices and discard all pending packets destined for the FRU so that:
      - Congestion to this RapidIO port does not occur and cause other system problems
      - The FRU is not immediately flooded with old packets after insertion causing other errors like unsolicited responses
   c. Note that this RapidIO endpoint has a specific bit to enable the discard of pending packets (OBDEN in PnPCR). Also note that setting OBDEN in P*n*PCR forces the output state from error or retry to normal. In a switch, the time-to-live feature may be used to discard packets.
   d. Force the input state to normal. In this RapidIO endpoint, this occurs by disabling the input port receiver.
   e. Leave the input port receivers and output port drivers enabled so that initialization can complete when the FRU is inserted
   f. Clear all errors pertaining to the extracted RapidIO port in the FRU link partner and any other RapidIO port that was affected by the FRU failing (port response time-outs)

g. If RapidIO endpoint, clear OBDEN in P*n*PCR for normal operation
h. Set the outbound and inbound ackID to 0x00 in the port *n*local ackID status CSR so that the ackID is correct and packets can be transmitted and received when the FRU is inserted

3. The host indicates that the FRU should be removed.
4. The FRU is removed from the system.

21.6.14.1.2 Insertion Sequence (Method 1)

The insertion sequence using method 1 is as follows:

1. The FRU is inserted into the system.
2. Link initialization occurs and initialization complete status is indicated (port OK bit in the port n error and status CSR) in both RapidIO ports.
3. The host determines that this link partner has been inserted by periodically polling the initialization complete status in the FRU link partner (PO bit = 1 in the port n error and status command and status register).
4. The host clears the output and input port enable bits and clears the port lockout bit in the port n control CSR in the FRU link partner allowing only maintenance transactions.
5. The host sets the master enabled and discovered bits in the FRU (M = 1 and D = 1 in the port general control CSR). Note that the master enable bit does not prevent the RapidIO endpoint from transmitting packets.
6. The host completes configuration of the FRU
7. The host sets the output and input port enable bits in the port n control CSR in the FRU link partner allowing transmission and reception of all packet types.
8. The host re-enables packets to be sent to this RapidIO port by other RapidIO devices.
9. System operation resumes.

21.6.14.2 Method 2 with RapidIO Port Hot-Swapped

One possible sequence for when the FRU brings itself into the system is given. This RapidIO Port can be either the device being hot-inserted/extracted or the link partner of the device being hot-inserted/extracted. The goal of this sequence is to bring the FRU into the system cleanly without generating errors.

21.6.14.2.1 Extraction Sequence (Method 2)

The extraction sequence using method 2 is as follows:

1. The FRU fails.

2. New packets are prevented from arriving to the FRU link partner by all other RapidIO devices and all pending packets destined for the FRU are discarded so that the following occur:
   a. Congestion to this RapidIO port does not occur and cause other system problems.
   b. The FRU is not immediately flooded with old packets after insertion causing other errors such as unsolicited responses.
3. This FRU link partner continues to have its drivers enabled.
4. The FRU link partner continues to allow the transmission and reception of all packet types since its output and input port enable bits are set and the port lockout bit is cleared in the port n control CSR.
5. The FRU is removed from the system.

21.6.14.2.2 Insertion Sequence (Method 2)

The insertion sequence using method 2 is as follows:

1. The FRU is inserted.
2. Link initialization occurs, initialization complete status is indicated in both RapidIO ports (PO bit = 1 in the port n error and status command and status register).
3. After initialization is complete, both devices set the port OK bit = 1 in the port n control CSR.
4. The FRU sets its port lockout bit in the port n control CSR.
5. The FRU generates a link-request/input-status to its link partner using the port n link maintenance request register. The FRU determines the link partner's inbound ackID by reading the port n link maintenance response register. The FRU then sets its outbound ackID in the port n local ackID status CSR.
6. The FRU only enables maintenance transactions by clearing its output and input port enable bits in the port n control CSR and by clearing its port lockout bit in the port n control CSR.
7. The FRU generates a maintenance write to its link partner's port n local ackID status CSR to set the link partner's outbound ackID value to 0. Upon receipt of the maintenance write, the link partner sets its outbound ackID value and generates the maintenance response using the new value. Note that if all packets intended for the link partner have not been discarded, or if any new packets intended for the link partner arrive, this step may fail (the RapidIO endpoint has no way to protect against this).
8. The FRU completes configuration.
9. The FRU enables all packets to be transmitted and received by setting its output and input port enable bits in the port n control CSR.
10. System operation resumes.

21.6.15 Software Re-Training

You can issue a software controlled re-training of the link in LP-Serial mode to update certain configurations by following these steps:

1. Software on the host must ensure that all RapidIO transactions have completed and link activity has stopped.
2. Set PnCCSR[PD] on the host to disable the port and force the initialization state machines to reset.
3. Set P*n*PCR[OBDEN] on the host to enable the discarding of any pending packets. (There should be none if proper steps were taken to ensure there was no link activity.)
4. Clear P*n*PCR[OBDEN] on the host.
5. Configure new operating width (via PnCCSR[PWO]) or any other new configurations.
6. Clear PnCCSR[PD] on the host to re-enable the drivers.
7. Poll PnESCSR[PO] on the host until it is set, which indicates the link has attained port and link initialization.
8. Poll PnESCSR[PO] on the agent until it is set, which indicates the link has attained port and link initialization.
9. Poll PnESCSR[OES] on the agent until it is clear, which indicates the link has completed the error recovery sequence initiated from the port disable.
10. Poll PnESCSR[OES] on the host until it is clear, which indicates the link has completed the error recovery sequence initiated from the port disable.
11. Clear the agent's error and status registers.
12. Clear the host's error and status registers.
13. Begin normal packet transfer.

21.6.16 Errors and Error Handling

This section describes how the logical and physical layers detect and react to RapidIO errors. The action of the core on notification of any of these errors is described minimally here. See *RapidIO Interconnect Specification, Revision 1.2* Part VII (Error Management Extensions Specifications) for more details on specific errors described below.

21.6.16.1 RapidIO Error Description

RapidIO errors are classified under three categories: recoverable errors, notification errors, and fatal errors.

Recoverable errors are non-fatal transmission errors (such as corrupt packet or control symbols, and general protocol errors) that RapidIO supports hardware detection of and a recovery mechanism for, as described in the *RapidIO Interconnect Specification, Revision 1.2*. In these cases, the appropriate bit is set in the port n error detect CSR. Only the packet containing the first detected recoverable error that is enabled for error capture (by port n error enable CSR) is captured in the port n error capture CSRs. No interrupt is generated or actions required for a recoverable error. Recoverable errors are detected in the physical layer only.

Notification errors are non-recoverable non-fatal errors detected by RapidIO (such as degraded threshold, port-write received, and all logical/transport layer (LTL) errors captured). Because they are non-recoverable (and in some cases have caused a packet to be dropped), notification by interrupt is available. However, because they are non-fatal, response to the interrupt is not crucial to port performance; that is, the port is still functional. When a notification error is detected, the appropriate bit is set in the error-specific register, an interrupt is generated, and in some cases, the error is captured. In all cases, the RapidIO port continues operating. Notification errors are detected in both the physical and logical layer. Note that to prevent processor stalls the P*n*LOPTTLCR registers must be initialized to a non-zero value.)

The RapidIO controller detects two fatal errors: exceeded failed threshold and exceeded consecutive retry threshold. In these cases, the port has failed because its recoverable error rate has exceeded a predefined failed threshold or because it has received too many packet retries in a row. In the first case, the controller sets the output failed-encountered bit in the port n error and status CSR; the RapidIO output hardware may or may not stop (based on stop-on-port-failed-encounter-enable and drop-packet-enable bits). In the second case, the controller sets the retry counter threshold trigger exceeded bit in the port n implementation error CSR; the RapidIO hardware continues to operate. In both cases, an interrupt is generated, and while the port continues operating at least partially, a system-level fix (such as reset) is recommended to clean up the controller's internal queues and resume normal operation. Fatal errors are detected in the physical layer only.

21.6.16.2 Physical Layer RapidIO Errors Detected

This table lists all the RapidIO link errors detected by the RapidIO endpoint physical layer and the actions taken by the RapidIO endpoint. The error enable column lists the control bits that may disable the error checking associated with a particular error (if blank, error checking cannot be disabled). The cause field column indicates what cause field is used with the associated packet-not-accept control symbol for input error recovery. The EME error enable/detect column indicates which bit of the P*n*ERECSR allows the error to increment the error rate counter and lock the port n error capture registers, and likewise which bit of the P*n*EDCSR is set when the error has been detected.

Table 21-267 lists the RapidIO endpoint behavior after exceeding certain preset limits.

Table 21-266. Physical RapidIO Errors Detected

Level	Error	Error Enable	RapidIO Endpoint Action	Cause Field	EME Error Type	EME Error Enable / Detect
Recoverable Errors						
1a	Received character had a disparity error.	-	Enter Input Error Stopped. Enter Output Error Stopped.	5: Received invalid/illegal character	Delineation Error	DE
1a	Received an invalid character, or valid but illegal character	-	Enter Input Error Stopped. Enter Output Error Stopped.	5: Received invalid/illegal character		
1b	The four control character bits associated with the received symbol do not make sense (not 0000, 1000, 1111)	-	Enter Input Error Stopped. Enter Output Error Stopped.	5: Received invalid/illegal character		
1b	Control symbol does not begin with an /SC/ or /PD/ control character.	-	Enter Input Error Stopped. Enter Output Error Stopped.	5: Received invalid/illegal character		
1c	Received packet with embedded idles.	-	Enter Input Error Stopped.	5: Received invalid/illegal character		
1d	Received a control symbol with a bad CRC	PCR[CCC]enables detect.	Enter Input Error Stopped. Enter Output Error Stopped.	2: Received a control symbol with bad CRC	Received corrupt control symbol	CCS
1d	Missing start: Packet data received w/o previous SOP control symbol	-	Enter Input Error Stopped.	7/31: General error	Protocol Error (unexpected packet/control symbol received)	PE
1e	Received packet that is < 64 bits	-	Enter Input Error Stopped.	7/31: General error		
1e	Received an EOP control symbol when there is no packet being received.	-	Enter Input Error Stopped.	7/31: General error		
1e	Received a stomp control symbol when there is no packet being received.	-	Enter Input Error Stopped.	7/31: General error		
2a	Received a Restart-from-retry control symbol when in the "OK" state	-	Enter Input Error Stopped	7/31: General error	Protocol Error (unexpected packet/control symbol received)	PE

Table continues on the next page...

Table 21-266. Physical RapidIO Errors Detected (continued)

Level	Error	Error Enable	RapidIO Endpoint Action	Cause Field	EME Error Type	EME Error Enable / Detect
2a	Received packet with a bad CRC value.	PCR[CCP] enables detect.	Enter Input Error Stopped.	4: bad CRC on packet.	Received packet with bad CRC	CRC
2a	Received packet which exceeds the maximum allowed size by the RapidIO spec.	-	Enter Input Error Stopped.	7/31: General error	Received packet exceeds 276 Bytes	EM
2b	Received packet with unexpected ackID value (out-of-sequence ACKID)	-	Enter Input Error Stopped.	1: Received unexpected ACKID on packet	Received packet with unexpected ackID	UA
2c	Received a non-maintenance packet when non-maintenance packet reception is stopped	Non-maint. packet reception is stopped when 'Input Port Enable' = 0.	Enter Input Error Stopped.	3: Non-maintenance packet reception is stopped	Not Captured	-
2d	Any packet received while Port Lockout bit is set	All packet reception is stopped when Port Lockout bit is set.	Enter Input Error Stopped.	3: Non-maintenance packet reception is stopped	Not Captured	-
-	Received a Link request control symbol before servicing previous link request.	Not detected.				
2a	Received packet-not-accepted ACK control symbol.	-	Enter Output Error Stopped.	-	Received packet-not-accepted symbol	PNA
2b	Received an ACK (accepted, or retry) control symbol when there are no outstanding packets	-	Enter Output Error Stopped.	-	Unsolicited ACK symbol	UCS
2b	Received packet ACK (accepted) for a packet whose transmission has not finished	-	Enter Output Error Stopped.	-		
2b	Received a Link response control symbol when no outstanding request.	-	Enter Output Error Stopped.	-		
2c	Received an ACK (accepted or retry) control symbol with an unexpected ACKID.	-	Enter Output Error Stopped.	-	Received ack. control symbol with unexpected ackID	AUA
2c	Link_response received with an ackID that is not outstanding	-	Re-enter Output Error Stopped.	-	Non-outstanding ackID	NOA

Table continues on the next page...

Table 21-266. Physical RapidIO Errors Detected (continued)

Level	Error	Error Enable	RapidIO Endpoint Action	Cause Field	EME Error Type	EME Error Enable / Detect
2d	An ACK control symbol is not received within the specified time-out interval.	PLTOCCSR[TV] > 0 enables detect.	Enter Output Error Stopped.	-	Link time-out	LTO
2d	A Link response is not received within the specified time-out interval	PLTOCCSR[TV] > 0 enables detect.	(re-) Enter Output Error Stopped.	-		

This table lists the RapidIO endpoint behavior after exceeding certain preset limits (degraded threshold, failed threshold, retry threshold).

Table 21-267. Physical RapidIO Threshold Response

Error	Error Enable	RapidIO Endpoint Action	EME Error Type	Error Detect	Interrupt clear[1]
Notification Errors					
Error Rate Counter has met or exceeded the Degraded Threshold.	ERTCSR[ERDTT] > 0 & any bit in EECSR enables detect and interrupt generation.	Generate Interrupt. Continue to operate normally.	Degraded Threshold	ESCSR[ODE]	Write 1 to ESCSR[ODE]
Fatal Errors					
Consecutive Retry Counter has met or exceeded the Retry Counter Threshold Trigger	PRETCR[RET] > 0 enables detect and interrupt generation	Generate Interrupt. Port is in priority order.	Consecutive Retry Threshold	IECSR[RETE]	Write 1 to IECSR[RETE]
Error Rate Counter has met or exceeded the Failed Threshold.	ERTCSR[ERFTT] > 0 & any bit in EECSR enables detect and interrupt generation.	Generate Interrupt. Port behavior depends on CCSR[SPF] and CCSR[DPE] -- port can continue transmitting packets or can stop sending output packets, keeping or dropping them.	Failed Threshold	ESCSR[OFE]	Write 1 to ESCSR[OFE] .

1. Information given here is minimal for clearing the interrupt. More detailed steps should be taken to find the cause of the interrupt.

21.6.16.3 Logical Layer Errors and Error Handling

This section describes how the logical layer detects and reacts to RapidIO errors. The action of the core on notification of any of these errors is described minimally here. Reference *RapidIO Interconnect Specification, Revision 1.2* Part VII (Error Management Extensions Specifications). Further details of message related errors are detailed in the RMan error management section for each each type.

21.6.16.3.1 Logical Layer RapidIO Errors Detected

The following tables list all the errors detected by the RapidIO endpoint logical layer and the actions taken by the RapidIO endpoint. Note that when the RapidIO endpoint action includes sending an error response to either the internal platform or RapidIO, an error response is only sent if the original transaction was a request that required a response. Otherwise, no error response is sent. When dealing with multiple errors, discard of packet has higher priority than error response.

For misaligned transactions, the error management extension registers are updated with each child.

All packet field positions are assumed to be in the mode (small or large transport) configured. For example, when configured for small transport mode with pass-through mode not enabled and a large transport mode NREAD packet is received, the transaction type field bit positions checked correspond to a small transport type NREAD packet.

Table 21-268. Hardware Errors For NRead Transaction

Error	Interrupt Generated	Status Bit Set	RapidIO Error Response Generated	Logical/Transport Layer Capture Register	Comments
Priority Priority of Read transaction is 3	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	No	Using the incoming RapidIO packet, for Small Transport type packet, LTLACCSR[XA] gets packet bits 78-79, LTLACCSR[A] gets packet bits 48-76, LTLDIDCCSR[DIDMSB] gets 0's, LTLDIDCCSR[DID] gets packet bits 16-23, LTLDIDCCSR[SIDMSB] gets 0's, LTLDIDCCSR[SID] gets packet bits 24-31, LTLCCCSR[FT] gets	RapidIO packet is dropped.

Table continues on the next page...

Table 21-268. Hardware Errors For NRead Transaction (continued)

Error	Interrupt Generated	Status Bit Set	RapidIO Error Response Generated	Logical/Transport Layer Capture Register	Comments
				packet bits 12-15, LTLCCCSR[TT] gets packet bits 32-35, LTLCCCSR[MI] gets 0's For Large Transport type packets r. LTLACCSR[XA] gets packet bits 94-95, LTLACCSR[A] gets packet bits 64-92, LTLTLTLDIDCCSR[DIDMSB] gets 16-23, LTLDIDCCSR[DID] LTL gets packet bits 24-31, LTLDIDCCSR[SIDMSB] gets bits 32-39, LTLDIDCCSR[SID] gets bits 40-47, LTLCCCSR[FT] gets packet bits 12-15, LTLCCCSR[TT] gets packet bits 48-51, LTLCCCSR[MI] gets 0's	
Critical Request Flow Not Checked for error.	-	-	-	-	-
TransportType Received reserved TT	Yes if LTLEECSR[TSE] is set	LTLEDCSR[TSE]	No	Same as first entry	RapidIO packet is dropped
Received TT which is not enabled. - Error valid when pass_through is disabled and accept_all is disabled Or when accept_all is enabled.	Yes if LTLEECSR[TSE] is set	LTLEDCSR[TSE]	No	Same as first entry	RapidIO packet is dropped
DestID DestID does not match this port's DeviceID if Alternate DeviceID is disabled or DestId does not match either Alternate DeviceID or DeviceId if Alternate DeviceID is enabled. Error valid when (pass_through or accept_all) is false	Yes if LTLEECSR[ITTE] is set	LTLEDCSR[ITTE]	Yes	Same as first entry	Internal platform error response is generated to self
SourceID Not Checked for error.	-	-	-	-	-
TransactionType	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	Yes	Same as first entry	Internal platform error

Table continues on the next page...

Table 21-268. Hardware Errors For NRead Transaction (continued)

Error	Interrupt Generated	Status Bit Set	RapidIO Error Response Generated	Logical/Transport Layer Capture Register	Comments
Received RapidIO packet with reserved TType for this ftype					response is generated to self
RdSize Not Checked for error.	-	-	-	-	-
SrcTID Not Checked for error.	-	-	-	-	-
Address:WdPtr:Xambs Read request hits overlapping ATMU windows Refer to Window Boundary Crossing Errors-Inbound	Yes if LTLEECSR[IACB] is set	LTLEDCSR[IACB]	Yes	Same as first entry	Internal platform error response is generated to self
Address:WdPtr:Xambs Request hits a protected ATMU window	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	Yes	Same as first entry	Internal platform error response is generated to self
Address:WdPtr:Xambs Beginning address matches LCSBA1CSR with non 32 bit read request. Performed only when ttype == 4'b0100	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	Yes	Same as first entry	Internal platform error response is generated to self
Header Size Header size is not 12 Bytes for small Transport packet or not 16 Bytes for Large Transport packet. Large Transport packet has 14 valid bytes and two bytes of padding of 0's. Padding of 0's is not checked.	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	Yes	Same as first entry	Internal platform error response is generated to self
PayloadSize Not Applicable	-	-	-	-	-
LIODN: Lookup table miss	Yes if LTLEECSR[ILE] is set	LTLEDCSR[ILE]	Yes	Same as first entry	Internal platform error response is generated to self

Table 21-269. Hardware Errors For Maintenance Read/Write Req Transaction

Error	Interrupt Generated	Status Bit Set	RapidIO Error Response Generated	Logical/Transport Layer Capture Register	Comments
Priority Priority of maintenance read or write request transaction is 3	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	No	Using the incoming RapidIO packet, for Small Transport type packets, LTLACCSR[XA] gets packet bits 78-79, LTLACCSR[A] gets packet bits 48-76, LTLDIDCCSR[DIDMSB] gets 0's, LTLDIDCCSR[DID] gets packet bits 16-23, LTLDIDCCSR[SIDMSB] gets 0's, LTLDIDCCSR[SID] gets packet bits 24-31, LTLCCCSR[FT] gets packet bits 12-15, LTLCCCSR[TT] gets packet bits 32-35, LTLCCCSR[MI] gets 0's For Large Transport type packets. LTLACCSR[XA] gets packet bits 94-95, LTLACCSR[A] gets packet bits 64-92, LTLTLTLDIDCCSR[DIDMSB] gets 16-23, LTLDIDCCSR[DID] LTL gets packet bits 24-31, LTLDIDCCSR[SIDMSB] gets bits 32-39, LTLDIDCCSR[SID] gets bits 40-47, LTLCCCSR[FT] gets packet bits 12-15, LTLCCCSR[TT] gets packet bits 48-51, LTLCCCSR[MI] gets 0's	RapidIO packet is dropped
Critical Request Flow Not Checked for error.	-	-	-	-	-
TransportType Received reserved TT	Yes	LTLEDCSR[TSE]	No	Same as first entry	RapidIO packet is dropped

Table continues on the next page...

Table 21-269. Hardware Errors For Maintenance Read/Write Req Transaction (continued)

Error	Interrupt Generated	Status Bit Set	RapidIO Error Response Generated	Logical/Transport Layer Capture Register	Comments
Received TT which is not enabled. - Error valid when passthrough is disabled and accept_all is disabled Or when accept_all is enabled.	Yes if LTLEECSR[TSE] is set	LTLEDCSR[TSE]	No	Same as first entry	RapidIO packet is dropped
DestID DestID does not match this port's DeviceID if Alternate DeviceID is disabled or DestId does not match either Alternate DeviceID or DeviceId if Alternate DeviceID is enabled. Error valid when (passthrough or accept_all) is false	Yes if LTLEECSR[ITTE] is set	LTLEDCSR[ITTE]	Yes	Same as first entry	Internal platform error response is generated to self
SourceID Not Checked for error.	-	-	-	-	-
TransactionType Reserved Transaction Type for this ftype	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	Yes	Same as first entry	RapidIO packet is dropped
RdSize Read/Write request size is not for 4 bytes	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	Yes	Same as first entry	Internal platform error response is generated to self
SrcTID Not Checked for error.	-	-	-	-	-
HopCount Not Checked for error.	-	-	-	-	-
Config Offset Not Checked for error.	-	-	-	-	-
Header Size Maintenance Read request - Header size is not 12 Bytes for small Transport packet or not 16 Bytes for Large Transport packet. Maintenance Write request - total header size is not 12 Bytes for Small Transport packet or not 16 Bytes for Large Transport packet. Padding of 0's in last two bytes of Large Transport packet is not checked.	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	Yes	Same as first entry	Internal platform error response is generated to self
PayloadSize Write request with payload not equal to 8 bytes. Read request with payload not 0 bytes	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	Yes	Same as first entry	Internal platform error

Table continues on the next page...

Table 21-269. Hardware Errors For Maintenance Read/Write Req Transaction (continued)

Error	Interrupt Generated	Status Bit Set	RapidIO Error Response Generated	Logical/Transport Layer Capture Register	Comments
					response is generated to self
LIODN: Lookup table miss	Yes if LTLEECSR[ILE] is set	LTLEDCSR[ILE]	Yes	Same as first entry	Internal platform error response is generated to self

Table 21-270. Hardware Errors For Atomic (inc, dec, set, or clr) Read Transaction

Error	Interrupt Generated	Status Bit Set if corresponding bit is enabled	RapidIO Error Response Generated	Logical/Transport Layer Capture Register	Comments
Priority Priority of read transaction is 3	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	No	Using the incoming RapidIO packet, for Small Transport type packets, LTLACCSR[XA] gets packet bits 78-79, LTLACCSR[A] gets packet bits 48-76, LTLDIDCCSR[DIDMSB] gets 0's, LTLDIDCCSR[DID] gets packet bits 16-23, LTLDIDCCSR[SIDMSB] gets 0's, LTLDIDCCSR[SID] gets packet bits 24-31, LTLCCCSR[FT] gets packet bits 12-15, LTLCCCSR[TT] gets packet bits 32-35, LTLCCCSR[MI] gets 0's For Large Transport type packets.LTLACCSR[XA] gets packet bits 94-95, LTLACCSR[A] gets packet bits 64-92,	RapidIO packet is dropped

Table continues on the next page...

Table 21-270. Hardware Errors For Atomic (inc, dec, set, or clr) Read Transaction (continued)

Error	Interrupt Generated	Status Bit Set if corresponding bit is enabled	RapidIO Error Response Generated	Logical/Transport Layer Capture Register	Comments
				LTLTLTLDIDCCSR[DID MSB] gets 16-23, LTLDIDCCSR[DID] gets packet bits 24-31, LTLDIDCCSR[SIDMSB] gets bits 32-39, LTLDIDCCSR[SID] gets bits 40-47, LTLCCCSR[FT] gets packet bits 12-15, LTLCCCSR[TT] gets packet bits 48-51, LTLCCCSR[MI] gets 0's	
Critical Request Flow Not Checked for error.	-	-	-	-	-
TransportType Received reserved TT	Yes if LTLEECSR[TSE] is set	LTLEDCSR[TSE]	No	Same as first entry	RapidIO packet is dropped
Received TT which is not enabled. - Error valid when passthrough is disabled and accept_all is disabled Or when accept_all is enabled.	Yes if LTLEECSR[TSE] is set	LTLEDCSR[TSE]	No	Same as first entry	RapidIO packet is dropped
DestID DestID does not match this port's DeviceID if Alternate DeviceID is disabled or DestId does not match either Alternate DeviceID or DeviceId if Alternate DeviceID is enabled. Error valid when (passthrough or accept_all) is false	Yes if LTLEECSR[ITTE] is set	LTLEDCSR[ITTE]	Yes	Same as first entry	Internal platform error response is generated to self
SourceID Not Checked for error.	-	-	-	-	-
TransactionType Non-Atomic ttype is tested with Nread	-	-	-	-	-
TransactionType Received Atomic Increment request with DOCAR[AI] disabled.	Yes if LTLEECSR[UT] is se	LTLEDCSR[UT]	Yes	Same as second entry	Error response is generated to self

Table continues on the next page...

Table 21-270. Hardware Errors For Atomic (inc, dec, set, or clr) Read Transaction (continued)

Error	Interrupt Generated	Status Bit Set if corresponding bit is enabled	RapidIO Error Response Generated	Logical/Transport Layer Capture Register	Comments
Received Atomic decrement request with DOCAR[AD] disabled. Received Atomic Set request with DOCAR[AS] disabled. Received Atomic Clear request with DOCAR[AC] disabled.					
RdSize Not unsupported RdSize request is not for contiguous one, two or four bytes	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	Yes	Same as first entry	Error response is generated to self
SrcTID Not Checked for error.	-	-	-	-	-
Address:WdPtr:Xambs Not unsupported Request hits a protected ATMU window or the LCSBA1CSR Refer to Window Boundary Crossing Errors-Outbound	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	Yes	Same as first entry	Error response is generated to self
Address:WdPtr:Xambs Read request hits overlapping ATMU windows Refer to Window Boundary Crossing Errors-Outbound	Yes if LTLEECSR[IACB] is set	LTLEDCSR[IACB]	Yes	Same as first entry	Error response is generated to self
Header Size Not unsupported Header size is not 12 Bytes for small Transport packet and not 16 Bytes for Large Transport packet Padding of 0's in last two bytes of Large Transport packet is not checked	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	Yes	Same as first entry	Error response is generated to self
LIODN: Lookup table miss	Yes if LTLEECSR[ILE] is set	LTLEDCSR[ILE]	Yes	Same as first entry	Internal platform error response is generated to self

Table 21-271. Hardware Errors For NWrite, NWrite_r, and Unsupported Atomic Test-and-Swap Transactions

Error	Interrupt Generated	Status Bit Set	RapidIO Error Response Generated	Logical/Transport Layer Capture Register	Comments
Priority Transaction priority is 3	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	No	Using the incoming RapidIO packet, for Small Transport type packets, LTLACCSR[XA] gets packet bits 78-79, LTLACCSR[A] gets packet bits 48-76, LTLDIDCCSR[DIDMSB] gets 0's, LTLDIDCCSR[DID] gets packet bits 16-23, LTLDIDCCSR[SIDMSB] gets 0's, LTLDIDCCSR[SID] gets packet bits 24-31, LTLCCCSR[FT] gets packet bits 12-15, LTLCCCSR[TT] gets packet bits 32-35, LTLCCCSR[MI] gets 0's For Large Transport type packets. LTLACCSR[XA] gets packet bits 94-95, LTLACCSR[A] gets packet bits 64-92, LTLTLTLDIDCCSR[DIDMSB] gets 16-23, LTLDIDCCSR[DID] gets packet bits 24-31,LTLDIDCCSR[SIDMSB] gets bits 32-39, LTLDIDCCSR[SID] gets bits 40-47, LTLCCCSR[FT] gets packet bits 12-15, LTLCCCSR[TT] gets packet bits 48-51, LTLCCCSR[MI] gets 0's	RapidIO packet is dropped.
Critical Request Flow Not Checked for error.	-	-	-	-	-
TransportType Received reserved TT	Yes	LTLEDCSR[TSE]	No	Same as first entry	RapidIO packet is dropped
Received TT which is not enabled. - Error valid when passthrough is disabled and accept_all is disabled Or when accept_all is enabled.	Yes if LTLEECSR[TSE] is set	LTLEDCSR[TSE]	No	Same as first entry	RapidIO packet is dropped
DestID	Yes if LTLEECSR[ITTE] is set	LTLEDCSR[ITTE]	Yes for Nwrite_r. No for Nwrite.	Same as first entry	Internal platform error response is generated to self for

Table continues on the next page...

Table 21-271. Hardware Errors For NWrite, NWrite_r, and Unsupported Atomic Test-and-Swap Transactions (continued)

Error	Interrupt Generated	Status Bit Set	RapidIO Error Response Generated	Logical/Transport Layer Capture Register	Comments
DestID does not match this port's DeviceID if Alternate DeviceID is disabled or DestId does not match either Alternate DeviceID or DeviceId if Alternate DeviceID is enabled. Error valid when (passthrough or accept_all) is false					Nwrite_r. RapidIO packet is dropped for Nwrite
SourceID Not applicable	-	-	-	-	-
TransactionType Received RapidIO packet for Atomic test-and-swap transaction	Yes if LTLEECSR[UT] is set	LTLEDCSR[UT]	Yes	Same as first entry	Internal platform error response is generated to self
TransactionType Received RapidIO packet with reserved TType for this ftype Packet is treated as Nwrite Transaction	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	No	Same as first entry	RapidIO packet is dropped
WrSize Not unsupported transaction WrSize request is for one of reserved sizes	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	Yes for Nwrite_r. No for Nwrite.	Same as first entry	Internal platform error response is generated to self for Nwrite_r. RapidIO packet is dropped for Nwrite
Address:WdPtr:Xambs Not unsupported transaction Nwrite request hits LCSBA1CSR	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	No for Nwrite.	Same as first entry	RapidIO packet is dropped for Nwrite
Address:WdPtr:Xambs Not unsupported transaction Request hits a protected ATMU window	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	Yes for Nwrite_r. No for Nwrite.	Same as first entry	Internal platform error response is generated to self for Nwrite_r. RapidIO packet is dropped for Nwrite

Table continues on the next page...

Table 21-271. Hardware Errors For NWrite, NWrite_r, and Unsupported Atomic Test-and-Swap Transactions (continued)

Error	Interrupt Generated	Status Bit Set	RapidIO Error Response Generated	Logical/Transport Layer Capture Register	Comments
Address:WdPtr:Xambs Write request hits overlapping ATMU windows Refer to Window Boundary Crossing Errors-Outbound	Yes if LTLEECSR[IACB] is set	LTLEDCSR[IACB]	Yes for Nwrite_r. No for Nwrite.	Same as first entry	Internal platform error response is generated to self for Nwrite_r. RapidIO packet is dropped for Nwrite
SrcTID Not Checked for error.	-	-	-	-	-
Address:WdPtr:Xambs Nwrite_r address matches LCSBA1CSR with non 32 bit read request. Performed only for Nwrite_r packet	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	Yes for Nwrite_r.	Same as first entry	Internal platform error response is generated to self
Header Size Not unsupported transaction Header size is less than 12 bytes for small Transport packet or less than 16 bytes for Large Transport packet - that is, no payload present. Large Transport packet has 14 valid bytes and 2 bytes of padding of 0s. Padding of 0s is not checked.	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	Yes for Nwrite_r. No for Nwrite.	Same as first entry	Internal platform error response is generated to self for Nwrite_r. RapidIO packet is dropped for Nwrite
PayloadSize Not unsupported transaction Payload is greater than that indicated by {wdptr:wrsize} field, payload is not double word aligned or does not have any payload	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	Yes for Nwrite_r. No for Nwrite.	Same as first entry	Internal platform error response is generated to self for Nwrite_r. RapidIO packet is dropped for Nwrite
LIODN: Lookup table miss	Yes if LTLEECSR[ILE] is set	LTLEDCSR[ILE]	Yes for Nwrite_r. No for Nwrite.	Same as first entry	Internal platform error response is generated to self for Nwrite_r.

Table 21-271. Hardware Errors For NWrite, NWrite_r, and Unsupported Atomic Test-and-Swap Transactions

Error	Interrupt Generated	Status Bit Set	RapidIO Error Response Generated	Logical/Transport Layer Capture Register	Comments
					RapidIO packet is dropped for Nwrite

Table 21-272. Hardware Errors For SWrite Transactions

Error	Interrupt Generated	Status Bit Set	Error Response Generated	Logical/Transport Layer Capture Register	Comments
Priority Swrite transaction priority is 3	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	No	Same as third entry	RapidIO packet is dropped
Critical Request Flow Not Checked for error.	-	-	-	-	-
TransportType Received reserved TT	Yes if LTLEECSR[TSE] is set	LTLEDCSR[TSE]	No	Using the incoming RapidIO packet, for Small Transport type packets, LTLACCSR[XA] gets packet bits 62-63, LTLACCSR[A] gets packet bits 32-60, LTLDIDCCSR[DIDMSB] gets 0's, LTLDIDCCSR[DID] gets packet bits 16-23, LTLDIDCCSR[SIDMSB] gets 0's, LTLDIDCCSR[SID] gets packet bits 24-31, LTLCCCSR[FT] gets packet bits 12-15, LTLCCCSR[TT] gets packet bits 32-35, LTLCCCSR[MI] gets 0's For Large Transport type packets. LTLACCSR[XA] gets packet bits 78-79, LTLACCSR[A] gets packet bits 48-76, LTLTLTLDIDCCSR[DIDMSB] gets 16-23, LTLDIDCCSR[DID] gets packet bits 24-31, LTLDIDCCSR[SIDMSB] gets bits 32-39, LTLDIDCCSR[SID] gets bits 40-47, LTLCCCSR[FT] gets packet bits 12-15, LTLCCCSR[TT] gets packet bits 48-51, LTLCCCSR[MI] gets 0's	RapidIO packet is dropped

Table continues on the next page...

Table 21-272. Hardware Errors For SWrite Transactions (continued)

Error	Interrupt Generated	Status Bit Set	Error Response Generated	Logical/Transport Layer Capture Register	Comments
Received TT which is not enabled. - Error valid when passthrough is disabled and accept_all is disabled Or when accept_all is enabled.	Yes if LTLEECSR[TSE] is set	LTLEDCSR[TSE]	No	Same as third entry	RapidIO packet is dropped
DestID DestID does not match this port's DeviceID if Alternate DeviceID is disabled or DestId does not match either Alternate DeviceID or DeviceId if Alternate DeviceID is enabled. Error valid when (passthrough or accept_all) is false	Yes if LTLEECSR[ITTE] is set	LTLEDCSR[ITTE]	No	Same as third entry	RapidIO packet is dropped
SourceID Not Checked for error.	-	-	-	-	-
Address:WdPtr:Xambs Swrite request hits overlapping ATMU windows. Refer to Window Boundary Crossing Errors-Outbound	Yes if LTLEECSR[IACB] is set	LTLEDCSR[IACB]	No	Same as third entry	RapidIO packet is dropped
Address:WdPtr:Xambs Request hits a protected ATMU window or the LCSBA1CSR	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	No	Same as third entry	RapidIO packet is dropped
PayloadSize Payload size is not in DWs, has exceeded 256 bytes or has no payload.	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	No	Same as third entry	RapidIO packet is dropped
LIODN: Lookup table miss	Yes if LTLEECSR[ILE] is set	LTLEDCSR[ILE]	No	Same as third entry	RapidIO packet is dropped

Table 21-273. Hardware Errors For Maintenance Response Transactions

Error	Interrupt Generated	Status Bit Set	Internal platform Error Response Generated	Logical/Transport Layer Capture Register	Comments
Priority Not UR	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	No	Using the incoming RapidIO packet, for Small Transport type packets, LTLACCSR[XA] gets packet bits 78-79, LTLACCSR[A] gets packet bits 48-76, LTLDIDCCSR[DIDMSB] gets 0's, LTLDIDCCSR[DID] gets packet bits 16-23, LTLDIDCCSR[SIDMSB] gets 0's,	RapidIO packet is dropped and ignored

Table continues on the next page...

Table 21-273. Hardware Errors For Maintenance Response Transactions (continued)

Error	Interrupt Generated	Status Bit Set	Internal platform Error Response Generated	Logical/Transport Layer Capture Register	Comments
Response priority is not higher than RapidIO maintenance request priority				LTLDIDCCSR[SID] gets packet bits 24-31, LTLCCCSR[FT] gets packet bits 12-15, LTLCCCSR[TT] gets packet bits 32-35, LTLCCCSR[MI] gets 0's For Large Transport type packets. LTLACCSR[XA] gets packet bits 94-95, LTLACCSR[A] gets packet bits 64-92, LTLTLTLDIDCCSR[DIDMSB] gets 16-23, LTLDIDCCSR[DID] gets packet bits 24-31, LTLDIDCCSR[SIDMSB] gets bits 32-39, LTLDIDCCSR[SID] gets bits 40-47, LTLCCCSR[FT] gets packet bits 12-15, LTLCCCSR[TT] gets packet bits 48-51, LTLCCCSR[MI] gets 0's	
Critical Request Flow CRF field is set when corresponding request's CRF field was clear	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	No	Same as first entry	RapidIO packet is dropped and ignored.
TransportType Received reserved TT	Yes if LTLEECSR[TSE] is set	LTLEDCSR[TSE]	No	Same as first entry	RapidIO packet is dropped and ignored
Received TT which is not enabled. - Error valid when passthrough is disabled and accept_all is disabled Or when accept_all is enabled.	Yes if LTLEECSR[TSE] is set	LTLEDCSR[TSE]	No	Same as first entry	RapidIO packet is dropped and ignored
DestID DestID does not match this port's DeviceID if Alternate DeviceID is disabled or DestId does not match either Alternate DeviceID or DeviceId if Alternate DeviceID is enabled. Error valid when (passthrough or accept_all) is false	Yes if LTLEECSR[ITTE] is set	LTLEDCSR[ITTE]	No	Same as first entry	RapidIO packet is dropped and ignored
SourceID Does not match the request's DestID	Yes if LTLEECSR[UR] is set	LTLEDCSR[UR]	No	Same as first entry	RapidIO packet is dropped and ignored

Table continues on the next page...

Table 21-273. Hardware Errors For Maintenance Response Transactions (continued)

Error	Interrupt Generated	Status Bit Set	Internal platform Error Response Generated	Logical/Transport Layer Capture Register	Comments
TransactionType Not UR Received RapidIO packet with reserved TType for this ftype	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	No	Same as first entry	RapidIO packet is dropped and ignored
Not UR Maintenance read/write response does not correspond to an outstanding valid message read/write request.	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	No	Same as first entry	RapidIO packet is dropped and ignored
HopCount Not Checked for error.	-	-	-	-	-
Status Not UR Is not "Done" or "Error" Not "Done" status for "read_response" transaction type with payload	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	No	Same as first entry	RapidIO packet is dropped and ignored
Status Not UR Error Response with or without payload.	Yes if LTLEECSR[IER] is set	LTLEDCSR[IER]	Yes	Same as first entry except error capture is done from original request	Internal platform error response is generated to requestor.
TargetTID No outstanding transaction for this TargetTID	Yes if LTLEECSR[UR] is set	LTLEDCSR[UR]	No	Same as first entry	RapidIO packet is dropped and ignored
Header Size Not UR Maintenance Read response - total payload size with done status is not greater than 4 Bytes. Maintenance Write response - total header size is less than 12 Bytes for Small Transport	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	No	Same as first entry	RapidIO packet is dropped and ignored

Table continues on the next page...

Table 21-273. Hardware Errors For Maintenance Response Transactions (continued)

Error	Interrupt Generated	Status Bit Set	Internal platform Error Response Generated	Logical/Transport Layer Capture Register	Comments
packet or is less than 16 Bytes for Large Transport packet. Padding of 0's for Small or Large Transport packets is not verified.					
PayloadSize Not UR Maintenance write response has payload. Maintenance read response with done status and payload not matching valid request size, request size for the response is invalid or payload size is not dword aligned.	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	No	Same as first entry	RapidIO packet is dropped and ignored
Packet response time-out Response is not received by configured time	Yes if LTLEECSR[PRT] is set	LTLEDCSR[PRT]	Yes	Same as first entry except error capture is done from original request.	Internal platform error response is generated to requestor.

Table 21-274. Hardware Errors For IO/GSM Response Transactions (Not Maintenance)

Error	Interrupt Generated	Status Bit Set	Internal platform Error Response Generated	Logical/Transport Layer Capture Register	Comments
Priority Not UR Response priority is not higher than RapidIO request priority	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	No	Using the incoming RapidIO packet, for Small Transport type packets, LTLACCSR[XA] gets packet bits 78-79 (if available), LTLACCSR[A] gets packet bits 48-76 (if available), LTLDIDCCSR[DIDMSB] gets 0's, LTLDIDCCSR[DID] gets packet bits 16-23, LTLDIDCCSR[SIDMSB] gets 0's, LTLDIDCCSR[SID] gets packet bits	RapidIO packet is dropped and ignored

Table continues on the next page...

Table 21-274. Hardware Errors For IO/GSM Response Transactions (Not Maintenance) (continued)

Error	Interrupt Generated	Status Bit Set	Internal platform Error Response Generated	Logical/Transport Layer Capture Register	Comments
				24-31, LTLCCCSR[FT] gets packet bits 12-15, LTLCCCSR[TT] gets packet bits 32-35, LTLCCCSR[MI] gets 0's For Large Transport type packets. LTLACCSR[XA] gets packet bits 94-95 (if available), LTLACCSR[A] gets packet bits 64-92 (if available), LTLTLTLDIDCCSR[DIDMSB] gets 16-23, LTLDIDCCSR[DID] gets packet bits 24-31, LTLDIDCCSR[SIDMSB] gets bits 32-39, LTLDIDCCSR[SID] gets bits 40-47, LTLCCCSR[FT] gets packet bits 12-15, LTLCCCSR[TT] gets packet bits 48-51, LTLCCCSR[MI] gets 0's	
Critical Request Flow CRF field is set when corresponding request's CRF field was clear	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	No	Same as first entry	RapidIO packet is dropped and ignored.
TransportType Received reserved TT for this ftype	Yes if LTLEECSR[TSE] is set	LTLEDCSR[TSE]	No	Same as first entry	RapidIO packet is dropped and ignored
Received TT which is not enabled. - Error valid when passthrough is disabled and accept_all is disabled Or when accept_all is enabled.	Yes if LTLEECSR[TSE] is set	LTLEDCSR[TSE]	No	Same as first entry	RapidIO packet is dropped and ignored
DestID DestID does not match this port's DeviceID if Alternate DeviceID is disabled or DestId does not match either Alternate DeviceID or DeviceId if Alternate DeviceID is enabled. Error valid when (passthrough or accept_all) is false	Yes if LTLEECSR[ITTE] is set	LTLEDCSR[ITTE]	No	Same as first entry	RapidIO packet is dropped and ignored
SourceID Does not match the request's DestID	Yes if LTLEECSR[UR] is set	LTLEDCSR[UR]	No	Same as first entry	RapidIO packet is dropped and ignored

Table continues on the next page...

Table 21-274. Hardware Errors For IO/GSM Response Transactions (Not Maintenance) (continued)

Error	Interrupt Generated	Status Bit Set	Internal platform Error Response Generated	Logical/Transport Layer Capture Register	Comments
TransactionType Not UR Received RapidIO packet with reserved TType	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	No	Same as first entry	RapidIO packet is dropped and ignored
Not UR IO read response does not correspond to an outstanding valid IO/GSM read request. IO write response does not correspond to an outstanding valid IO/GSM write request.	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	No	Same as first entry	RapidIO packet is dropped and ignored
Status Not UR IO transaction - Is not "Done" or "Error" GSM transaction IO_Read_Home Is not "Done - Data-Only", "Done - Done-Intervention", "Done", "Retry" or "Error". Flush_w_Data response is not "Done", "Retry" or "Error" Transaction type of "Response_with_data" and status is not done	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	No	Same as first entry	RapidIO packet is dropped and ignored.
Status Not UR GSM Error response	Yes if LTLEECSR[GER] is set	LTLEDCSR[GER]	Yes if data is not received for this request.	Same as first entry except error capture is done from original request packet.	Internal platform error response is generated to requestor if data is not forwarded to it. Else the RapidIO packet is dropped.
Status Not UR IO Error Response	Yes if LTLEECSR[IER] is set	LTLEDCSR[IER]	Yes	Same as first entry except error capture is done from original request packet.	Internal platform error response is

Table continues on the next page...

Table 21-274. Hardware Errors For IO/GSM Response Transactions (Not Maintenance) (continued)

Error	Interrupt Generated	Status Bit Set	Internal platform Error Response Generated	Logical/Transport Layer Capture Register	Comments
					generated to requestor.
TargetTID No outstanding transaction for this TargetTID	Yes if LTLEECSR[UR] is set	LTLEDCSR[UR]	No	Same as first entry	RapidIO packet is dropped and ignored.
Packet Size Not UR (All non-maintenance and non-message) Write response - Header size is not 8 Bytes for Small Transport packet or not 12 Bytes for Large Transport packet GSM - "Done" response packet size to "Flush" is not 8 Bytes for Small Transport packet or not 12 Bytes for Large Transport packet. "Done-Intervention" is not 8 Bytes for Small Transport and 12 Bytes for Large Transport field. Two byte padding of 0's in Large Transport field packet is not checked.	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	No	Same as first entry	RapidIO packet is dropped and ignored.
Payload Size Not UR IO - Read Response - total payload is not of the size requested. "Done" or "Done-Data_Only" response to IO_Read_Home with incorrect payload size. Response with transaction type "response_with_no_data" has payload	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	No	Same as first entry	RapidIO packet is dropped and ignored.

Table continues on the next page...

Table 21-274. Hardware Errors For IO/GSM Response Transactions (Not Maintenance) (continued)

Error	Interrupt Generated	Status Bit Set	Internal platform Error Response Generated	Logical/Transport Layer Capture Register	Comments
Retry Not UR GSM request has had one more than configured number of retries for non misaligned request. The misaligned GSM request has had one to four (cumulative for the corresponding child requests) more than configured number of retries.	Yes if LTLEECSR[RETE] is set	LTLEDCSR[RETE]	Yes	Same as first entry except error capture is done from original request.	Internal platform error response is generated to requestor.
Packet response time-out Response is not received by configured time for packets requiring RapidIO response. "GSM - IO_Read_Home" - Done_With_Data is not received in configured time when Done_Intervention is received for non misaligned request or last child of misaligned request. Done response is not received in configured time for non misaligned request or last child of misaligned request. EME capture occurs for each child packet response time-out.	Yes if LTLEECSR[PRT] is set	LTLEDCSR[PRT]	Yes	Same as first entry except error capture is done from original request.	Internal platform error response is generated to requestor.
Packet response time-out Response is not received by configured time for packets requiring RapidIO response. "GSM - IO_Read_Home" - Done_Intervention is not received in configured time when Done_With_Data is received. This is true for both non misaligned or misaligned requests.	Yes if LTLEECSR[PRT] is set	LTLEDCSR[PRT]	No	Same as first entry except error capture is done from original request.	An Internal platform done response is generated when the Done_With_Data is received for non misaligned requests or the last child of a misaligned

Table continues on the next page...

Table 21-274. Hardware Errors For IO/GSM Response Transactions (Not Maintenance) (continued)

Error	Interrupt Generated	Status Bit Set	Internal platform Error Response Generated	Logical/Transport Layer Capture Register	Comments
					request. Therefore, an error response cannot be sent when the packet response time-out occurs.
GSM - IO_Read_Home Not UR Done response, Retry response, or Error response is after Done_Intervention response or Data_only is received.	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	No	Same as first entry	RapidIO packet is dropped and ignored.
Not UR Response for Internal platform packets not requiring response, but converted to "response" type packet by ATMU receives Error response. For example, NWrite converted to NWrite_r received "Error" response	Yes if LTLEECSR[IER]/[GER]/[RETE] is set	LTLEDCSR[IER]/[GER]/[RETE]	No	Same as first entry except error capture is done from original request.	RapidIO packet is dropped and ignored.
Response for Internal platform packets not requiring response, but converted to "response" type packet by ATMU is not received by configured time.	Yes if LTLEECSR[PRT] is set	LTLEDCSR[PRT]	No	Same as first entry except error capture is done from original request.	No error response is generated.

Table 21-275. Hardware Errors For Message Request Transactions

Error	Interrupt Generated	Status Bit Set	RapidIO Error Response Generated	Logical/Transport Layer Capture Register	Comments
Critical Request Flow Not Checked for error.	-	-	-	-	-

Table continues on the next page...

Table 21-275. Hardware Errors For Message Request Transactions (continued)

Error	Interrupt Generated	Status Bit Set	RapidIO Error Response Generated	Logical/Transport Layer Capture Register	Comments
TransportType Received reserved TT	Yes if LTLEECSR[TSE] is set	LTLEDCSR[TSE]	No	Using the original request RapidIO packet, for small Transport type, LTLACCSR[XA] gets packet bits 78-79, LTLACCSR[A] gets packet bits 48-76, LTLDIDCCSR[DIDMSB] gets 0's, LTLDIDCCSR[DID] gets packet bits 16-23,LTLDIDCCSR[SIDMSB] gets 0's, LTLDIDCCSR[SID] gets packet bits 24-31, LTLCCCSR[FT] gets packet bits 12-15, LTLCCCSR[TT] gets packet bits 32-35, LTLCCCSR[MI] gets packet bits 40-47. For Large Transport type packets. LTLACCSR[XA] gets packet bits 94-95, LTLACCSR[A] gets packet bits 64-92, LTLTLTLDIDCCSR[DIDMSB] gets 16-23, LTLDIDCCSR[DID] LTL gets packet bits 24-31, LTLDIDCCSR[SIDMSB] gets bits 32-39, LTLDIDCCSR[SID] gets bits 40-47, LTLCCCSR[FT] gets packet bits 12-15, LTLCCCSR[TT] gets packet bits 48-51, LTLCCCSR[MI] gets packet bits 56-63.	RapidIO packet is dropped
Received TT which is not enabled. - Error valid when passthrough is disabled and accept_all is disabled Or when accept_all is enabled.	Yes if LTLEECSR[TSE] is set	LTLEDCSR[TSE]	No	Same as third entry	RapidIO packet is dropped
DestID DestID does not match this port's DeviceID if Alternate DeviceID is disabled or DestId does not match either Alternate DeviceID or DeviceId if Alternate DeviceID is enabled. Error valid when (passthrough or accept_all) is false	Yes if LTLEECSR[ITTE] is set	LTLEDCSR[ITTE]	Yes if priority is not 3. Else packet is dropped	Same as third entry	Internal platform error response is sent to self if request priority is not 3. Else packet is dropped.
SourceID Not Checked for error.	-	-	-	-	-
MsgLen,Ssize,Ltr,Mbox, MsgSeg Not Checked for error.	-	-	-	-	-
Other Received Message request with SOCAR[M] disabled.	-	-	-	-	-

Table 21-275. Hardware Errors For Message Request Transactions

Error	Interrupt Generated	Status Bit Set	RapidIO Error Response Generated	Logical/Transport Layer Capture Register	Comments
Not Checked for error.					

Table 21-276. Hardware Errors For Message Response Transactions

Error	Interrupt Generated	Status Bit Set	Internal platform Error Response Generated	Logical/Transport Layer Capture Register	Comments
Priority Not Checked for error.	-	-	-	Using the original request RapidIO packet, for small Transport type, LTLACCSR[XA] gets packet bits 78-79, LTLACCSR[A] gets packet bits 48-76, LTLDIDCCSR[DIDMSB] gets 0's, LTLDIDCCSR[DID] gets packet bits 16-23, LTLDIDCCSR[SIDMSB] gets 0's, LTLDIDCCSR[SID] gets packet bits 24-31, LTLCCCSR[FT] gets packet bits 12-15, LTLCCCSR[TT] gets packet bits 32-35, LTLCCCSR[MI] gets packet bits 40-47. For Large Transport type packets. LTLACCSR[XA] gets packet bits 94-95, LTLACCSR[A] gets packet bits 64-92, LTLTLTLDIDCCSR[DIDMSB] gets 16-23, LTLDIDCCSR[DID] LTL gets packet bits 24-31,LTLDIDCCSR[SIDMSB] gets bits 32-39, LTLDIDCCSR[SID] gets bits 40-47, LTLCCCSR[FT] gets packet bits 12-15, LTLCCCSR[TT] gets packet bits 48-51, LTLCCCSR[MI] gets packet bits 56-63.	-
Critical Request Flow Not Checked for error.	-	-	-	-	-
TransportType Received reserved TT	Yes if LTLEECSR[TSE] is set	LTLEDCSR[TSE]	No	Same as first entry except capture registers are loaded from the response RapidIO packet	RapidIO packet is dropped and ignored
Received TT which is not enabled. - Error valid when passthrough is disabled and accept_all is disabled Or when accept_all is enabled.	Yes if LTLEECSR[TSE] is set	LTLEDCSR[TSE]	No	Same as first entry except capture registers are loaded from the response RapidIO packet	RapidIO packet is dropped and ignored

Table continues on the next page...

Table 21-276. Hardware Errors For Message Response Transactions (continued)

Error	Interrupt Generated	Status Bit Set	Internal platform Error Response Generated	Logical/Transport Layer Capture Register	Comments
DestID (All non-maintenance) DestID does not match this port's DeviceID if Alternate DeviceID is disabled or DestId does not match either Alternate DeviceID or DeviceId if Alternate DeviceID is enabled. Error valid when (passthrough or accept_all) is false	Yes if LTLEECSR[ITTE] is set	LTLEDCSR[ITTE]	No	Same as first entry except capture registers are loaded from the response RapidIO packet	RapidIO packet is dropped and ignored
SourceID Not Checked for error.	-	-	-	-	-
Status Not Checked for error.	-	-	-	-	-
Other Received Message response with SOCAR[M] disabled. Not Checked for error.	-	-	-	-	-

Table 21-277. Hardware Errors For Doorbell Request Transaction

Error	Interrupt Generated	Status Bit Set	Error Response Generated	Logical/Transport Layer Capture Register	Comments
Critical Request Flow Not Checked for error.	-	-	-	-	-
TransportType Received reserved TT	Yes if LTLEECSR[TSE] is set	LTLEDCSR[TSE]	No	Using the original request RapidIO packet, for small Transport type, LTLACCSR[XA] gets packet bits 78-79, LTLACCSR[A] gets packet bits 48-76, LTLDIDCCSR[DIDMSB] gets 0's, LTLDIDCCSR[DID] gets packet bits 16-23, LTLDIDCCSR[SIDMSB] gets 0's, LTLDIDCCSR[SID] gets packet bits 24-31, LTLCCCSR[FT] gets packet bits 12-15, LTLCCCSR[TT] gets packet bits 32-35, LTLCCCSR[MI] gets 0's For Large Transport type packets. LTLACCSR[XA] gets packet bits 94-95, LTLACCSR[A] gets packet bits 64-92, LTLTLTLDIDCCSR[DIDMSB] gets	RapidIO packet is dropped

Table continues on the next page...

Table 21-277. Hardware Errors For Doorbell Request Transaction (continued)

Error	Interrupt Generated	Status Bit Set	Error Response Generated	Logical/Transport Layer Capture Register	Comments
				16-23, LTLDIDCCSR[DID] LTL gets packet bits 24-31, LTLDIDCCSR[SIDMSB] gets bits 32-39, LTLDIDCCSR[SID] gets bits 40-47, LTLCCCSR[FT] gets packet bits 12-15, LTLCCCSR[TT] gets packet bits 48-51, LTLCCCSR[MI] gets 0's	
Received TT which is not enabled. - Error valid when passthrough is disabled and accept_all is disabled Or when accept_all is enabled.	Yes if LTLEECSR[TSE] is set	LTLEDCSR[TSE]	No	Same as the second entry	RapidIO packet is dropped
DestID DestID does not match this port's DeviceID if Alternate DeviceID is disabled or DestId does not match either Alternate DeviceID or DeviceId if Alternate DeviceID is enabled. Error valid when (passthrough or accept_all) is false	Yes if LTLEECSR[ITTE] is set	LTLEDCSR[ITTE]	Yes if priority is not 3. Else packet is dropped	Same as the second entry	Internal platform error response is sent to self if request priority is not 3. Else packet is dropped.
SourceID Not Checked for error.	-	-	-	-	-
SrcTID Not Checked for error.	-	-	-	-	-
Other Received Doorbell request with DOCAR[D] disabled. Not Checked for error.	-	-	-	-	-

Table 21-278. Hardware Errors For Doorbell Response Transactions

Error	Interrupt Generated	Status Bit Set	Internal platform Error Response Generated	Logical/Transport Layer Capture Register	Comments
TransportType Received reserved TT	Yes if LTLEECSR[TSE] is set	LTLEDCSR[TSE]	No	Using the incoming RapidIO packet, for small Transport type, LTLACCSR[XA] gets packet bits 78-79, LTLACCSR[A] gets packet bits 48-76, LTLDIDCCSR[DIDMSB] gets 0's, LTLDIDCCSR[DID] gets packet bits 16-23, LTLDIDCCSR[SIDMSB] gets 0's, LTLDIDCCSR[SID] gets packet bits 24-31, LTLCCCSR[FT] gets packet bits 12-15, LTLCCCSR[TT] gets packet bits 32-35, LTLCCCSR[MI] gets 0's For Large Transport type packets r. LTLACCSR[XA] gets packet bits 94-95, LTLACCSR[A] gets packet bits 64-92, LTLTLTLDIDCCSR[DIDMSB] gets 16-23, LTLDIDCCSR[DID] LTL gets packet bits 24-31,LTLDIDCCSR[SIDMSB] gets bits 32-39, LTLDIDCCSR[SID] gets bits 40-47, LTLCCCSR[FT] gets packet bits 12-15, LTLCCCSR[TT] gets packet bits 48-51, LTLCCCSR[MI] gets 0's	RapidIO packet is dropped and ignored
Received TT Error valid when passthrough is disabled and accept_all is disabled Or when accept_all is enabled.	Yes if LTLEECSR[TSE] is set	LTLEDCSR[TSE]	No	Same as first entry	RapidIO packet is dropped and ignored
DestID DestID does not match this port's DeviceID if Alternate DeviceID is disabled or DestId does not match either Alternate DeviceID or DeviceId if Alternate DeviceID is enabled. Error valid when (passthrough or accept_all) is false	Yes if LTLEECSR[ITTE] is set	LTLEDCSR[ITTE]	No	Same as first entry	RapidIO packet is dropped and ignored

Table 21-279. Hardware Errors for PortWrite Transaction

Error	Interrupt Generated	Status Bit Set	Error Response Generated	Logical/Transport Layer Capture Register	Comments
Priority Not Applicable	-	-	-	-	-

Table continues on the next page...

Table 21-279. Hardware Errors for PortWrite Transaction (continued)

Error	Interrupt Generated	Status Bit Set	Error Response Generated	Logical/Transport Layer Capture Register	Comments
Critical Request Flow Not Checked for error.	-	-	-	-	-
TransportType Received reserved TT	Yes if LTLEECSR[TSE] is set	LTLEDCSR[TSE]	No	Using the original request RapidIO packet, for small Transport type, LTLACCSR[XA] gets packet bits 78-79, LTLACCSR[A] gets packet bits 48-76, LTLDIDCCSR[DIDMSB] gets 0's,LTLDIDCCSR[DID] gets packet bits 16-23, LTLDIDCCSR[SIDMSB] gets 0's, LTLDIDCCSR[SID] gets packet bits 24-31, LTLCCCSR[FT] gets packet bits 12-15, LTLCCCSR[TT] gets packet bits 32-35, LTLCCCSR[MI] gets 0's Large Transport type packets. LTLACCSR[XA] gets packet bits 94-95, LTLACCSR[A] gets packet bits 64-92, LTLTLTLDIDCCSR[DIDMSB] gets 16-23, LTLDIDCCSR[DID] LTL gets packet bits 24-31, LTLDIDCCSR[SIDMSB] gets bits 32-39, LTLDIDCCSR[SID] gets bits 40-47, LTLCCCSR[FT] gets packet bits 12-15, LTLCCCSR[TT] gets packet bits 48-51, LTLCCCSR[MI] gets 0's	RapidIO packet is dropped
Received TT which is not enabled. - Error valid when passthrough is disabled and accept_all is disabled Or when accept_all is enabled.	Yes if LTLEECSR[TSE] is set	LTLEDCSR[TSE]	No	Same as third entry	RapidIO packet is dropped
DestID DestID does not match this port's DeviceID if Alternate DeviceID is disabled or DestId does not match either Alternate DeviceID or DeviceId if Alternate DeviceID is enabled. Error valid when (passthrough or accept_all) is false	Yes if LTLEECSR[ITTE] is set	LTLEDCSR[ITTE]	No	Same as third entry	RapidIO packet is dropped
SourceID Not Checked for error.	-	-	-	-	-
TransactionType Not Checked for error.	-	-	-	-	-
SrcTID Not Checked for error.	-	-	-	-	-
HopCount	-	-	-	-	-

Table continues on the next page...

Table 21-279. Hardware Errors for PortWrite Transaction (continued)

Error	Interrupt Generated	Status Bit Set	Error Response Generated	Logical/Transport Layer Capture Register	Comments
Not Checked for error.					
ConfigOffset Not Checked for error.	-	-	-	-	-
Other Received PortWrite transaction with DOCAR[PW] disabled. Not Checked for error.	-	-	-	-	-

Table 21-280. Hardware Errors For Type9 Transactions

Error	Interrupt Generated	Status Bit Set	Internal platform Error Response Generated	Logical/Transport Layer Capture Register	Comments
TransportType Received reserved TT	Yes if LTLEECSR[TSE] is set	LTLEDCSR[TSE]	No	Using the incoming RapidIO packet, for small Transport type, LTLACCSR[XA] gets packet bits 78-79, LTLACCSR[A] gets packet bits 48-76, LTLDIDCCSR[DIDMSB] gets 0's, LTLDIDCCSR[DID] gets packet bits 16-23, LTLDIDCCSR[SIDMSB] gets 0's, LTLDIDCCSR[SID] gets packet bits 24-31, LTLCCCSR[FT] gets packet bits 12-15, LTLCCCSR[TT] gets packet bits 32-35, LTLCCCSR[MI] gets 0's For Large Transport type packets r. LTLACCSR[XA] gets packet bits 94-95, LTLACCSR[A] gets packet bits 64-92, LTLTLTLDIDCCSR[DIDMSB] gets 16-23, LTLDIDCCSR[DID] LTL gets packet bits 24-31,LTLDIDCCSR[SIDMSB] gets bits 32-39, LTLDIDCCSR[SID] gets bits 40-47, LTLCCCSR[FT] gets packet bits 12-15, LTLCCCSR[TT] gets packet bits 48-51, LTLCCCSR[MI] gets 0's	RapidIO packet is dropped and ignored
Received TT Error valid when passthrough is disabled and accept_all is disabled Or when accept_all is enabled.	Yes if LTLEECSR[TSE] is set	LTLEDCSR[TSE]	No	Same as first entry	RapidIO packet is dropped and ignored
DestID DestID does not match this port's DeviceID if Alternate DeviceID is disabled or DestId does not match either	Yes if LTLEECSR[ITTE] is set	LTLEDCSR[ITTE]	No	Same as first entry	RapidIO packet is dropped and ignored

Table 21-280. Hardware Errors For Type9 Transactions

Error	Interrupt Generated	Status Bit Set	Internal platform Error Response Generated	Logical/Transport Layer Capture Register	Comments
Alternate DeviceID or DeviceId if Alternate DeviceID is enabled. Error valid when (passthrough or accept_all) is false					

Table 21-281. Hardware Errors for Reserved Ftype

Error	Interrupt Generated	Status Bit Set if corresponding bit is enabled	RapidIO Error Response Generated	Logical/Transport Layer Capture Register	Comments
Ftype Ftype is not IO Read, IO Write, SWrite, Maintenance request, Maintenance Response, Response (Ftype 13), Doorbell or Message class and it is not a passthrough transaction. (passthrough is not enabled or accept_all is enabled or transaction is addressed to this port)	Yes if LTLEECSR[IUT] is set	LTLEDCSR[UT]	No	Using the original request RapidIO packet, for small Transport type, LTLACCSR[XA] gets packet bits 78-79, LTLACCSR[A] gets packet bits 48-76, LTLDIDCCSR[DIDMSB] gets 0's,LTLDIDCCSR[DID] gets packet bits 16-23, LTLDIDCCSR[SIDMSB] gets 0's, LTLDIDCCSR[SID] gets packet bits 24-31, LTLCCCSR[FT] gets packet bits 12-15, LTLCCCSR[TT] gets packet bits 32-35, LTLCCCSR[MI] gets 0's Large Transport type packets. LTLACCSR[XA] gets packet bits 94-95, LTLACCSR[A] gets packet bits 64-92, LTLTLTLDIDCCSR[DIDMSB] gets 16-23, LTLDIDCCSR[DID] LTL gets packet bits 24-31,LTLDIDCCSR[SIDMSB] gets bits 32-39, LTLDIDCCSR[SID] gets bits 40-47, LTLCCCSR[FT] gets packet bits 12-15, LTLCCCSR[TT] gets packet bits 48-51, LTLCCCSR[MI] gets 0's	RapidIO packet is dropped
TransportType Received reserved TT	Yes if LTLEECSR[TSE] is set	LTLEDCSR[TSE]	No	Same as first entry	RapidIO packet is dropped
Received TT which is not enabled. - Error valid when passthrough is disabled and accept_all is disabled Or when accept_all is enabled.	Yes if LTLEECSR[TSE] is set	LTLEDCSR[TSE]	No	Same as first entry	RapidIO packet is dropped

Table continues on the next page...

Table 21-281. Hardware Errors for Reserved Ftype (continued)

Error	Interrupt Generated	Status Bit Set if corresponding bit is enabled	RapidIO Error Response Generated	Logical/Transport Layer Capture Register	Comments
DestID DestID does not match this port's DeviceID if Alternate DeviceID is disabled or DestId does not match either Alternate DeviceID or DeviceId if Alternate DeviceID is enabled. Error valid when (passthrough or accept_all) is false	Yes if LTLEECSR[ITTE] is set	LTLEDCSR[ITTE]	No	Same as first entry	RapidIO packet is dropped
Address:WdPtr:Xambs Request hits overlapping ATMU windows. Refer to Window Boundary Crossing Errors-Outbound Packet checked as non Swrite packet	Yes if LTLEECSR[IACB] is set	LTLEDCSR[IACB]	No	Same as first entry	RapidIO packet is dropped
Address:WdPtr:Xambs Not unsupported transaction Request hits a protected ATMU window or the LCSBA1CSR Packet checked as non Swrite packet	Yes if LTLEECSR[ITD] is set	LTLEDCSR[ITD]	No	Same as first entry	RapidIO packet is dropped

Table 21-282. Hardware Errors for Outbound Transaction Crossed ATMU Boundary

Error	Interrupt Generated	Status Bit Set if corresponding bit is enabled		Logical/Transport Layer Capture Register	Comments
Internal platform Address and payload size Internal platform address range for Outbound RapidIO transaction hits multiple ATMU windows. Refer to Window Boundary Crossing Errors-Outbound	Yes if LTLEECSR[OACB] is set and/or LTLEECSR[PRT] is set	LTLEDCSR[OACB], LTLEDCSR[PRT]		Using the original request RapidIO packet send out by OB, for small Transport type, LTLACCSR[XA] gets packet bits 78-79, LTLACCSR[A] gets packet bits 48-76, LTLDIDCCSR[DIDMSB] gets 0's, LTLDIDCCSR[DID] gets packet bits 16-23,	-

Table 21-282. Hardware Errors for Outbound Transaction Crossed ATMU Boundary

Error	Interrupt Generated	Status Bit Set if corresponding bit is enabled		Logical/Transport Layer Capture Register	Comments
				LTLDIDCCSR[SIDMSB] gets 0's, LTLDIDCCSR[SID] gets packet bits 24-31, LTLCCCSR[FT] gets packet bits 12-15, LTLCCCSR[TT] gets packet bits 32-35, LTLCCCSR[MI] gets 0's For Large Transport type packets, LTLACCSR[XA] gets packet bits 94-95, LTLACCSR[A] gets packet bits 64-92, LTLTLTLDIDCCSR[DIDMSB] gets 16-23, LTLDIDCCSR[DID] LTL gets packet bits 24-31, LTLDIDCCSR[SIDMSB] gets bits 32-39, LTLDIDCCSR[SID] gets bits 40-47, LTLCCCSR[FT] gets packet bits 12-15, LTLCCCSR[TT] gets packet bits 48-51, LTLCCCSR[MI] gets 0's	

Table 21-283. Hardware Errors for Outbound Packet Time-to-live Errors

Error	Interrupt Generated	Status Bit Set if corresponding bit is enabled		Logical/Transport Layer Capture Register	Comments
Packet time-to-live error	Yes if LTLEECSR[PTTL] is set	LTLEDCSR[PTTL]		Using the RapidIO packet attempted to send outbound, if configured in small transport mode, LTLACCSR[XA] gets packet bits 78-79, LTLACCSR[A] gets packet bits 48-76,LTLDIDCCSR[DIDMSB] gets 0's, LTLDIDCCSR[DID] gets packet bits 16-23, LTLDIDCCSR[SIDMSB] gets 0's, LTLDIDCCSR[SID] gets packet bits 24-31, LTLCCCSR[FT] gets packet bits 12-15, LTLCCCSR[TT] gets packet bits 32-35, LTLCCCSR[MI] gets 0's For large transport mode, LTLACCSR[XA] gets packet bits 94-95, LTLACCSR[A] gets packet bits 64-92, LTLTLTLDIDCCSR[DIDMSB] gets 16-23, LTLDIDCCSR[DID] LTL gets packet bits 24-31, LTLDIDCCSR[SIDMSB] gets bits	-

Table 21-283. Hardware Errors for Outbound Packet Time-to-live Errors

Error	Interrupt Generated	Status Bit Set if corresponding bit is enabled		Logical/Transport Layer Capture Register	Comments
				32-39, LTLDIDCCSR[SID] gets bits 40-47, LTLCCCSR[FT] gets packet bits 12-15, LTLCCCSR[TT] gets packet bits 48-51, LTLCCCSR[MI] gets 0's	

Chapter 22
SATA Controller

22.1 SATA overview

The serial ATA controller is a high-performance SATA solution incorporating some of the latest SATA-IO extensions. The SATA may also be referred to as a host bus adapter (HBA). The SATA controller is designed to operate in a system that supports command queuing and, in particular, a switching scheme based on a frame information structure (FIS) using port multipliers.

FIS-based switching requires the SATA controllers to maintain in hardware a context for each command it has queued at the devices that are attached to it. FIS-based switching also requires the SATA controller to maintain a queue per attached device ensuring that the command issue order for each device is maintained. It can be used in SATA controllers, as well as storage area network (SAN), network attached storage (NAS), and RAID (redundant array of independent/inexpensive disks) devices.

22.2 SATA features summary

SATA controller includes the following features:

- Designed to comply with *Serial ATA 2.6 Specification*
- Supports speeds: 1.5 Gbps (first-generation SATA), 3 Gbps (second-generation SATA)
- Supports FIS-based switching
- Supports advanced technology attachment packet interface (ATAPI) devices
- Contains high-speed descriptor-based DMA controller
- Supports native command queuing (NCQ) commands
- Supports port multiplier operation
- Supports hot plug including asynchronous signal recovery

This figure shows the SATA block diagram.

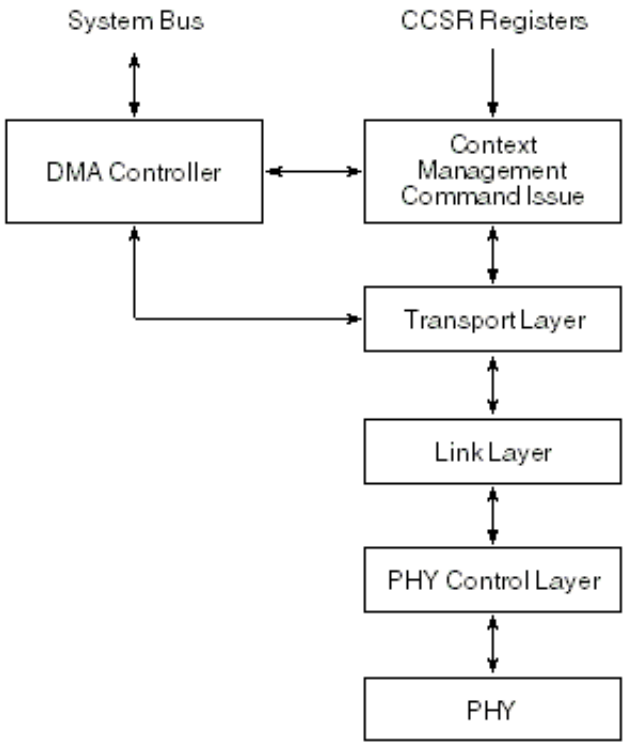


Figure 22-1. SATA block diagram

There are four layers in the SATA architecture: application, transport, link, and PHY. The application layer is responsible for overall ATA command execution, including controlling command block register accesses. The transport layer is responsible for placing control information and data to be transferred between the host and device in a packet/frame, known as a frame information structure (FIS). The link layer is responsible for taking data from the constructed frames, inserting control characters and moving data to the PHY layer. The PHY layer is responsible for 8B/10B encoding/decoding, then transmitting and receiving the encoded information as a serial data stream on the wire.

22.3 Command operation

The SATA controller maintains a queue consisting of up to 16 commands.

These commands can be distributed to a single attached device or, if the system contains a port multiplier, over each of the attached devices. It is possible to queue queued commands and non-queued commands into the SATA controller, provided the host software does not break protocol to any particular device (it is illegal to issue a non-queued command to a device that still has a queued command active as per ATAPI/ATA protocol).

22.3.1 Command issue

When the host software is preparing to issue a command, it first builds a command descriptor as shown in Figure 22-66. The format of the command FIS is defined in the Serial ATA 2.6 standard shown in Table 22-68. The software is also responsible for the creation of a scatter/gather list for data movement. This list should be defined to exactly match the transfer length as programmed into the command FIS. If the 16 entries are not sufficient, then an extended entry (ext) can be used to refer to an alternate table. When the descriptor is built, the host software locates a free command slot within the SATA controller by examining the command queue register. To issue the command, the host software programs the address of the command descriptor and the attributes into the appropriate command header locations and then issues the command by writing the PMP and setting the appropriate CQ bit in the command queue register.

22.3.2 Command service

After a command is issued, the SATA controller takes ownership of the command descriptor, transferring the command FIS to the targeted device when required, servicing the data transfer using the scatter/gather list provided and transferring the status back into the command descriptor (if programmed).

22.3.3 Command completion interrupt timing

When a command completes, it is possible to enable the SATA controller to generate an interrupt.

Associated with some commands there will be a command completion status FIS. The SATA controller will always transfer the status FIS to memory whether it indicates an error or good command completion.

22.3.4 DMA context (read data)

When receiving FIS's from attached devices, the SATA controller has to support interleaving from various devices.

Data FIS's from device 0 could be interleaved with data FIS's for device 1. In order to accomplish this, the SATA controller maintains in hardware a context for each command which is pushed onto and pulled from the DMA controller when needed to service the transfers.

22.3.5 DMA context (write data)

When the SATA controller receives an FIS indicating that the next operation to a particular device should be a data write transfer, the SATA controller will lock the interface by forcing the link layer to transition to X_RDY immediately and not go through idle SYNC.

This will mean that write transfers will not have to be interleaved, which simplifies the transmit data path and eliminates the need for a complex scheduler.

22.3.6 DMAT primitive processing

The SATA controller supports the reception of the DMAT primitive.

When the SATA controller receives a DMAT primitive from the device, it will perform the following actions.

The DMA controller will complete the current read burst and transfer the data to the transport layer FIFO. The DMA controller signals an EOF on the last data of the burst, which causes the link layer to insert the CRC and EOF. The context for this transfer is returned to the context store. Once this action is completed, the device can terminate the transfer or re-initiate the transfer as per Serial ATA Revision 2.6 .

22.4 Command layer overview

The function of the SATA command layer is to allow host software to queue commands.

It then manages the commands issued and services them using a hardware context to complete the queued commands.

22.5 SATA memory map/register definition

This table shows the memory map for the SATA registers. The offsets to the memory map table are defined for both SATA hosts. Undefined 4-byte address spaces within offset 0x000-0xFFF are reserved.

NOTE

All registers (except SYSPR) described in this section and descriptors described in Command header and Command descriptor use little-endian byte ordering. Software running on the local processor in big-endian mode must byte-swap the data.

SATA memory map

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
22_0000	Command queue register (SATA1_CQR)	32	R/W	0000_0000h	22.5.1/1385
22_0008	Command active register (SATA1_CAR)	32	R	0000_0000h	22.5.2/1385
22_0010	Command completed register (SATA1_CCR)	32	w1c	0000_0000h	22.5.3/1386
22_0018	Command error register (SATA1_CER)	32	w1c	0000_0000h	22.5.4/1387
22_0020	Device error register (SATA1_DER)	32	w1c	0000_0000h	22.5.5/1387
22_0024	Command header base address (SATA1_CHBA)	32	R/W	0000_0000h	22.5.6/1388
22_0028	Host status register (SATA1_HStatus)	32	w1c	2000_0000h	22.5.7/1389
22_002C	Host control register (SATA1_HControl)	32	R/W	0000_0100h	22.5.8/1392
22_0030	Port number queue register (SATA1_CQPMP)	32	R/W	0000_0000h	22.5.9/1394
22_0034	Signature register (SATA1_SIG)	32	R	FFFF_FFFFh	22.5.10/ 1395
22_0038	Interrupt coalescing control register (SATA1_ICC)	32	R/W	0100_0000h	22.5.11/ 1395
22_0100	SATA interface status register (SATA1_SStatus)	32	R	0000_0000h	22.5.12/ 1396
22_0104	SATA interface error register (SATA1_SError)	32	w1c	0000_0000h	22.5.13/ 1397
22_0108	SATA interface control register (SATA1_SControl)	32	R/W	0000_0300h	22.5.14/ 1400
22_010C	SATA interface notification register (SATA1_SNotification)	32	w1c	0000_0000h	22.5.15/ 1401
22_0140	Transport layer configuration (SATA1_TransCfg)	32	R/W	0800_0016h	22.5.16/ 1402
22_0148	Link layer configuration (SATA1_LinkCfg)	32	R/W	3800_FF34h	22.5.17/ 1403
22_014C	Link layer configuration1 (SATA1_LinkCfg1)	32	R/W	0000_0000h	22.5.18/ 1404
22_0150	Link layer configuration2 (SATA1_LinkCfg2)	32	R/W	0000_0000h	22.5.19/ 1405
22_015C	PHY control configuration1 (SATA1_PhyCtrlCfg1)	32	R/W	0000_3800h	22.5.20/ 1405

Table continues on the next page...

SATA memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
22_0410	System priority register (SATA1_SYSPR)	32	R/W	0000_0000h	22.5.21/ 1407
22_1000	Command queue register (SATA2_CQR)	32	R/W	0000_0000h	22.5.1/1385
22_1008	Command active register (SATA2_CAR)	32	R	0000_0000h	22.5.2/1385
22_1010	Command completed register (SATA2_CCR)	32	w1c	0000_0000h	22.5.3/1386
22_1018	Command error register (SATA2_CER)	32	w1c	0000_0000h	22.5.4/1387
22_1020	Device error register (SATA2_DER)	32	w1c	0000_0000h	22.5.5/1387
22_1024	Command header base address (SATA2_CHBA)	32	R/W	0000_0000h	22.5.6/1388
22_1028	Host status register (SATA2_HStatus)	32	w1c	2000_0000h	22.5.7/1389
22_102C	Host control register (SATA2_HControl)	32	R/W	0000_0100h	22.5.8/1392
22_1030	Port number queue register (SATA2_CQPMP)	32	R/W	0000_0000h	22.5.9/1394
22_1034	Signature register (SATA2_SIG)	32	R	FFFF_FFFFh	22.5.10/ 1395
22_1038	Interrupt coalescing control register (SATA2_ICC)	32	R/W	0100_0000h	22.5.11/ 1395
22_1100	SATA interface status register (SATA2_SStatus)	32	R	0000_0000h	22.5.12/ 1396
22_1104	SATA interface error register (SATA2_SError)	32	w1c	0000_0000h	22.5.13/ 1397
22_1108	SATA interface control register (SATA2_SControl)	32	R/W	0000_0300h	22.5.14/ 1400
22_110C	SATA interface notification register (SATA2_SNotification)	32	w1c	0000_0000h	22.5.15/ 1401
22_1140	Transport layer configuration (SATA2_TransCfg)	32	R/W	0800_0016h	22.5.16/ 1402
22_1148	Link layer configuration (SATA2_LinkCfg)	32	R/W	3800_FF34h	22.5.17/ 1403
22_114C	Link layer configuration1 (SATA2_LinkCfg1)	32	R/W	0000_0000h	22.5.18/ 1404
22_1150	Link layer configuration2 (SATA2_LinkCfg2)	32	R/W	0000_0000h	22.5.19/ 1405
22_115C	PHY control configuration1 (SATA2_PhyCtrlCfg1)	32	R/W	0000_3800h	22.5.20/ 1405
22_1410	System priority register (SATA2_SYSPR)	32	R/W	0000_0000h	22.5.21/ 1407

22.5.1 Command queue register (SATAx_CQR)

Before queuing a command into the SATA controller, the CQR is first examined to detect a free command queue (CQ) slot. A free CQ slot is indicated by a 0 in a bit position. To queue a command, the bit corresponding to the CQ slot to use is set. At this point the SATA controller takes ownership of the command header space and command descriptor associated with the command slot. While the command is queued in the SATA controller or at the device, the command queue bit remains 1. When the command completes, this bit is cleared to 0 by the hardware. For a device error, the CQR holds the command queue bits at 1 for each command queued or issued to the device in error. When the host software clears the device error, the hardware in turn clears each of the commands queued.

Address: Base address + 0h 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	Reserved																CQn															
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

SATAx_CQR field descriptions

Field	Description
31–16 -	This field is reserved. Reserved
CQn	Command *n* queue bit

22.5.2 Command active register (SATAx_CAR)

When a command is issued from the SATA controller to the device, the command is marked as active by the hardware setting the appropriate command active bit of the CAR. Once a command completes, the hardware clears the appropriate bit of the CAR.

For a device error, the CAR holds the command active bits at 1 for each command issued to the device in error. When the host software clears the device error, the hardware in turn clears each of the commands queued.

Address: Base address + 8h offset

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

SATAx_CAR field descriptions

Field	Description
31–16 -	This field is reserved. Reserved
CAn	Command n active bit

22.5.3 Command completed register (SATAx_CCR)

When a command completes, the hardware sets the command completed bit for that command in the CCR to a 1. The hardware also clears both the command queue and the command active bit for that command. When the software needs to acknowledge the reception of the completed command, it can do so in two ways:

- Writing a 1 to the command complete bit
- Issuing a command to the command slot

An interrupt coalescing scheme runs on the CCR. When the register contains a value other than 0x0000_0000, an interrupt coalescing timer runs. Each time a command completion is acknowledged, the timer is reset. When the timer times out, an interrupt is generated.

Address: Base address + 10h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved																CCn															
W																	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SATAx_CCR field descriptions

Field	Description
31–16 -	This field is reserved. Reserved
CCn	Command n completed bit

22.5.4 Command error register (SATAx_CER)

When a device errors a command by setting the error bit in the status register, this is detected by the SATA controller as a single device error. The associated command completing due to error is indicated by the hardware setting the command error bit for that command in the CER . For safe operation under both command queuing and non-queuing operation, all commands queued into the SATA controller and at the device are considered aborted. The queue for that device is stopped. The values of the registers CQR, CAR, and CCR will allow the host software to know which commands have completed without error and those that were queued at the SATA controller and at the device.

When the host software clears the device error (by setting DER), the software is also responsible to clear CER by writing a 1 to the command error bit for the command that was in error. After the error condition at the device has been cleared, the host application software can reissue the commands to the SATA controller, which were aborted on the reception of the single device error.

Address: Base address + 18h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved																CEn															
W																	w1c															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SATAx_CER field descriptions

Field	Description
31–16 -	This field is reserved. Reserved
CEn	Command n error bit

22.5.5 Device error register (SATAx_DER)

When a single device error is detected, the device that issued the error is indicated by the hardware setting the device error bit in the DER. The procedure as outlined in the command error register applies to the queues and to restarting the device.

The host application software acknowledges the device in error by clearing the device error bit. The device error is cleared by writing a 1 to the appropriate device error bit. When this action is performed, the queue to the device in error is cleared and is ready to have commands queue.

While a device is in error, no command can be queued for that device.

Address: Base address + 20h offset

Bit	31–16	15–0
R	Reserved	DEn
W		w1c
Reset	0	0

SATAx_DER field descriptions

Field	Description
31–16 -	This field is reserved. Reserved
DEn	Device n error bit

22.5.6 Command header base address (SATAx_CHBA)

The CHBA holds the address in memory of where the command header block is located. It must be written as part of the host software initialization process. After the SATA controller hardware is brought online, the SATA controller takes ownership of this register. The address in this register should not be changed while the SATA controller is online.

Address: Base address + 24h offset

Bit	31–16	15–2	1–0
R W	CHBA	CHBA	Reserved
Reset	0	0	0

SATAx_CHBA field descriptions

Field	Description
31–2 CHBA	Command header base address
-	This field is reserved. Reserved, should be 00.

22.5.7 Host status register (SATAx_HStatus)

HStatus, holds the status of the SATA controller as well as the interrupt sources. When an event occurs, the interrupt bit is set regardless of the status of the associated interrupt enable bit. The interrupt signal from the SATA controller is gated with the associated interrupt enable register. For all interrupt bits other than the interrupt on command complete bit, when software has processed the interrupt condition, it acknowledges the interrupt by writing a 1 to the interrupt source bit. This action will clear the interrupt signal if there are no other outstanding interrupts in HStatus.

The interrupt on command complete requires special processing. This bit is set as a result of the programmed interrupt coalescing algorithm running on the register CCR contents. For the interrupt on command complete bit, the command(s) that have completed to cause this interrupt need to be cleared by clearing the command N completed bit of the CCR. When the number or staleness of the CCR falls below the programmed interrupt coalescing algorithm, the interrupt on command complete bit clears.

Address: Base address + 28h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	HS_ON	HS_OFF	BE	OP_MODE	Reserved									ME	PTx_Err	PRx_Err
W	w1c	w1c	w1c	w1c										w1c	w1c	w1c
Reset	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved			DLM	CET	CER	FOT	FOR	Reserved		FE	PR	SIGU	SNTFU	DE	CC
W				w1c	w1c	w1c	w1c	w1c			w1c	w1c	w1c	w1c	w1c	w1c
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SATAx_HStatus field descriptions

Field	Description
31 HS_ON	Online/offline. This bit indicates if the SATA controller is online or offline.

Table continues on the next page...

SATAx_HStatus field descriptions (continued)

Field	Description
	0 Offline. The SATA controller is non-operational and the PHY is held in reset. 1 Online. The SATA controller is operational.
30 HS_OFF	Going offline. This bit indicates that the SATA controller is going offline it is waiting for the commands queued within the SATA controller or active at the device to complete. 0 Host is not in process going offline 1 Host is in process going offline
29 BE	BIST error. When the protocol is placed into BIST, this bit maps the BIST error. 0 Indicates that the link layer is passing BIST 1 Indicates that the link layer is not passing BIST When the protocol is not in BIST this bit asserts high and can be ignored.
28 OP_MODE	Operating mode. 0 The host controller is operating in legacy mode. 1 The host controller is operating in enterprise mode.
27–19 -	This field is reserved. Reserved
18 ME	SATA controller master error. Indicates if the host received an error on the interface during the access to memory. 0 No error response is received when a transfer was made into the memory 1 Error response is received during the transfer into the memory
17 PTx_Err	SATA controller parity Tx error. 0 No parity errors were detected on Tx data path 1 One or more parity errors were detected on Tx data path
16 PRx_Err	SATA controller parity Rx error. 0 No parity errors were detected on Rx data path 1 One or more parity errors were detected on Rx data path
15–13 -	This field is reserved. Reserved
12 DLM	Data Length Mismatch
11 CET	CRC error Tx. When set, this bit indicates that one or more CRC errors occurred in Tx data path.
10 CER	CRC error Rx. When set, this bit indicates that one or more CRC errors occurred in Rx data path.
9 FOT	FIFO overflow Tx. When set, this bit indicates that Tx FIFO is in overflow condition while sending FIS.
8 FOR	FIFO overflow Rx. When set, this bit indicates that Rx FIFO is in overflow condition while receiving FIS.
7–6 -	This field is reserved. Reserved
5 FE	Fatal error. When set, this bit indicates that fatal error occurred in SATA controller. In this state, the interrupt will be generated if FATAL_INT is set in the host control register. Write '1' to clear the interrupt source.

Table continues on the next page...

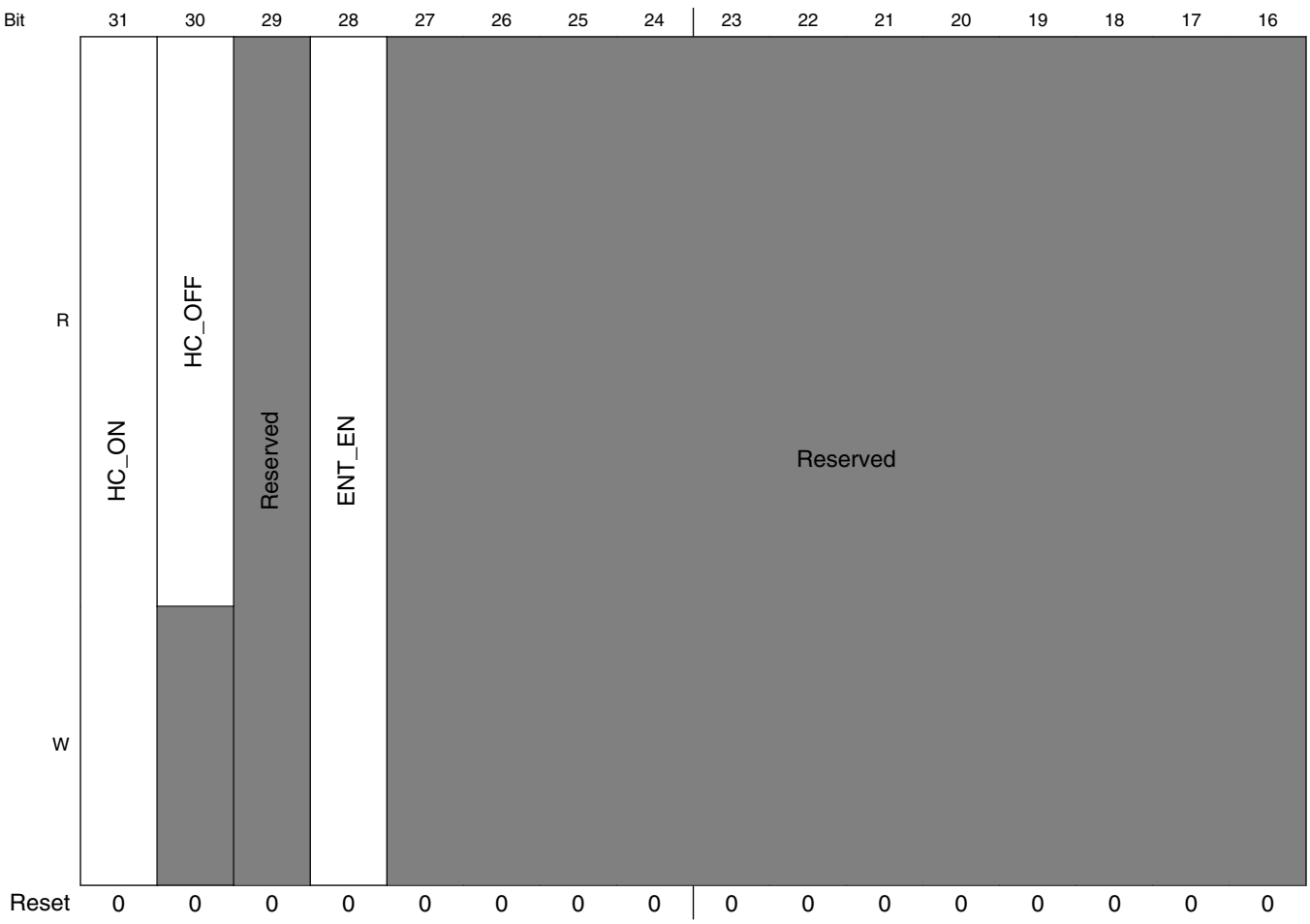
SATAx_HStatus field descriptions (continued)

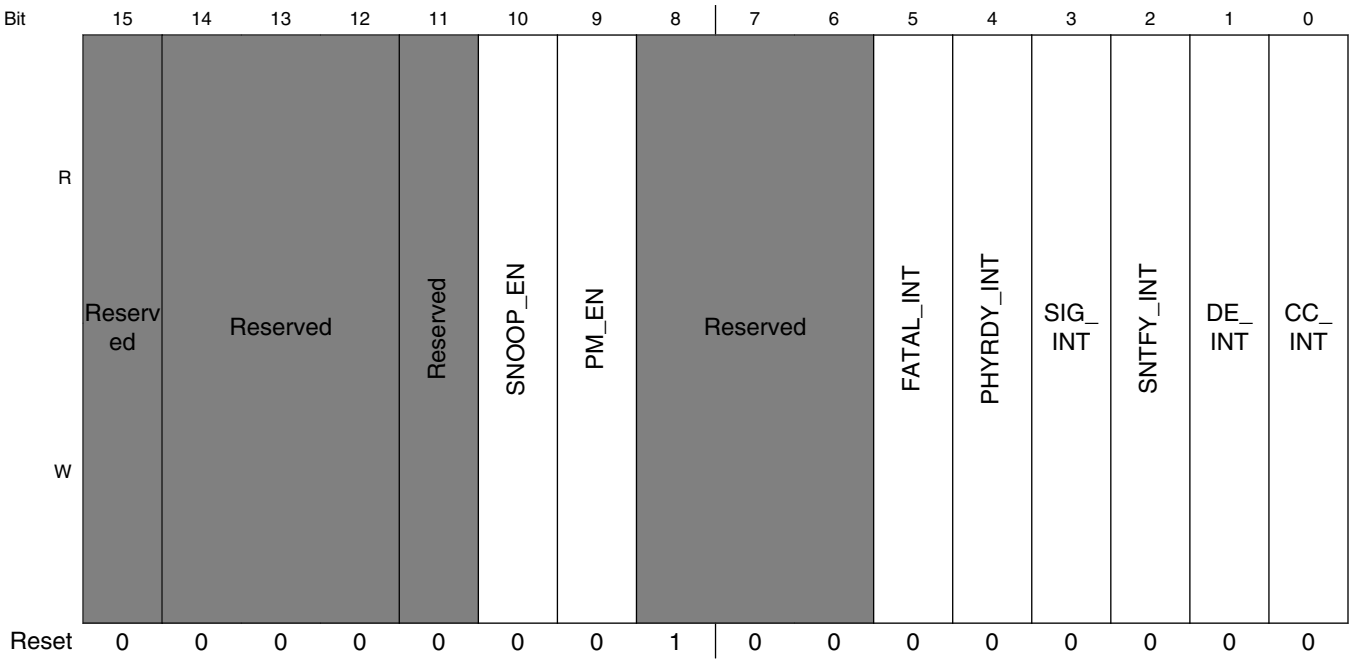
Field	Description
4 PR	PHY ready. When set, this bit indicates that PHY READY signal was changed. In this state, the interrupt will be generated if PHYRDY_INT is set in the host control register. Write '1' to clear the interrupt source.
3 SIGU	Signature update. When set, this bit indicates that the signature is updated in the host signature register. In this state, the interrupt will be generated if SIG_INT is set in the host control register. Write '1' to clear the interrupt source.
2 SNTFU	SNotification update. When set, this bit indicates that the SNotification register has at least one bit set. In this state, the interrupt will be generated if SNTFY_INT is set in the host control register. Write '1' to clear the interrupt source.
1 DE	Device error. When set, this bit indicates that the DE register has at least one bit set. In this state, the interrupt will be generated if DE_INT is set in the host control register. Write '1' to clear the interrupt source.
0 CC	Command complete. When set, this bit indicates that the register CCR has at least one bit set. In this state, the interrupt will be generated if CC_INT is set in the host control register. Write '1' to clear the interrupt source.

22.5.8 Host control register (SATA*x*_HControl)

HControl is written to control the operation of the SATA controller. To enable an interrupt, the associated bit must be set; to disable the interrupt, the associated bit must be cleared.

Address: Base address + 2Ch offset





SATAx_HControl field descriptions

Field	Description
31 HC_ON	Online/offline control 0 Offline. Bring the SATA controller offline and place the PHY in reset 1 Online. Bring the SATA controller online
30 HC_OFF	Offline request status. This is a read only bit. 1 The SATA controller is currently completing an operation and will go offline when the operation completes. When this bit is set , the SATA controller can be forced to go offline by aborting its current operation by writing 0 to the HC_ON bit.
29 -	This field is reserved. Reserved
28 ENT_EN	Enterprise enable bit 0 Switch the host controller to enterprise mode. Can only be achieved when the host is offline 1 Switch the host controller to legacy mode. Can only be achieved when the host is offline
27–15 -	This field is reserved. Reserved
14–12 -	This field is reserved. Reserved. Reset value must be preserved when writing to the register.
11 -	This field is reserved. Reserved
10 SNOOP_EN	Snoop enable during header fetch. 0 Snoop not enabled during command header fetch. 1 Snoop enabled during command header fetch.
9 PM_EN	Port multiplier attached. This bit is used to indicate if the HBA is attached to a port multiplier. This bit is set or cleared by software.

Table continues on the next page...

SATAx_HControl field descriptions (continued)

Field	Description
	0 This SATA controller is directly attached to a SATA device. The SATA controller hardware does not auto-detect the presence of a port multiplier; this is to allow for future changes in signature type for the port multiplier. 1 A port multiplier is attached to the SATA controller.
8–6 -	This field is reserved. Reserved. Reset value must be preserved when writing to the register.
5 FATAL_INT	Enable interrupt on fatal error
4 PHYRDY_INT	Enable interrupt on PHY ready change
3 SIG_INT	Enable interrupt signature update
2 SNTFY_INT	Enable interrupt on SNotify register update
1 DE_INT	Enable interrupt on single device error
0 CC_INT	Enable interrupt on command complete

22.5.9 Port number queue register (SATAx_CQPMP)

When queuing a command into the SATA controller, the CQPMP is written with the value of the PMP field that addresses the device to which the command will be issued. If the device is directly attached (that is, there is no port multiplier in the system), then this register is not required and should be cleared.

Address: Base address + 30h offset

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

SATAx_CQPMP field descriptions

Field	Description
31–4 -	This field is reserved. Reserved
CQPMP	Command queue port multiplier field

22.5.10 Signature register (SATAx_SIG)

The 32-bit SIG register contains the initial signature of an attached device when the first D2H register FIS is received from that device.

Address: Base address + 34h 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	LBA_HIGH								LBA_MID								LBA_LOW								SEC_CNT							
W																																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

SATAx_SIG field descriptions

Field	Description
31–24 LBA_HIGH	LBA high register
23–16 LBA_MID	LBA mid register
15–8 LBA_LOW	LBA low register
SEC_CNT	Sector count register

22.5.11 Interrupt coalescing control register (SATAx_ICC)

When a command completes, the SATA controller sets the corresponding bit in the command completed register. The interrupt coalescing scheme runs on the SIG register. The scheme runs in two ways:

- If the number of completed commands exceeds the threshold, then the interrupt will be signaled.
- If any command complete bit has been set in the register for a number of HCLK ticks equal to the programmed count, then the interrupt will be set. This timer will be reset each time a command completion is acknowledged by the application layer software.

Address: Base address + 38h 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	Reserved			ITC					Reserved								ITTCV															
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SATAx_ICC field descriptions

Field	Description
31–29 -	This field is reserved. Reserved
28–24 ITC	Interrupt threshold count. The number of command completions that will raise the interrupt. 00000 Implied no threshold and the interrupt will be signaled based on the threshold timer. 00001 The number of command complete bits which, if set, will cause the interrupt to be signaled. 01111 The number of command complete bits which, if set, will cause the interrupt to be signaled.
23–19 -	This field is reserved. Reserved
ITTCV	Interrupt threshold timer compare value. A value of 0 indicates that whenever a command complete bit is set the interrupt should be signaled.

22.5.12 SATA interface status register (SATAx_SStatus)

Serial ATA provides an additional block of registers to control the interface and to retrieve interface state information.

SStatus is a 32-bit read-only register that conveys the current state of the interface and host adapter. The register conveys the interface state at the time it is read and is updated continuously and asynchronously by the host adapter. Writes to this register have no effect.

Address: Base address + 100h offset

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

SATAx_SStatus field descriptions

Field	Description
31–12 -	This field is reserved. Reserved
11–8 IPM	Interface power management state. Indicates the current interface power management state. All other values reserved 0000 Device not present or communication not established 0001 Interface in active state 0010 Interface in partial power management state 0110 Interface in slumber power management state
7–4 SPD	Speed. Indicates the negotiated interface communication speed established. All other values reserved

Table continues on the next page...

SATAx_SStatus field descriptions (continued)

Field	Description
	0000 No negotiated speed (device not present or communication not established) 0001 First-generation communication rate negotiated 0010 Second-generation communication rate negotiated
DET	Detection. Indicates the interface device detection and PHY state. All other values reserved 0000 No device detected and PHY communication not established 0001 Device presence detected but PHY communication not established 0011 Device presence detected and PHY communication established 0100 PHY in offline mode as a result of the interface being disabled or running in a BIST loopback mode

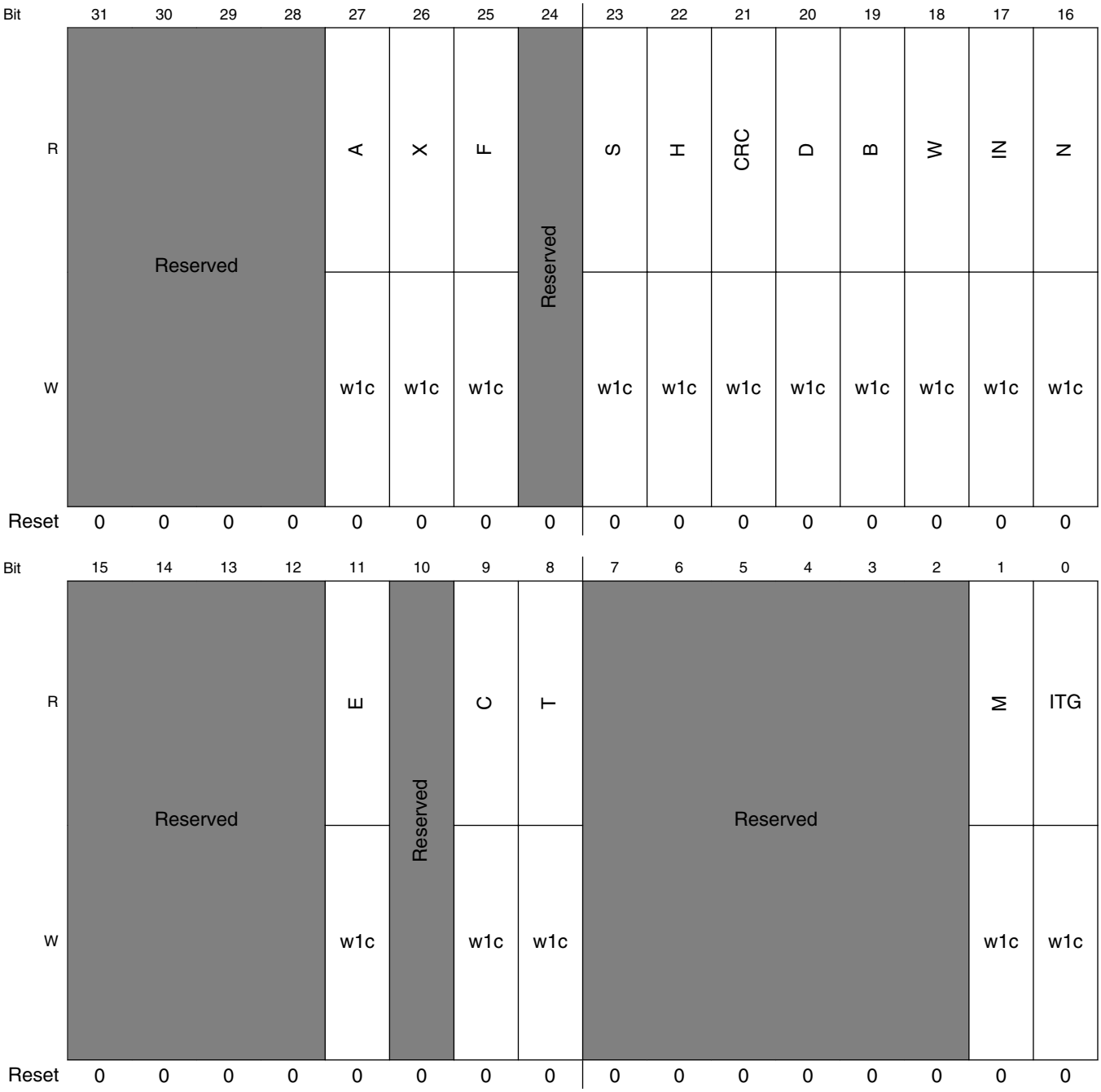
22.5.13 SATA interface error register (SATAx_SError)

SError is a 32-bit register that conveys supplemental interface error information to complement the error information available in the shadow register block error register. The register represents all the detected errors accumulated since the last time the SError register was cleared (whether recovered by the interface or not). Set bits in the error register are explicitly cleared by a write operation to the SError register or by a reset operation. The error bits that have been set in this register are cleared by writing a 1 to the corresponding field. Host software should clear the interface SError register at appropriate checkpoints in order to best isolate error conditions and the commands they impact.

Bits 31-16 of this register represent the DIAG decode bits, which contain diagnostic error information, for use by diagnostic software in validating correct operation or isolating failure modes. Bits 15-0 represent the ERR decode bits, which contain information for use by the host software in determining the appropriate response to the error condition.

SATA memory map/register definition

Address: Base address + 104h offset



SATAx_SError field descriptions

Field	Description
31–28 -	This field is reserved. Reserved bit for future use. Should be cleared.
27 A	Port selector presence detected. This bit is set when COMWAKE is received while the host is in state HP2: HR_AwaitCOMINIT. On power-up reset this bit is cleared. The bit is cleared when the host writes a 1 to this bit location.
26 X	Exchanged. When set to 1 this bit indicates that device presence has changed since the last time this bit was cleared. The means by which the implementation determines that the device presence has changed

Table continues on the next page...

SATAx_SError field descriptions (continued)

Field	Description
	is vendor specific. This bit may be set anytime a PHY reset initialization sequence occurs as determined by reception of the COMINIT signal, whether in response to a new device being inserted, to a COMRESET having been issued, or to power-up.
25 F	Unrecognized FIS type. When set to 1, this bit indicates that since the bit was last cleared, one or more FIS's were received by the transport layer with good CRC, but they had a type field that was not recognized.
24 -	This field is reserved. Reserved.
23 S	Link sequence error. When set to 1, this bit indicates that one or more link state machine error conditions were encountered since the last time this bit was cleared. The link layer state machine defines the conditions under which the link layer detects an erroneous transition.
22 H	Handshake error. When set to 1, this bit indicates that one or more R_ERRP handshake responses were received in response to frame transmission. Such errors may be the result of a CRC error detected by the recipient, of a disparity or 10b/8b decoding error, or of other error conditions leading to a negative handshake on a transmitted frame.
21 CRC	CRC error. When set to 1, this bit indicates that one or more CRC errors occurred with the link layer since the bit was last cleared.
20 D	Disparity error. When set to 1, this bit indicates that incorrect disparity was detected one or more times since the last time the bit was cleared.
19 B	10b to 8b decode error. When set to 1, this bit indicates that one or more 10-bit to 8-bit decoding errors occurred since the bit was last cleared.
18 W	COMWAKE detected. When set to 1, this bit indicates that a COMWAKE signal was detected by the PHY since the last time this bit was cleared.
17 IN	PHY internal error. When set to 1, this bit indicates that the PHY detected some internal error since the last time this bit was cleared.
16 N	PHYRDY change. When set to 1, this bit indicates that the PHYRDY signal changed state since the last time this bit was cleared.
15–12 -	This field is reserved. Reserved bit for future use; should be cleared.
11 E	E Internal error. 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. Host software should reset the interface before retrying the operation. If the condition persists, the host bus adapter may suffer from a design issue rendering it incompatible with the attached device.
10 -	This field is reserved. Reserved.
9 C	Non-recovered persistent communication or data integrity error. A communication error that was not recovered occurred that is expected to be persistent. Because the error condition is expected to be persistent, the operation need not be retried by the host software. 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.
8 T	Non-recovered transient data integrity error: A data integrity error occurred that was not recovered by the interface. Because the error condition is not expected to be persistent, the operation should be retried by the host software.
7–2 -	This field is reserved. Reserved
1 M	Recovered communications error. Communications between the device and host were temporarily lost but were 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 PHYRDY *n* signal between the

Table continues on the next page...

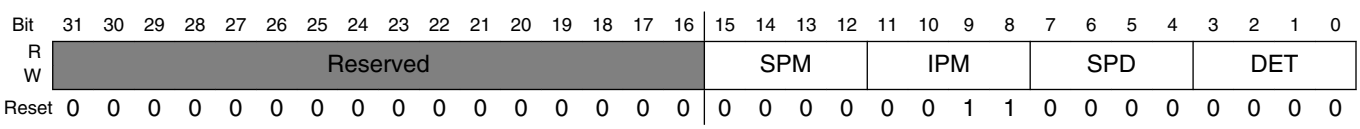
SATAx_SError field descriptions (continued)

Field	Description
	PHY and link layers. No action is required by the host software, because the operation ultimately succeeded. However, the host software may elect to track such recovered errors to gauge overall communications integrity and potentially step down the negotiated communication speed.
0 ITG	Recovered data integrity error. A data integrity error occurred that was recovered by the interface through a retry operation or other recovery action. This can arise from a noise burst in the transmission, a voltage supply variation, or other causes. No action is required by host software, because the operation ultimately succeeded. However, the host software may elect to track such recovered errors to gauge overall communications integrity and potentially step down the negotiated communication speed.

22.5.14 SATA interface control register (SATAx_SControl)

SControl is a 32-bit read-write register that provides the interface by which software controls SATA interface capabilities. Writes to the SControl register result in an action being taken by the host adapter or interface. Reads from the register return the last value written to it.

Address: Base address + 108h offset



SATAx_SControl field descriptions

Field	Description
31–16 -	This field is reserved. Reserved, should be cleared.
15–12 SPM	Select power management. Used to select a power management state. A non-zero value written to this field will cause the power management state specified to be initiated. A value written to this field is treated as a one-shot. All other values reserved 0000 No power management state transition requested 0001 Transition to the partial power management state initiated 0010 Transition to the slumber power management state initiated 0100 Transition to the active power management state initiated
11–8 IPM	Interface power management. The enabled interface power management states can be invoked via the SATA interface power management capabilities. All other values reserved 0000 No interface power management state restrictions 0001 Transitions to the partial power management state disabled 0010 Transitions to the slumber power management state disabled 0011 Transitions to both the partial and slumber power management states disabled

Table continues on the next page...

SATAx_SControl field descriptions (continued)

Field	Description
7–4 SPD	Speed. Highest allowed communication speed the interface is allowed to negotiate when interface communication speed is established. All other values reserved 0000 No speed negotiation restrictions 0001 Limit speed negotiation to a rate not greater than first-generation communication rate 0010 Limit speed negotiation to a rate not greater than second-generation communication rate
DET	Detection. Controls the host adapter device detection and interface initialization. All other values reserved 0000 No device detection or initialization action requested 0001 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 re-initialized. Upon a write to the SControl register that sets the DET field to 0001, the host interface should transition to the HP1: HR_Reset [Delete space after state and should remain in that state until the DET field is set to a value other than 0001 by a subsequent write to the SControl register. 0100 Disable the SATA interface and put PHY in offline mode

22.5.15 SATA interface notification register (SATAx_SNotification)

SNotification is a 32-bit, write-one-to-clear register that conveys the devices that have sent the host a set device bits FIS with the notification bit. When the host receives a set device bits FIS with the notification bit set to 1, the host should set the bit in SNotification corresponding to the value of the PM port field in the received FIS. For example, if the PM port field is set to 7 then the host should set bit 7 by writing a 1 to it. Next, the host should generate an interrupt if the I bit is set to one in the FIS and interrupts are enabled.

In this register, bits previously set are explicitly cleared by a write operation or by a power-on-reset operation. If the register is not cleared due to a COMRESET, the software is responsible for clearing the register as appropriate.The value written to clear set bits shall have ones encoded in the bit positions corresponding to the bits that are to be cleared.

Address: Base address + 10Ch offset

Bit	31–16	15–0
R	Reserved	Notify_n
W		w1c
Reset	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

SATAx_SNotification field descriptions

Field	Description
31–16 -	This field is reserved. Reserved, should be cleared.
Notify_n	Represents whether a particular device with the corresponding PM port number *n* has sent a set device bits FIS to the host with the notification bit set.

22.5.16 Transport layer configuration (SATAx_TransCfg)

TransCfg controls the configuration of the transport layer.

Address: Base address + 140h 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	DFIS_SIZE																Reserved											RX_WATER_ MARK					
Reset	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	1	0	1	1	0

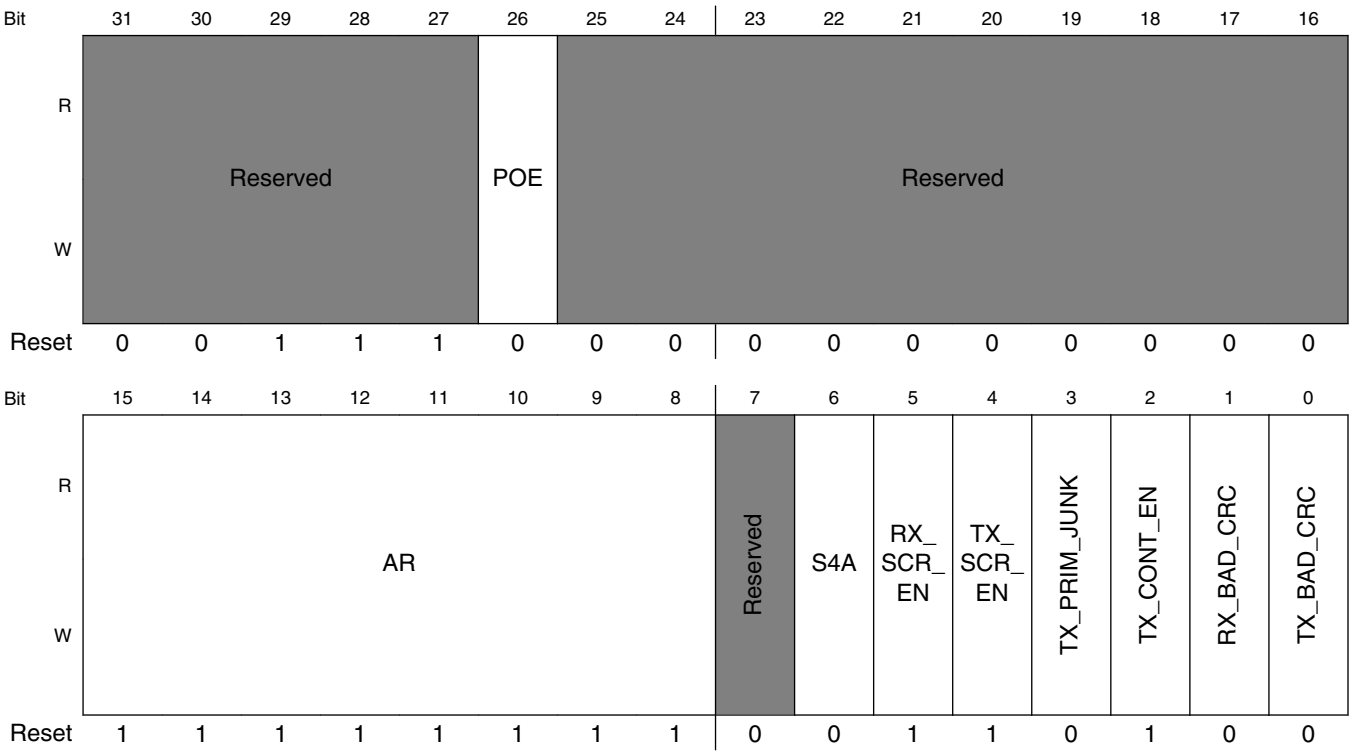
SATAx_TransCfg field descriptions

Field	Description
31–16 DFIS_SIZE	Data FIS framing length words. Determines the maximum length each data FIS should be. **NOTE:** Maximum data FIS size is 8K, it may vary depending on data size to be transfered.
15–5 -	This field is reserved. Reserved
RX_WATER_ MARK	This sets the number of locations in the 58-deep Rx FIFO that can be used before the transport layer instructs the link layer to transmit HOLDS to the transmitting end. Note that it can take some time for the HOLDs to get to the other end, and that in the interim there must be enough room in the FIFO to absorb all data that could arrive. An initial value of 15 is recommended.

22.5.17 Link layer configuration (SATAx_LinkCfg)

LinkCfg controls the configuration of the link layer.

Address: Base address + 148h offset



SATAx_LinkCfg field descriptions

Field	Description
31–27 -	This field is reserved. Reserved
26 POE	Primitive override enable. When set, this bit enables the replacement of a single primitive, as specified by CFG_PRIM/CFG_CD, when the link layer state machine is in the CFG_PRIM_OVR_STATE state. This bit has to be toggled from a 0 to a 1 to enable this feature.
25–16 -	This field is reserved. Reserved
15–8 AR	Align insertion rate. The SATA specification requires that the link layer send a pair of ALIGN primitives at least every 254 words of data. This is achieved by setting ALIGN_RATE to '11111111'. However, for test purposes it is possible to send ALIGNs at a higher rate. This can be achieved by setting ALIGN_RATE to a lower value (that is, ALIGN_RATE-1); words will be sent by the link layer between each set of ALIGN primitive pairs. **NOTE:** If SEND_4_ALIGNS is set, one should not set the ALIGN_RATE to be four or less. If SEND_4_ALIGNS is not set, one should not set the ALIGN_RATE to be two or less.
7 -	This field is reserved. Reserved

Table continues on the next page...

SATAx_LinkCfg field descriptions (continued)

Field	Description
6 S4A	Send four ALIGNs. When asserted, four ALIGN primitives are transmitted at the specified rate, instead of the normal two ALIGNS.
5 RX_SCR_EN	Rx scramble enable. If this bit is asserted, then descrambling of the receive data is enabled as per the SATA specification.
4 TX_SCR_EN	Tx scramble enable. If this bit is asserted, then scrambling of the transmit data is enabled as per the SATA specification.
3 TX_PRIM_JUNK	TX prim junk. If this bit is de-asserted, then scrambled junk data is sent after a CONT primitive, as per the SATA specification. If this bit is asserted, then the single character 0xDEADBEEF is sent continuously instead. This is to aid debug.
2 TX_CONT_EN	TX CONT. If this bit is asserted, then the transmission of CONT primitives is enabled. If de-asserted, then long sequences of repeated primitives can be sent by the link layer.
1 RX_BAD_CRC	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 to be toggled from a 0 to a 1 to enable this feature.
0 TX_BAD_CRC	Tx bad CRC. A bad CRC (inverted value of the correct CRC) value will be transmitted for one FIS only by the link layer when a rising edge is detected on this signal. This bit has to be toggled from a 0 to a 1 to enable this feature.

22.5.18 Link layer configuration1 (SATAx_LinkCfg1)

LinkCfg1 controls the configuration of the link layer.

Address: Base address + 14Ch offset

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									CD	PRIM_OVR_STATE					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SATAx_LinkCfg1 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6 CD	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_OVR_STATE = L_SendEOF and PRIM = WTRM, then a WTRM primitive will be inserted into the datastream instead of an EOF (whenever a rising edge is seen on PRIM_OVR_EN). If CD = 0, then a normal data character (as specified by PRIM) is inserted into the datastream instead of the EOF.
PRIM_OVR_ STATE	Prim override state. These 6 bits are used in the primitive override debug functionality. When the link layer detects a positive edge on PRIM_OVR_EN, it overrides the next primitive that would be inserted during the PRIM_OVR_STATE, with the data specified by the PRIM and CD configuration bits.

22.5.19 Link layer configuration2 (SATAx_LinkCfg2)

LinkCfg2 controls the configuration of the link layer.

Address: Base address + 150h 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	PRIM																															
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

SATAx_LinkCfg2 field descriptions

Field	Description
PRIM	This 32-bit bus specifies the data to be used in the overriding primitive debug logic, described in the definition of LinkCfg1 register.

22.5.20 PHY control configuration1 (SATAx_PhyCtrlCfg1)

PhyCtrlCfg1 controls the configuration of the link layer.

Address: Base address + 15Ch offset

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

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

SATAx_PhyCtrlCfg1 field descriptions

Field	Description
31–14 -	This field is reserved. Reserved
13 -	This field is reserved. Reserved. Reset value must be preserved when writing to the register.
12 ENDEC_EN	Encode decode enable. Reset value must be preserved when writing to the register. 0 When negated low, the PCS is configured to operate in 10-bit mode; the 8B/10B encoder/decoders are bypassed and it is assumed this is done elsewhere. 1 When asserted high, it enables the PCS to operate in 8 bit mode, and to enable 8B/10B encoding and decoding.
11–9 -	This field is reserved. Reserved. Reset value must be preserved when writing to the register.
8–7 FPRFTI	Force PHY ready, force Tx idle. This pair of signals determines how phyRdy is driven, how the output buffer IDLE condition is controlled and how disparity errors in ALIGN primitives should be tolerated during OOB. The IDLE condition is defined in SATA as both traces of the transmit differential pair being driven to common mode. • frcPhyRdy = 0 • frcTxIdle = 0 In this mode phyRdy and Tx buffer IDLE control driven by OOB state machine. Disparity errors in ALIGN primitives are not tolerated during OOB. • frcPhyRdy = 0 • frcTxIdle = 1 In this mode phyRdy and Tx buffer IDLE control driven by OOB state machine. Disparity errors in ALIGN primitives are tolerated during OOB. • frcPhyRdy = 1 • frcTxIdle = 0 In this mode phyRdy is asserted high and Tx buffer IDLE control is forced off, causing the output buffer to be enabled. Tolerance of disparity errors in ALIGN primitives is of no consequence, because OOB is bypassed. • frcPhyRdy = 1 • frcTxIdle = 1 In this mode phyRdy is asserted high and Tx buffer IDLE control is forced on, causing the output buffer to the IDLE condition. Tolerance of disparity errors in ALIGN primitives is of no consequence as OOB is bypassed.
6–4 -	This field is reserved. Reserved
3–1 LPB_EN	Loopback enable. These bits control both loopback modes and power management modes. **NOTE:** This field is available only for SATA1. 000 No loopback and in normal power mode 001 Far end re-timed (parallel) loopback enabled 010 Near end analog (serial) loopback enabled 011 Invalid 100 Invalid 101 goPartial. This encoding results in the OOB state machine entering the partial state. Note that in the PCS, partial and slumber have the same effect.

Table continues on the next page...

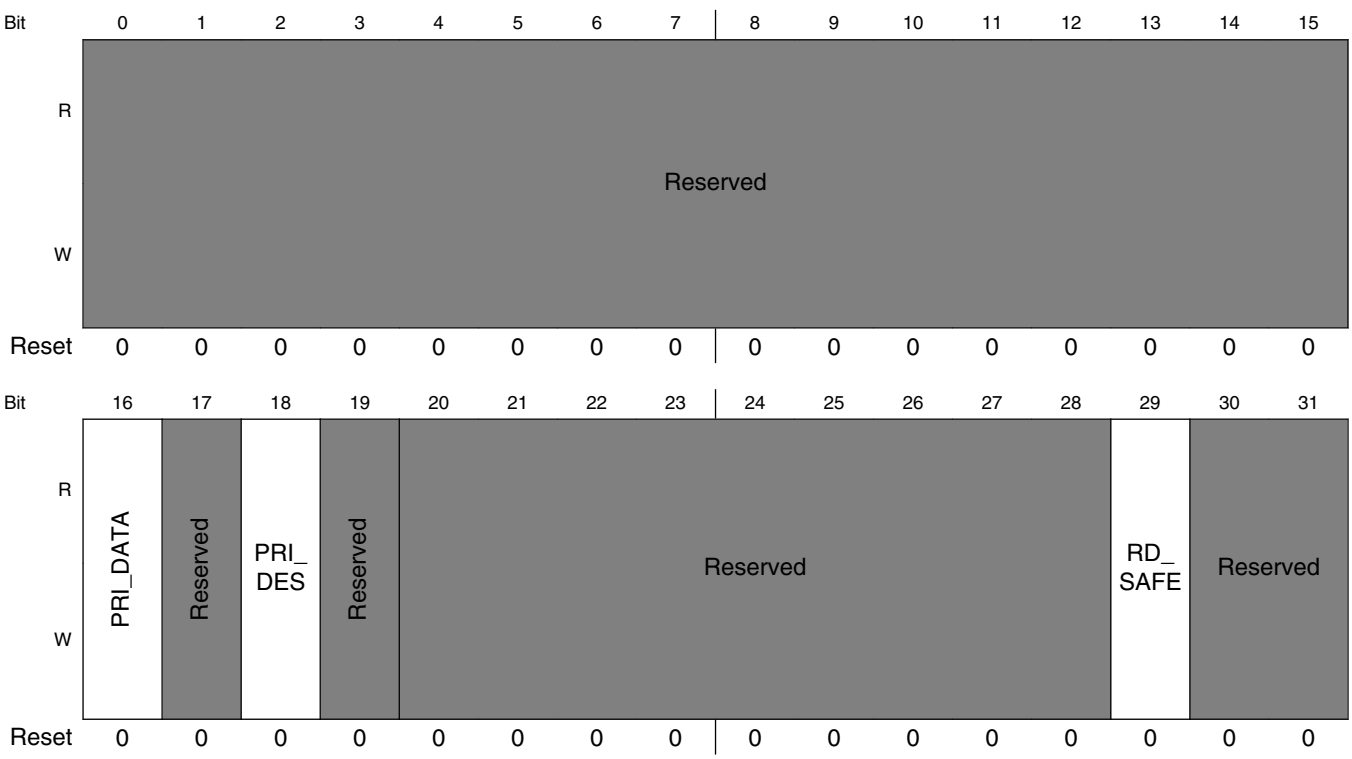
SATAx_PhyCtrlCfg1 field descriptions (continued)

Field	Description
	110 goSlumber. This encoding results in the OOB state machine entering the slumber state. Note that in the PCS, partial and slumber have the same effect. 111 Invalid
0 -	This field is reserved. Reserved

22.5.21 System priority register (SATAx_SYSPR)

SYSPR can be used to control various settings that affect the system response to DMA operations.

Address: Base address + 410h offset



SATAx_SYSPR field descriptions

Field	Description
0–15 -	This field is reserved. Reserved
16 PRI_DATA	Data high priority enable.

Table continues on the next page...

SATAx_SYSPR field descriptions (continued)

Field	Description
	0 Normal operation. SATA data transfers have default (low) priority. 1 High priority enabled for SATA data transfers.
17 - Reserved	This field is reserved.
18 PRI_DES	Descriptor fetch high priority enable. 0 Normal operation. SATA descriptor fetches have default (low) priority. 1 High priority enabled for SATA descriptor fetches.
19 - Reserved	This field is reserved.
20–28 - Reserved	This field is reserved.
29 RD_SAFE	Read safe. This bit should be set only if the target of the read DMA operation is a well behaved memory that is not affected by the read operation and which 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. 0 It is not safe to read more bytes that were intended. 1 It is safe to read more bytes that were intended.
30–31 - Reserved	This field is reserved.

22.6 Functional description

22.6.1 Error processing

On a single device error, perform the following tasks:

1. Examine the DER register to determine which device is in the error state. Note that there may be multiple devices in error.
2. Examine the CER register to determine which command was in error. Note that software knows which command belongs to which device.
3. Examine the status location of the descriptor of the command in error and determine the reason for the error.
4. If needed, software should send commands to the device to clear down the error condition on device or for further examination of the device's status.
5. Clear the DER*n* bit by writing 1 to bit *n*, where *n* indicates the device in error. This also clears out the outstanding commands for that device.
6. Clear the CER*n* bit by writing 1 to bit *n*, where *n* indicates the associated command in error. After that, the software can reissue command to the device if needed.

On fatal error, perform the following tasks:

1. Read the error register and other registers to determine how many commands are outstanding and how many have completed without error.
2. Bring the SATA controller offline. When this happens, all queues within the SATA controller are cleared.
3. Perform what corrective action the software determines is necessary.
4. Bring the SATA controller online. This will initiate Out-of-Band (OOB), resetting the BUS, host, and attached device.

22.6.2 Bringing the SATA controller online/offline

The HC_ON bit in HControl allows the host software to bring the SATA controller online or offline.

The SATA controller online status should only be changed when there are no commands queued in the SATA controller or at any attached device.

When the host application wishes to bring the SATA controller offline it clears the HC_ON control bit. This acts as a request to the SATA controller to go offline. The SATA controller will signal it has completed this operation by clearing the HS_ON bit in the HStatus register. If any commands are outstanding at SATA controller or device then the SATA controller will wait for the operation to complete before going offline.

If the host application wishes to bring the SATA controller offline regardless of the queue status, it clears the HC_ON bit a second time when the controller has set the HS_OFF bit of the HStatus register.

When the host application wishes to bring the SATA controller online, it sets the HC_ON control bit. This acts as a request to the SATA controller to go online. The SATA controller will signal it has completed this operation by setting the HS_ON status bit.

22.7 Command header

Each entry in the command header table consists of the structure shown in this figure.

NOTE

In this chapter, "word" refers to 4 bytes or 32 bits.

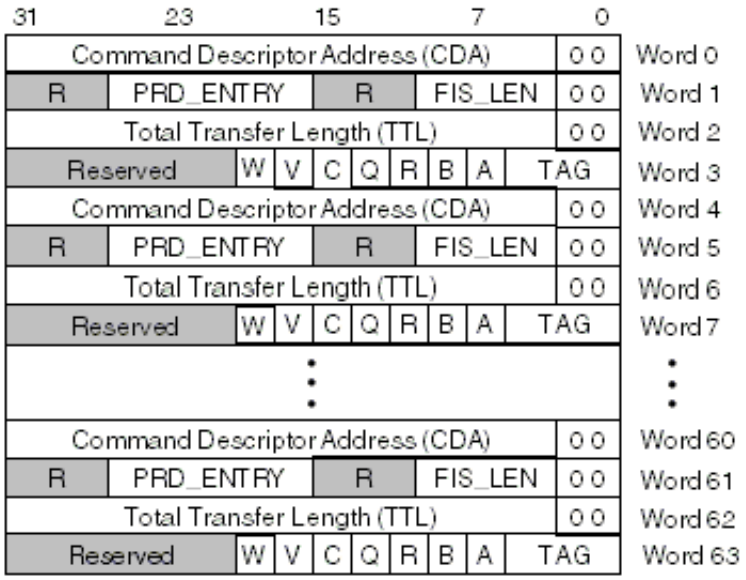


Figure 22-65. Command header

This table shows the command header word 0-data base address.

Table 22-64. Command header word 0-data base address

Bit	Name	Description
31-2	CDA	Command descriptor base address. Indicates the 32-bit physical address of the command descriptor block. The block must be word-aligned, indicated by bits 1-0 being reserved.
1-0	-	Reserved

This table shows the command header word 1-FIS_LEN.

Table 22-65. Command header word 1-FIS_LEN

Bit	Name	Description
31-22	-	Reserved
21-16	PRD_ENTRY	Number of PRD entries including indexed entries.
7	-	Reserved
6-2	FIS_LEN	FIS length. This is a 5-bit word count of the total length of the control or vendor-specific FIS to transfer.
1-0	-	Reserved

This table shows the command header word 2-data base address.

Table 22-66. Command header word 2-data base address

Bit	Name	Description
31-2	TTL	Total transfer length. This is a 30-bit word count of the total length of the data transfer. It is used to detect overruns/underruns between the transfer lengths programmed in the command and the PRDT.
1-0	-	Reserved

This table shows the command header word 3-description information.

Table 22-67. Command header word 3-description information

Bit	Name	Description
31-12	-	Reserved
11	W	Non-post last data write to memory. This bit gives the assurance that during disk read commands, last data was updated in memory before it receives an interrupt concerning the command completion. 0 Before generating the command completion interrupt SATA controller should not wait for acknowledgement from systemduring the last data write to memory of a read command. 1 Before generating the command completion interrupt SATA controller should wait for acknowledgement from system during the last data write to memory of a read command.
10	V	Vendor BIST. When this bit is set, it indicates that the command is a Vendor BIST, thus FIS will loop back at the PHY local test.
9	C	Snoop enable during all descriptor read/write operations associated with this command.
8	Q	Queued. Command is an FPDMA queued command.
7	R	Reset. The command is a SRST or device reset.
6	B	BIST. The command will require the host to enter BIST mode.
5	A	ATAPI command. The command is an ATAPI command and thus will require that the host uses the ATAPI portion of the command descriptor to issue the command. The CFIS also has to be written with the packet command.
4-0	TAG	The 5-bit TAG assigned by software for command tracking. It is the same as the value written to the command register host-to-device.

22.7.1 Command descriptor

Each entry in the command list points to a structure called the command descriptor.

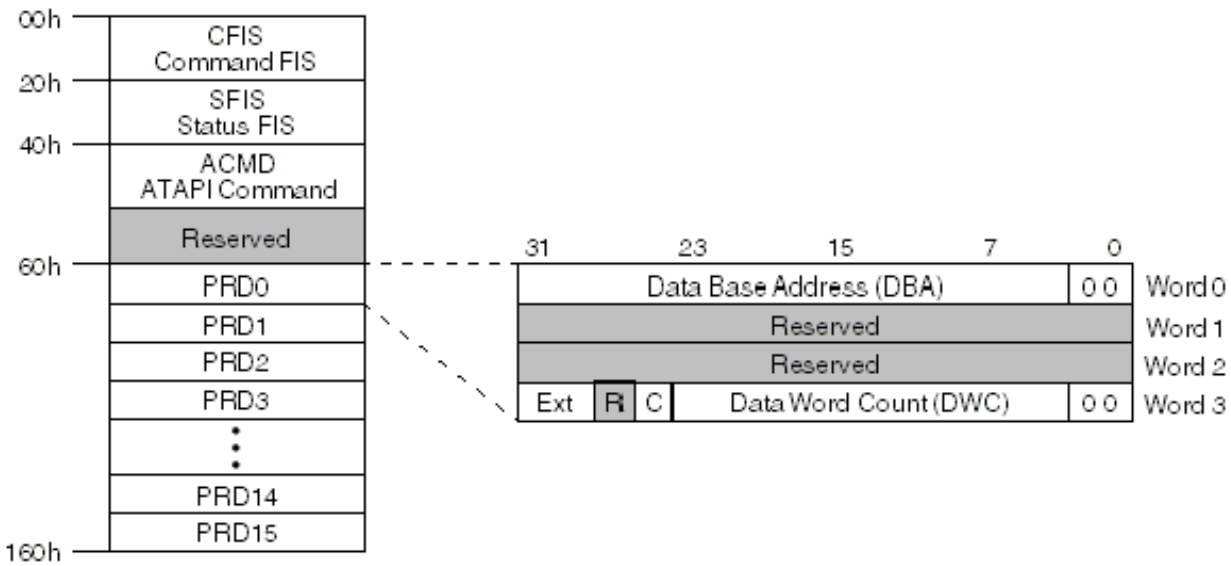


Figure 22-66. Command descriptor

22.7.1.1 Command FIS non-queued commands (CFIS)

Command FIS is a software constructed FIS. For data transfer operations, this is the H2D register FIS format as specified in the Serial ATA 2.6 standard.

The SATA controller fetches this from memory and sends the appropriate amount of data to the attached port. If a port multiplier is attached, this field must have the port multiplier port number in the FIS itself. CFIS lengths are two to eight words and must be in word granularity. A typical command FIS is shown in this figure.

Table 22-68. Register host-to-device

	Register FIS host-to-device			
	31 24	23 16	15 12	11 8	7 0
0	Features	Command	C R R R	PM port	FIS type (27h)
1	Device	LBA high	LBA mid		LBA low
2	Features (exp)	LBA high (exp)	LBA mid (exp)		LBA low (exp) (0)
3	Control	Reserved (0)	Sector count (exp)		Sector count
4	Reserved (0)	Reserved (0)	Reserved (0)		Reserved (0)

22.7.1.2 Command FIS first party DMA commands NCQ

This figure shows register host-to-device first party DMA commands NCQ.

The shaded components show where this FIS differs for the non-NCQ register host-to-device FIS.

Table 22-69. Register host-to-device first party DMA commands NCQ

	Register FIS host to device-Read/write FPDMA queued			
	31 … 24	23 … 16	15 … 12	11 … 8	7 … 3	2	0
0	Features	Command	C	R	R	R	PM port	FIS type (27h)
	Sector count 7:0						
1	Device	LBA high	LBA mid	LBA low
2	Features (exp)	LBA high (exp)	LBA mid (exp)	LBA low (exp) (0)
	Sector count 15:8			
3	Control	Reserved (0)	Sector count (exp)	Sector count
			Reserved (0)	TAG
4	Reserved (0)	Reserved (0)	Reserved (0)	Reserved (0)

22.7.1.3 Status FIS (SFIS)

This FIS is created in hardware. For normal operations, this is the D2H register FIS format as specified in the Serial ATA 2.6 standard.

SFIS lengths are two to eight words and must be of word granularity. A typical status FIS is shown in Table 22-70.

Table 22-70. Register device-to-host

	Register FIS device to host			
	31 … 24	23 … 16	15 … 12	11 … 8	7 … 0
0	Error	Status	R	I	R	R	PM port	FIS type (34h)
1	Device	LBA high	LBA mid	LBA low
2	Reserved (0)	LBA high (exp)	LBA mid (exp)	LBA low (exp) (0)
3	Reserved (0)	Reserved (0)	Sector count (exp)	Sector count
4	Reserved (0)	Reserved (0)	Reserved (0)	Reserved (0)

22.7.1.4 ATAPI command (ACMD)

This is a software constructed region of three or four words in length that contains the ATAPI command to transmit if the 'A' bit is set in the command header.

The ATAPI command must be either 12 or 16 bytes in length. The length transmitted by the SATA IP is determined by the PIO setup FIS that is sent by the device requesting the ATAPI command.

22.7.1.5 Physical region descriptor table (PRDT)

This is a software constructed table of addresses to use to complete the data transfer. Up to 16 structures can be supported in the current command descriptor.

The format of the address entry is defined by the "Block vector structures for passing segmented data type of the IEEE Std. 1212.1-1993". The total definable length supported in the 16 entries is 64 Mbytes.

This table shows the PRDT word 0-data base address.

Table 22-71. PRDT word 0-data base address

Bit	Name	Description
31-2	DBA	Data base address. Indicates the 32-bit physical address of the data block. The block must be word aligned, indicated by bits 1-0 being "reserved, must be 00."
1-0	-	Reserved

This table shows the PRDT word 3-description information.

Table 22-72. PRDT Word 3-description information

Bit	Name	Description
31	EXT	If the extension flag is set to 1, then the DBA field contains the address of the extension segment, and the DWC field contains the size of this extension segment (this is called an "indirect descriptor").
30-29	-	Reserved
28	C	Data snoop enable bit. When this bit is set, all data read/write operations associated with the PRD entry for this command will be snoopable.
27-26	-	Reserved
25-2	DDC	Data DWord count. A 0-based value that Indicates the length, in DWords, of the data block. A maximum length of 32 Mbytes may exist for any entry. Bits 1-0 of this field must always be 0 to indicate that size is in 4-byte words. A value of 24'b0 indicates a full 32 Mbytes transfer.
1-0	-	Reserved

22.7.2 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 PHY that supports serial loopback, allows for the test of the SATA controller operation.

The 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 on SATA1, lane A, the software performs the following operations:

1. Builds the vendor-specific header and descriptor as detailed in the vendor-specific BIST header and descriptor.
2. Modifies the SerDes control register 2to place the PHY into analog loopback mode.
3. Issues this command to the SATA controller.
4. When the SATA controller indicates the command has completed, the software examines the contexts of the command descriptor's stats field. If the pre-programmed values are present, the test has passed.
5. The contents of the vendor-specific FIS to be created in memory are as follows:
   - Build command header in the memory. See Table 22-73 :

Table 22-73. Vendor BIST test-command header

Word number	Hexadecimal value
Word 0	CDA
Word 1	0x0000_000C
Word 2	All zeros
Word 3	0x0000_0400

   - Build command descriptor in the memory. See Table 22-74.

Table 22-74. Vendor BIST test-command descriptor

Word number	Hexadecimal value	Comments
Word 0	0x0001_0058	-
Word 1	0xAAAA_A034	AAAA_A is the first test pattern (can be any value)
Word 2	0xBBBB_B034	BBBB_B is the second test pattern (can be any value)
Word 3	Reserved	Reserved, must be all zeros
Word 4	Reserved	Reserved, must be all zeros
Word 5	Reserved	Reserved, must be all zeros
Word 6	Reserved	Reserved, must be all zeros
Word 7	Reserved	Reserved, must be all zeros

The values of A and B can be filled in by the user with the pattern to test as shown in the figure.

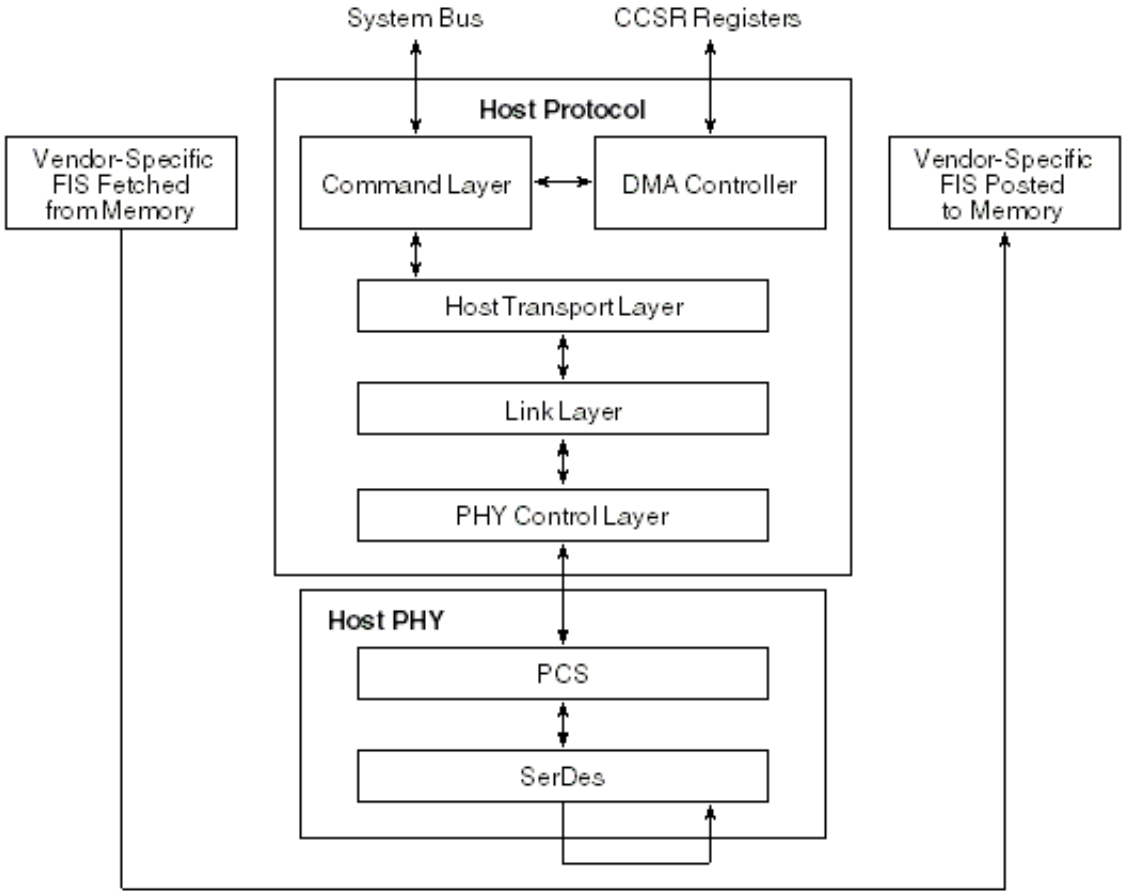


Figure 22-67. Vendor-specific BIST operation

22.8 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 FIS.

On the transmit path, the transport layer frames the FIS's placed into the Tx FIFO. The FIS's are framed based on a programmed length for non-data FIS and are 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 acceptance of the transmitted FIS.

On the reception path, the transport layer deframes the FIS's and places them 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 the end-of-frame (EOF) is signaled by the link layer and the FIS has been received with a good CRC. For a short vendor-specific FIS, the FIS is considered as a non-data FIS. For a 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 word (header) is written into the FIFO.

The receive FIFO is written with data contained in the FIS sent by the link layer. When the data is stable at the output of the receive FIFO, 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 will in turn send HOLD primitives to the far end. 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 retainers in between the host and device.

The transmit FIFO is written with data to be sent in the FIS transferred by the DMA controller. When the data is stable at the output of the transmit FIFO, 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 will in turn send HOLD primitives to the far end.

22.9 Link layer overview

The function of the SATA link layer is to interface between the transport and physical 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. The link layer can work at either SATA first-generation (1.5 Gbps) or second-generation 2 (3 Gbps) speeds.

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 can then begin to take data in the form of words 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 EOF primitive at the end. The link layer scrambles the contents of the frame, including the calculated CRC, but excluding the SOF and EOF diameters and any other embedded primitives. The 8B/10B encoding of the data is done in the PHY layer. 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 with 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 in the PHY layer. Next, the link layer removes all primitives including the SOF and EOF diameters. 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 (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 words, or more frequently if programmed by the user.

22.9.1 Link layer functionality

The link layer is composed of a number of functions:

- Link layer state machines
- Frame content scrambler and descrambler
- CRC generation and checking
- Bus interfaces to PHY and transport layer
- CONT primitive processing
- ALIGN insertion on transmit
- Debug functionality
- BIST support

The four link layer state machines are described in the following sections.

22.9.1.1 Link idle state machine

The link idle state machine is responsible for detecting a transmit request from the transport layer or a frame reception request from the far end.

The state machine arbitrates whether these two events coincide. The SATA specification defines that the host end always backs down in this case. Furthermore, this machine interprets power mode change requests from both the transport and PhyCtrl layers and initiates actions to enable the power mode change. Power mode change can only occur if the feature is enabled via the PhyCtrlCfg register LPB_EN bits. Finally, this state machine also detects the negation and assertion of PHY_READY from the PHY and notifies the transport layer of the change.

22.9.1.2 Transmit state machine

This state machine is responsible for frame transmission to the PHY.

The state machine places the SOF and 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 word from the transport layer and uses this as the next word of the frame. The link layer also inserts a pair of ALIGN primitives every 254 words of frame data. Finally, at the end of the frame transmission, the state machine waits for status from the far end link layer via received R_OK or R_ERR primitives. If the far end received the frame correctly, the local link layer signals TX_OK to the transport layer; otherwise, it signals TX_NOT_OK to the transport layer.

The transmit state machine also partakes in flow control actions, if necessary, during packet transmission. If the transport layer cannot supply a new word and the frame is not finished, the transmit state machine responds by sending HOLD primitives until 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. The first is 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. The second is when the transport layer wishes to send a control register frame signaled via 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 the link layer detects the following conditions:

- PHY_READY negates
- SYNC primitive is received during frame transmission

22.9.1.3 Receive state machine

This state machine is responsible for frame reception from the PHY layer.

The state machine removes the SOF and 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 word from the Phy layer and uses this as the next word 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 via RX_CRC_OK and

RX_CRC_NOT_OK. 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 was detected during reception. If an error was detected, it sends a 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 word, (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 via 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 the link layer detects the following conditions:

- PHY_READY negates
- SYNC primitives is received during frame transmission
- WTRM primitive is received before EOF

22.9.1.4 Power mode change state machine

This state machine is responsible for handling change of power mode requests.

These requests can come from the command layer superset registers or the far end. 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 a partial or slumber state.

A write to the SControl register SPM field or reception of a COMWAKE from the far end will initiate a resume to active power mode.

If the link layer receives a PMREQ_P/PMREQ_S primitive from a peer link layer and is enabled to perform power management modes (SControl IPM bits are cleared), it responds by sending at least four PMACK primitives. A write to the SControl register SPM field or reception of a COMWAKE from the far end will initiate a resume to active power mode.

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, SError[S] = 1.

22.9.1.5 Frame content scrambler and descrambler

There are two separate scramblers used in the SATA controller, one for the data payload and the other for repeated primitive suppression.

The contents of each word of data (excluding all primitives) between SOF and EOF must be scrambled before 8B/10B encoding. Scrambling is performed on word quantities according to the following polynomial:

G(X) = X16 + X15 + X13 + X4 + 1

The scrambler is initialized with a seed value of 0xFFFF at each SOF transmission and rolls over every 2048 words. 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, then the intervening data between the last CONT primitive and a subsequent primitive must be scrambled also. 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. If a CONT primitive is transmitted or received during a frame transfer, then the current data payload scrambler value at the last word is held.

When payload data is received by the link layer it is descrambled 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.

22.9.1.6 CRC generator and checker

A 32-bit CRC is calculated on the data contents of each frame and is inserted in the word 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 word 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 0x52325032 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, then 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, then the current CRC scrambler value at the last word is held.

22.9.1.7 8B/10B encode and decode

All data and primitives must be encoded prior to transmission on the line.

The 8B/10B encode/decode occur in the PHY layer.

22.9.1.8 CONT primitive processing

Using the CONT primitive, the link layer is capable of replacing repetitive primitive streams with scrambled data.

This reduces EMI emissions because 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 then 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 will cause the last received valid primitive to be implied to 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 the transmission and reception of CONT primitives.

22.9.1.9 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 words. 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 to as much as two ALIGNS per one word; for example, ALIGN, ALIGN, data, ALIGN, ALIGN. In addition, the SEND_4_ALIGNS bit can be set to instruct the link layer to send four ALIGNs at a time instead of two.

22.9.1.10 Debug functionality

There are a number of useful features designed into the link layer to aid debug, as follows:

- The align insertion rate can be increased using the ALIGN_RATE register field in the command layer.
- Four error counters can be monitored by issuing register reads to the command layer: the disparity error counter, the code error counter, the PHY internal error counter, and the control character error counter.
- A number of configuration bits in the command layer can be used to override normal primitive insertion. For example,
  - Set PRIM_OVR_STATE = (L_SendHold state (16))
  - Set PRIM = 0xb5b5957c, that is, a SYNC primitive
  - During the transfer, set PRIM_OVRD_EN = 1
- When the link layer detects a rising edge on PRIM_OVRD_EN, it will insert one SYNC primitive into the datastream in place of the HOLD, when the LINK_STATE reaches the L_SendHold state. Only one HOLD primitive will be overridden; the PRIM_OVRD_EN must be cleared and written to again to force another override to occur.

22.9.1.11 BIST support

The transmit and receive subblocks 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. 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. 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 PHY to detect the BIST data pattern is as specified in the BIST activate FIS. When it detects the correct data sequence, the HStatus[BIST_Err] is deasserted. The BIST_Err bit will stay deasserted unless an error occurs in the datastream from the far end.

22.10 PHY control layer overview

The PHY control layer operates between the PHY and link layers.

On receive, the PHY control layer converts the 16-bit parallel data from the PHY to a 32-bit word, 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 word. The PHY control layer takes in the per-byte error signals and the per-byte control/data bits output by the PHY and converts them into 4-bit buses, with each bit of the bus corresponding to a byte in the word.

On transmit, the PHY control layer takes in the 32-bit transmit data from the link layer and converts it to 16 bits of data which it presents to the PHY. The control/data bit from the link layer (which is always assumed to be associated with the lowest byte position of the transmit word) is also passed onto the PHY with the appropriate word.

22.11 SATA initialization/application information

22.11.1 SATA controller initialization steps

These steps bring the SATA controller online, synchronize the SATA controller with the attached device, and issue typical command for execution.

1. Write HControl[HC_ON] = 1 to bring the SATA controller online.
2. Poll the HStatus[HS_ON] till HS_ON = 1, indicating that the controller is online.
3. Poll the SStatus[DET] till DET = 3'b0011 meaning that the device presence is detected and PHY communication is established. In this state, SStatus[SPD] indicates the negotiated communication speed.
4. To read the device's signature, poll HStatus[SIG_UPD] till it goes up. Read the signature from the SIG register.
5. Initialize the CHBA register to point to the command header block.
6. Build a command header block in memory. Refer to Command header.
7. Build a command descriptor block in memory. Refer to Command descriptor.
8. Build a number of PRD tables in memory as defined by the PRD_NUM field in the command header. Refer to Physical region descriptor table (PRDT).
9. Initialize the CQPMP register with the device's PM number. If the port multiplier is not used, clear this field.
10. Poll the CQR[CQ*n*] to determine which command can be issued.

11. After CQ*n* is determined, write 1 to CQR[CQ*n*] to issue this command and start execution.
12. Poll the CCR[CC*n*] till CC*n* goes up, indicating that the command is completed.

The following example presents the structure of descriptors to issue the ReadDMA command:

- Build the command header in memory.

Table 22-75. Read DMA command-command header

Word number	Hexadecimal value	Description
Word 0	CDA	Pointer to memory where command descriptor begins
Word 1	0x0002_0014	Two PRD tables contain the data, FIS length = 20 bytes
Word 2	0x0000_0200	Length of data associated with this command = 0x200
Word 3	All zeros	Tag = 0

- Build command descriptor in memory.

Table 22-76. Read DMA command-command descriptor

Word number	Hexadecimal value	Description
Word 0	0x00C8_8027	Command = Read DMA
Word 1	0x0000_0010	LBA = 24'h10
Word 2	All zeros	LBA(exp) = 24'h0
Word 3	0x0000_0001	Sector count = 1
Word 4	All zeros	Reserved

- Build two PRD entries in memory.

Table 22-77. Read DMA command-PRD entries

Word number	Hexadecimal value	Description
Word 0	PRD1	Address of first portion of data.
Word 1	All zeros	Reserved
Word 2	All zeros	Reserved
Word 3	0x0000_0100	PRD1 contains 0x100 bytes of data
Word 4	PRD2	Address of second portion of data.
Word 5	All zeros	Reserved
Word 6	All zeros	Reserved
Word 7	0x0000_0100	PRD2 contains 0x100 bytes of data

- Fill the PRD1 and PRD2 with user-defined data.

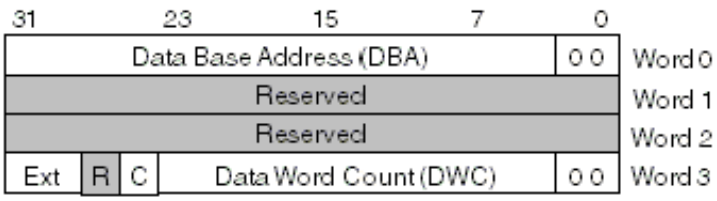


Figure 22-68. PRD entry

Chapter 23
DMA Controller

23.1 DMA overview

This chapter describes the DMA controller offered on this chip.

It describes a single DMA controller, which is instantiated three times on this chip. As such, all functionality is identical between the three controllers with the exception of the register offsets, as noted in the DMA register summary table , and the external signals, as noted in DMA signals table.

The DMA controller transfers blocks of data between the many interface and functional modules of this chip, independent of the cores or external hosts. It has eight high-speed DMA channels. Both the cores and external devices can initiate DMA transfers. The channels are capable of complex data movement and advanced transaction chaining. Operations such as descriptor fetches and block transfers are initiated by each channel. A channel is selected by the arbitration logic and information is passed to the source and destination control blocks for processing. The source and destination blocks generate read and write requests to the address tenure engine, which manages the DMA master port address interface. After a transaction is accepted by the master port, control is transferred to the data tenure engine that manages the read and write data transfers. A channel remains active in the shared resources for the duration of the data transfer unless the allotted bandwidth per channel is reached.

This figure shows the block diagram of the DMA controller.

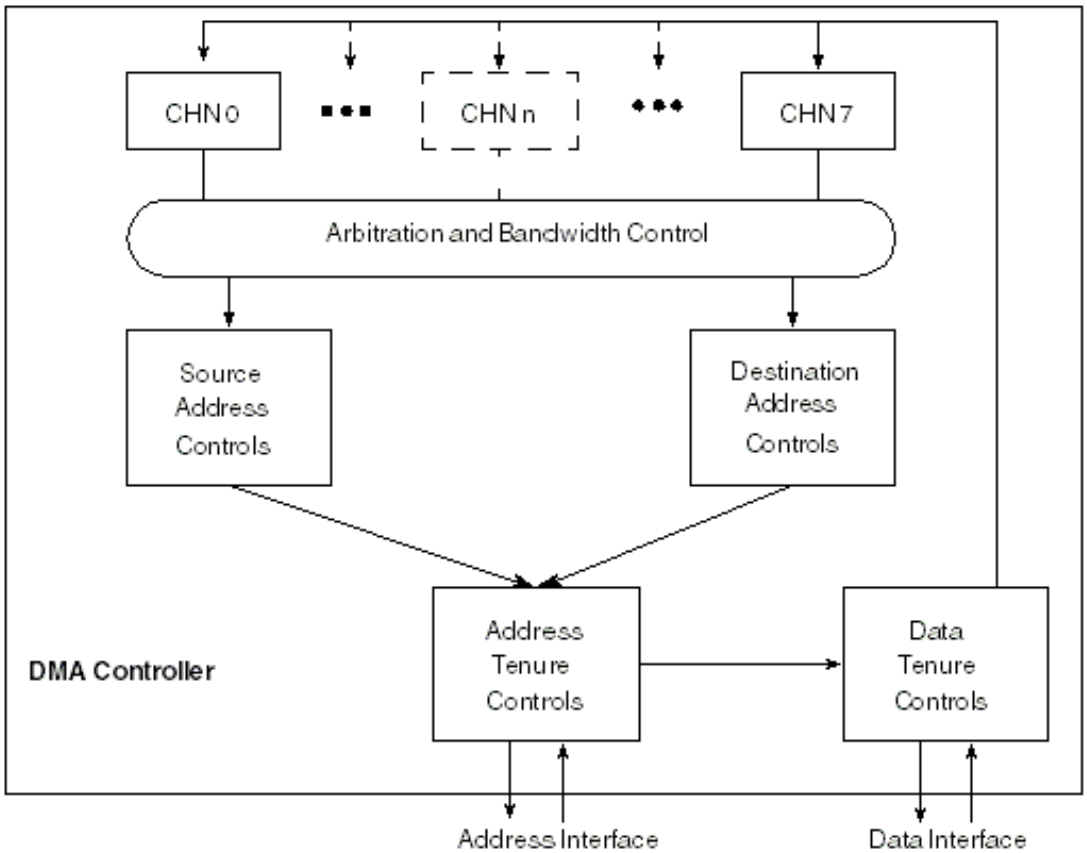


Figure 23-1. DMA block diagram

23.1.1 DMA features summary

The DMA controller offers the following features:

- Eight high-speed/high-bandwidth channels accessible by local and remote masters
- Basic DMA operation modes (direct, simple chaining)
- Extended DMA operation modes (advanced chaining and stride capability)
- Cascading descriptor chains
- Misaligned transfers
- Programmable bandwidth control between channels
- Up to 256 bytes for DMA sub-block transfers to maximize performance
- Interrupt on error and completed segment, list, or link
- Externally-controlled transfer using DMA_DREQ_B, DMA_DACK_B, and DMA_DDONE_B.

NOTE

These external control signals are available on DMA1 and DMA2. They are not supported by DMA3. For more details see Global Source and Target IDs

23.1.2 DMA modes of operation

The DMA module has two modes of operation: basic and extended.

Basic mode is the DMA legacy mode, which does not support advanced features. Extended mode supports advanced features such as striding and flexible descriptor structures.

These two basic modes allow users to initiate and end DMA transfers in various ways. Table 23-1 summarizes the relationship between the modes and the following features:

- Direct mode-No descriptors are involved. Software must initialize the required fields as described in Table 23-2 before starting a transfer.
- Chaining mode-Software must initialize descriptors in memory and the required fields as described in Table 23-2 before starting a transfer.
- Single-write start mode-The DMA process can be started using a single-write command to either the descriptor address register in one of the chaining modes or the source/destination address registers in one of the direct modes.
- External control capability-This allows an external agent to start, pause, and check the status of a DMA transfer that has already been initialized.
- Channel continue capability-The channel continue capability allows software the flexibility of having the DMA controller start with descriptors that have already been programmed while software continues to build more descriptors in memory.
- Channel abort capability-The software can abort a previously initiated transfer by setting the bit MR*n*[CA]. The DMA controller terminates all outstanding transfers initiated by the channel without generating any errors before entering an idle state.

Table 23-1. Relationship of modes and features

Mode	Mode with one additional feature	Mode with two additional features
B (Basic)	BD (basic direct)	BDS (BD single-write start)
		BDE (BD external control)
	BC (basic chaining)	BCE (BC external control)
		BCS (BC single-write start)
Ext (Extended)	ExtD (extended direct)	ExtDS (ExtD single-write start)
		ExtDE (ExtD external control)
	ExtC (extended chaining)	ExtCE (ExtC external control)

Table continues on the next page...

Table 23-1. Relationship of modes and features (continued)

Mode	Mode with one additional feature	Mode with two additional features
		ExtCS (ExtC single-write start)

The following table describes bit settings required for each DMA mode of operation.

Table 23-2. DMA mode bit settings

Modes with features	MRn[XFE]	MRn[CTM]	MRn[SRW]	MRn[CDSM_SWSM]	MRn[EMS_EN]
Basic direct modes					
Basic direct	0	1	0	0	0
Basic direct external control	0	1	0	0	1
Basic direct single-write start	0	1	1	1 or 0	0
Basic chaining modes					
Basic chaining	0	0	Reserved	0	0
Basic chaining external control	0	0	Reserved	0	1
Basic chaining single-write start	0	0	Reserved	1	0
Extended direct modes					
Extended direct	1	1	0	0	0
Extended direct external control	1	1	0	0	1
Extended direct single-write start	1	1	1	1 or 0	0
Extended chaining modes					
Extended chaining	1	0	Reserved	0	0
Extended chaining external control	1	0	Reserved	0	1
Extended chaining single-write start	1	0	Reserved	1	0

See DMA functional description for details on these modes.

This figure shows the general DMA operational flow chart.

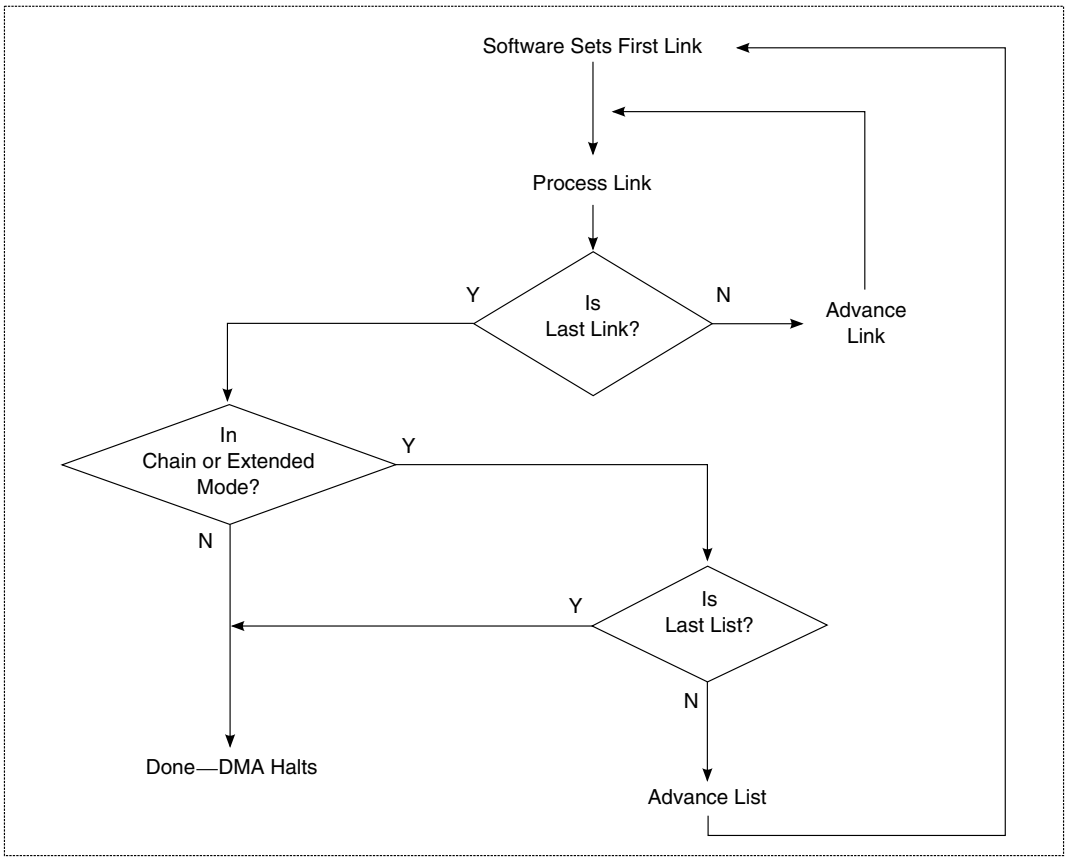


Figure 23-2. DMA operational flow chart

23.2 DMA external signal description

This section describes the external DMA signals.

23.2.1 Signal overview

This figure summarizes the DMA controller signals.

NOTE

External Control is supported for DMA channel 0 only.

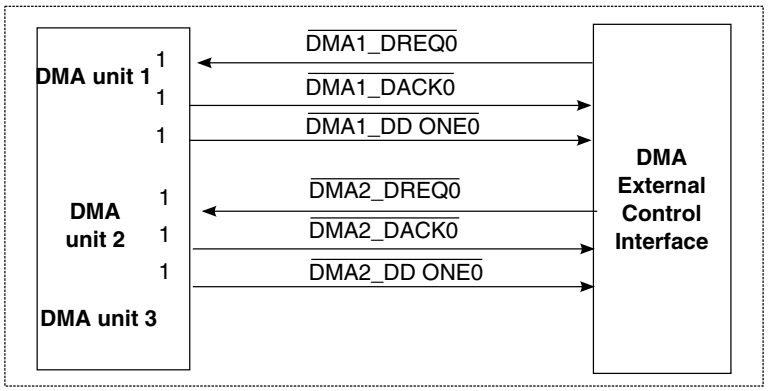


Figure 23-3. DMA signal summary

Note the DMA signals are multiplexed with a variety of other functionality.

23.2.2 DMA signal descriptions

This table describes the external DMA control signals. These signals are only utilized when operating in external master mode.

See External control mode transfer.

NOTE

External Control is supported on channel 0 only

Table 23-3. DMA signals-detailed signal descriptions

Signal	I/O	Description	
DMA_DREQ_B	I	DMA request. Indicates the start of a DMA transfer or a restart from a paused request. Assertion of DMA_DREQ_B causes MRn[CS] to be set, thereby activating the corresponding DMA channel.	
		State Meaning	Asserted-Assertion of DMA_DREQ_B while DMA_DACK_B is negated causes a new transfer to start OR resumes a paused transfer if the EMP_EN bit is set. Assertion while DMA_DACK_B is asserted results in an illegal condition. Negated-Negation while DMA_DACK_B is asserted has no effect. Negation before the assertion of DMA_DACK_B results in an illegal condition.
		Timing	Assertion-Can be asserted asynchronously Negation- Must remain asserted at least until the assertion of the corresponding DMA_DACK_B
DMA_DACK_B	O	DMA acknowledge. Indicates that a DMA transfer is currently in progress	
		State Meaning	Asserted-Indicates that a DMA transfer is currently in progress. Asserted after the assertion of DMA_DREQ_B to indicate the start of a transfer Negated-Negated after finishing a complete transfer or after entering a paused state if MRn[EMP_EN] is set
		Timing	Assertion-Asynchronous assertion; asserted for more than three system clocks Negation-Asynchronous negation; negated for more than three system clocks

Table continues on the next page...

Table 23-3. DMA signals-detailed signal descriptions (continued)

Signal	I/O	Description	
DMA_DDONE_B	O	DMA done. Indicates that a DMA transfer is complete	
		State Meaning	Asserted-Indicates transfer completion. SRn[CB] is clear. Note, however, that write data may still be queued at the target interface or in the process of transfer on an external interface. Negated-Indicates that the current transfer is in process
		Timing	Assertion-Always asserts asynchronously after the negation of the final DMA_DACK_B to indicate completion of a transfer. For a paused transfer, DMA_DDONE_B is asserted asynchronously after the negation of the final DMA_DACK_B. Negation-Negated asynchronously after the assertion of DMA_DREQ_B for the next transfer

NOTE

These signals are available on DMA1 and DMA2. They are not supported by DMA3. For more details see Signals Overview

23.3 DMA controller memory map

This section provides a detailed description of all accessible DMA memory and registers . The descriptions include individual, bit-level descriptions and reset states of each register. Undefined 4-byte address spaces within the map are reserved.

This table lists the DMA registers and their offsets. Note that the full register address is comprised of the programmable CCSRBAR together with the fixed DMA block base address and offset listed in table below.

DMA memory map

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
10_0100	DMA mode register (DMA1_MR0)	32	R/W	0800_0000h	23.3.1/1453
10_0104	DMA status register (DMA1_SR0)	32	R/W	0000_0000h	23.3.2/1457
10_0108	DMA current link descriptor extended address register (DMA1_ECLNDAR0)	32	R/W	0000_0000h	23.3.3/1458
10_010C	DMA current link descriptor address register (DMA1_CLNDAR0)	32	R/W	0000_0000h	23.3.4/1459
10_0110	DMA source attributes register (DMA1_SATR0)	32	R/W	0000_0000h	23.3.5/1461
10_0114	DMA source address register (DMA1_SAR0)	32	R/W	0000_0000h	23.3.6/1462
10_0118	DMA destination attributes register (DMA1_DATR0)	32	R/W	0000_0000h	23.3.7/1463

Table continues on the next page...

DMA memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
10_011C	DMA destination address register (DMA1_DAR0)	32	R/W	0000_0000h	23.3.8/1464
10_0120	DMA byte count register (DMA1_BCR0)	32	R/W	0000_0000h	23.3.9/1465
10_0124	DMA extended next link descriptor address register (DMA1_ENLNDAR0)	32	R/W	0000_0000h	23.3.10/ 1466
10_0128	DMA next link descriptor address register (DMA1_NLNDAR0)	32	R/W	0000_0000h	23.3.11/ 1466
10_0130	DMA extended current list descriptor address register (DMA1_ECLSDAR0)	32	R/W	0000_0000h	23.3.12/ 1467
10_0134	DMA current list descriptor address register (DMA1_CLSDAR0)	32	R/W	0000_0000h	23.3.13/ 1468
10_0138	DMA extended next list descriptor address register (DMA1_ENLSDAR0)	32	R/W	0000_0000h	23.3.14/ 1469
10_013C	DMA next list descriptor address register (DMA1_NLSDAR0)	32	R/W	0000_0000h	23.3.15/ 1469
10_0140	DMA source stride register (DMA1_SSR0)	32	R/W	0000_0000h	23.3.16/ 1470
10_0144	DMA destination stride register (DMA1_DSR0)	32	R/W	0000_0000h	23.3.17/ 1471
10_0180	DMA mode register (DMA1_MR1)	32	R/W	0800_0000h	23.3.1/1453
10_0184	DMA status register (DMA1_SR1)	32	R/W	0000_0000h	23.3.2/1457
10_0188	DMA current link descriptor extended address register (DMA1_ECLNDAR1)	32	R/W	0000_0000h	23.3.3/1458
10_018C	DMA current link descriptor address register (DMA1_CLNDAR1)	32	R/W	0000_0000h	23.3.4/1459
10_0190	DMA source attributes register (DMA1_SATR1)	32	R/W	0000_0000h	23.3.5/1461
10_0194	DMA source address register (DMA1_SAR1)	32	R/W	0000_0000h	23.3.6/1462
10_0198	DMA destination attributes register (DMA1_DATR1)	32	R/W	0000_0000h	23.3.7/1463
10_019C	DMA destination address register (DMA1_DAR1)	32	R/W	0000_0000h	23.3.8/1464
10_01A0	DMA byte count register (DMA1_BCR1)	32	R/W	0000_0000h	23.3.9/1465
10_01A4	DMA extended next link descriptor address register (DMA1_ENLNDAR1)	32	R/W	0000_0000h	23.3.10/ 1466
10_01A8	DMA next link descriptor address register (DMA1_NLNDAR1)	32	R/W	0000_0000h	23.3.11/ 1466
10_01B0	DMA extended current list descriptor address register (DMA1_ECLSDAR1)	32	R/W	0000_0000h	23.3.12/ 1467
10_01B4	DMA current list descriptor address register (DMA1_CLSDAR1)	32	R/W	0000_0000h	23.3.13/ 1468
10_01B8	DMA extended next list descriptor address register (DMA1_ENLSDAR1)	32	R/W	0000_0000h	23.3.14/ 1469
10_01BC	DMA next list descriptor address register (DMA1_NLSDAR1)	32	R/W	0000_0000h	23.3.15/ 1469
10_01C0	DMA source stride register (DMA1_SSR1)	32	R/W	0000_0000h	23.3.16/ 1470

Table continues on the next page...

DMA memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
10_01C4	DMA destination stride register (DMA1_DSR1)	32	R/W	0000_0000h	23.3.17/ 1471
10_0200	DMA mode register (DMA1_MR2)	32	R/W	0800_0000h	23.3.1/1453
10_0204	DMA status register (DMA1_SR2)	32	R/W	0000_0000h	23.3.2/1457
10_0208	DMA current link descriptor extended address register (DMA1_ECLNDAR2)	32	R/W	0000_0000h	23.3.3/1458
10_020C	DMA current link descriptor address register (DMA1_CLNDAR2)	32	R/W	0000_0000h	23.3.4/1459
10_0210	DMA source attributes register (DMA1_SATR2)	32	R/W	0000_0000h	23.3.5/1461
10_0214	DMA source address register (DMA1_SAR2)	32	R/W	0000_0000h	23.3.6/1462
10_0218	DMA destination attributes register (DMA1_DATR2)	32	R/W	0000_0000h	23.3.7/1463
10_021C	DMA destination address register (DMA1_DAR2)	32	R/W	0000_0000h	23.3.8/1464
10_0220	DMA byte count register (DMA1_BCR2)	32	R/W	0000_0000h	23.3.9/1465
10_0224	DMA extended next link descriptor address register (DMA1_ENLNDAR2)	32	R/W	0000_0000h	23.3.10/ 1466
10_0228	DMA next link descriptor address register (DMA1_NLNDAR2)	32	R/W	0000_0000h	23.3.11/ 1466
10_0230	DMA extended current list descriptor address register (DMA1_ECLSDAR2)	32	R/W	0000_0000h	23.3.12/ 1467
10_0234	DMA current list descriptor address register (DMA1_CLSDAR2)	32	R/W	0000_0000h	23.3.13/ 1468
10_0238	DMA extended next list descriptor address register (DMA1_ENLSDAR2)	32	R/W	0000_0000h	23.3.14/ 1469
10_023C	DMA next list descriptor address register (DMA1_NLSDAR2)	32	R/W	0000_0000h	23.3.15/ 1469
10_0240	DMA source stride register (DMA1_SSR2)	32	R/W	0000_0000h	23.3.16/ 1470
10_0244	DMA destination stride register (DMA1_DSR2)	32	R/W	0000_0000h	23.3.17/ 1471
10_0280	DMA mode register (DMA1_MR3)	32	R/W	0800_0000h	23.3.1/1453
10_0284	DMA status register (DMA1_SR3)	32	R/W	0000_0000h	23.3.2/1457
10_0288	DMA current link descriptor extended address register (DMA1_ECLNDAR3)	32	R/W	0000_0000h	23.3.3/1458
10_028C	DMA current link descriptor address register (DMA1_CLNDAR3)	32	R/W	0000_0000h	23.3.4/1459
10_0290	DMA source attributes register (DMA1_SATR3)	32	R/W	0000_0000h	23.3.5/1461
10_0294	DMA source address register (DMA1_SAR3)	32	R/W	0000_0000h	23.3.6/1462
10_0298	DMA destination attributes register (DMA1_DATR3)	32	R/W	0000_0000h	23.3.7/1463
10_029C	DMA destination address register (DMA1_DAR3)	32	R/W	0000_0000h	23.3.8/1464
10_02A0	DMA byte count register (DMA1_BCR3)	32	R/W	0000_0000h	23.3.9/1465
10_02A4	DMA extended next link descriptor address register (DMA1_ENLNDAR3)	32	R/W	0000_0000h	23.3.10/ 1466

Table continues on the next page...

DMA memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
10_02A8	DMA next link descriptor address register (DMA1_NLNDAR3)	32	R/W	0000_0000h	23.3.11/ 1466
10_02B0	DMA extended current list descriptor address register (DMA1_ECLSDAR3)	32	R/W	0000_0000h	23.3.12/ 1467
10_02B4	DMA current list descriptor address register (DMA1_CLSDAR3)	32	R/W	0000_0000h	23.3.13/ 1468
10_02B8	DMA extended next list descriptor address register (DMA1_ENLSDAR3)	32	R/W	0000_0000h	23.3.14/ 1469
10_02BC	DMA next list descriptor address register (DMA1_NLSDAR3)	32	R/W	0000_0000h	23.3.15/ 1469
10_02C0	DMA source stride register (DMA1_SSR3)	32	R/W	0000_0000h	23.3.16/ 1470
10_02C4	DMA destination stride register (DMA1_DSR3)	32	R/W	0000_0000h	23.3.17/ 1471
10_0300	DMA general status register 0 (DMA1_DGSR0)	32	R	0000_0000h	23.3.18/ 1472
10_0400	DMA mode register (DMA1_MR4)	32	R/W	0800_0000h	23.3.19/ 1475
10_0404	DMA status register (DMA1_SR4)	32	R/W	0000_0000h	23.3.20/ 1479
10_0408	DMA extended current link descriptor address register (DMA1_ECLNDAR4)	32	R/W	0000_0000h	23.3.21/ 1480
10_040C	DMA current link descriptor address register (DMA1_CLNDAR4)	32	R/W	0000_0000h	23.3.22/ 1481
10_0410	DMA source attributes register (DMA1_SATR4)	32	R/W	0000_0000h	23.3.23/ 1483
10_0414	DMA source address register (DMA1_SAR4)	32	R/W	0000_0000h	23.3.24/ 1484
10_0418	DMA destination attributes register (DMA1_DATR4)	32	R/W	0000_0000h	23.3.25/ 1485
10_041C	DMA destination address register (DMA1_DAR4)	32	R/W	0000_0000h	23.3.26/ 1486
10_0420	DMA byte count register (DMA1_BCR4)	32	R/W	0000_0000h	23.3.27/ 1487
10_0424	DMA next link descriptor extended address register (DMA1_ENLNDAR4)	32	R/W	0000_0000h	23.3.28/ 1488
10_0428	DMA next link descriptor address register (DMA1_NLNDAR4)	32	R/W	0000_0000h	23.3.29/ 1488
10_0430	DMA extended current list descriptor address register (DMA1_ECLSDAR4)	32	R/W	0000_0000h	23.3.30/ 1489
10_0434	DMA current list descriptor address register (DMA1_CLSDAR4)	32	R/W	0000_0000h	23.3.31/ 1490
10_0438	DMA extended next list descriptor address register (DMA1_ENLSDAR4)	32	R/W	0000_0000h	23.3.32/ 1491

Table continues on the next page...

DMA memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
10_043C	DMA next list descriptor address register (DMA1_NLSDAR4)	32	R/W	0000_0000h	23.3.33/ 1491
10_0440	DMA source stride register (DMA1_SSR4)	32	R/W	0000_0000h	23.3.34/ 1492
10_0444	DMA destination stride register (DMA1_DSR4)	32	R/W	0000_0000h	23.3.35/ 1493
10_0480	DMA mode register (DMA1_MR5)	32	R/W	0800_0000h	23.3.19/ 1475
10_0484	DMA status register (DMA1_SR5)	32	R/W	0000_0000h	23.3.20/ 1479
10_0488	DMA extended current link descriptor address register (DMA1_ECLNDAR5)	32	R/W	0000_0000h	23.3.21/ 1480
10_048C	DMA current link descriptor address register (DMA1_CLNDAR5)	32	R/W	0000_0000h	23.3.22/ 1481
10_0490	DMA source attributes register (DMA1_SATR5)	32	R/W	0000_0000h	23.3.23/ 1483
10_0494	DMA source address register (DMA1_SAR5)	32	R/W	0000_0000h	23.3.24/ 1484
10_0498	DMA destination attributes register (DMA1_DATR5)	32	R/W	0000_0000h	23.3.25/ 1485
10_049C	DMA destination address register (DMA1_DAR5)	32	R/W	0000_0000h	23.3.26/ 1486
10_04A0	DMA byte count register (DMA1_BCR5)	32	R/W	0000_0000h	23.3.27/ 1487
10_04A4	DMA next link descriptor extended address register (DMA1_ENLNDAR5)	32	R/W	0000_0000h	23.3.28/ 1488
10_04A8	DMA next link descriptor address register (DMA1_NLNDAR5)	32	R/W	0000_0000h	23.3.29/ 1488
10_04B0	DMA extended current list descriptor address register (DMA1_ECLSDAR5)	32	R/W	0000_0000h	23.3.30/ 1489
10_04B4	DMA current list descriptor address register (DMA1_CLSDAR5)	32	R/W	0000_0000h	23.3.31/ 1490
10_04B8	DMA extended next list descriptor address register (DMA1_ENLSDAR5)	32	R/W	0000_0000h	23.3.32/ 1491
10_04BC	DMA next list descriptor address register (DMA1_NLSDAR5)	32	R/W	0000_0000h	23.3.33/ 1491
10_04C0	DMA source stride register (DMA1_SSR5)	32	R/W	0000_0000h	23.3.34/ 1492
10_04C4	DMA destination stride register (DMA1_DSR5)	32	R/W	0000_0000h	23.3.35/ 1493
10_0500	DMA mode register (DMA1_MR6)	32	R/W	0800_0000h	23.3.19/ 1475
10_0504	DMA status register (DMA1_SR6)	32	R/W	0000_0000h	23.3.20/ 1479

Table continues on the next page...

DMA memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
10_0508	DMA extended current link descriptor address register (DMA1_ECLNDAR6)	32	R/W	0000_0000h	23.3.21/ 1480
10_050C	DMA current link descriptor address register (DMA1_CLNDAR6)	32	R/W	0000_0000h	23.3.22/ 1481
10_0510	DMA source attributes register (DMA1_SATR6)	32	R/W	0000_0000h	23.3.23/ 1483
10_0514	DMA source address register (DMA1_SAR6)	32	R/W	0000_0000h	23.3.24/ 1484
10_0518	DMA destination attributes register (DMA1_DATR6)	32	R/W	0000_0000h	23.3.25/ 1485
10_051C	DMA destination address register (DMA1_DAR6)	32	R/W	0000_0000h	23.3.26/ 1486
10_0520	DMA byte count register (DMA1_BCR6)	32	R/W	0000_0000h	23.3.27/ 1487
10_0524	DMA next link descriptor extended address register (DMA1_ENLNDAR6)	32	R/W	0000_0000h	23.3.28/ 1488
10_0528	DMA next link descriptor address register (DMA1_NLNDAR6)	32	R/W	0000_0000h	23.3.29/ 1488
10_0530	DMA extended current list descriptor address register (DMA1_ECLSDAR6)	32	R/W	0000_0000h	23.3.30/ 1489
10_0534	DMA current list descriptor address register (DMA1_CLSDAR6)	32	R/W	0000_0000h	23.3.31/ 1490
10_0538	DMA extended next list descriptor address register (DMA1_ENLSDAR6)	32	R/W	0000_0000h	23.3.32/ 1491
10_053C	DMA next list descriptor address register (DMA1_NLSDAR6)	32	R/W	0000_0000h	23.3.33/ 1491
10_0540	DMA source stride register (DMA1_SSR6)	32	R/W	0000_0000h	23.3.34/ 1492
10_0544	DMA destination stride register (DMA1_DSR6)	32	R/W	0000_0000h	23.3.35/ 1493
10_0580	DMA mode register (DMA1_MR7)	32	R/W	0800_0000h	23.3.19/ 1475
10_0584	DMA status register (DMA1_SR7)	32	R/W	0000_0000h	23.3.20/ 1479
10_0588	DMA extended current link descriptor address register (DMA1_ECLNDAR7)	32	R/W	0000_0000h	23.3.21/ 1480
10_058C	DMA current link descriptor address register (DMA1_CLNDAR7)	32	R/W	0000_0000h	23.3.22/ 1481
10_0590	DMA source attributes register (DMA1_SATR7)	32	R/W	0000_0000h	23.3.23/ 1483
10_0594	DMA source address register (DMA1_SAR7)	32	R/W	0000_0000h	23.3.24/ 1484
10_0598	DMA destination attributes register (DMA1_DATR7)	32	R/W	0000_0000h	23.3.25/ 1485

Table continues on the next page...

DMA memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
10_059C	DMA destination address register (DMA1_DAR7)	32	R/W	0000_0000h	23.3.26/ 1486
10_05A0	DMA byte count register (DMA1_BCR7)	32	R/W	0000_0000h	23.3.27/ 1487
10_05A4	DMA next link descriptor extended address register (DMA1_ENLNDAR7)	32	R/W	0000_0000h	23.3.28/ 1488
10_05A8	DMA next link descriptor address register (DMA1_NLNDAR7)	32	R/W	0000_0000h	23.3.29/ 1488
10_05B0	DMA extended current list descriptor address register (DMA1_ECLSDAR7)	32	R/W	0000_0000h	23.3.30/ 1489
10_05B4	DMA current list descriptor address register (DMA1_CLSDAR7)	32	R/W	0000_0000h	23.3.31/ 1490
10_05B8	DMA extended next list descriptor address register (DMA1_ENLSDAR7)	32	R/W	0000_0000h	23.3.32/ 1491
10_05BC	DMA next list descriptor address register (DMA1_NLSDAR7)	32	R/W	0000_0000h	23.3.33/ 1491
10_05C0	DMA source stride register (DMA1_SSR7)	32	R/W	0000_0000h	23.3.34/ 1492
10_05C4	DMA destination stride register (DMA1_DSR7)	32	R/W	0000_0000h	23.3.35/ 1493
10_0600	DMA general status register 1 (DMA1_DGSR1)	32	R	0000_0000h	23.3.36/ 1494
10_1100	DMA mode register (DMA2_MR0)	32	R/W	0800_0000h	23.3.1/1453
10_1104	DMA status register (DMA2_SR0)	32	R/W	0000_0000h	23.3.2/1457
10_1108	DMA current link descriptor extended address register (DMA2_ECLNDAR0)	32	R/W	0000_0000h	23.3.3/1458
10_110C	DMA current link descriptor address register (DMA2_CLNDAR0)	32	R/W	0000_0000h	23.3.4/1459
10_1110	DMA source attributes register (DMA2_SATR0)	32	R/W	0000_0000h	23.3.5/1461
10_1114	DMA source address register (DMA2_SAR0)	32	R/W	0000_0000h	23.3.6/1462
10_1118	DMA destination attributes register (DMA2_DATR0)	32	R/W	0000_0000h	23.3.7/1463
10_111C	DMA destination address register (DMA2_DAR0)	32	R/W	0000_0000h	23.3.8/1464
10_1120	DMA byte count register (DMA2_BCR0)	32	R/W	0000_0000h	23.3.9/1465
10_1124	DMA extended next link descriptor address register (DMA2_ENLNDAR0)	32	R/W	0000_0000h	23.3.10/ 1466
10_1128	DMA next link descriptor address register (DMA2_NLNDAR0)	32	R/W	0000_0000h	23.3.11/ 1466
10_1130	DMA extended current list descriptor address register (DMA2_ECLSDAR0)	32	R/W	0000_0000h	23.3.12/ 1467
10_1134	DMA current list descriptor address register (DMA2_CLSDAR0)	32	R/W	0000_0000h	23.3.13/ 1468
10_1138	DMA extended next list descriptor address register (DMA2_ENLSDAR0)	32	R/W	0000_0000h	23.3.14/ 1469

Table continues on the next page...

DMA memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
10_113C	DMA next list descriptor address register (DMA2_NLSDAR0)	32	R/W	0000_0000h	23.3.15/ 1469
10_1140	DMA source stride register (DMA2_SSR0)	32	R/W	0000_0000h	23.3.16/ 1470
10_1144	DMA destination stride register (DMA2_DSR0)	32	R/W	0000_0000h	23.3.17/ 1471
10_1180	DMA mode register (DMA2_MR1)	32	R/W	0800_0000h	23.3.1/1453
10_1184	DMA status register (DMA2_SR1)	32	R/W	0000_0000h	23.3.2/1457
10_1188	DMA current link descriptor extended address register (DMA2_ECLNDAR1)	32	R/W	0000_0000h	23.3.3/1458
10_118C	DMA current link descriptor address register (DMA2_CLNDAR1)	32	R/W	0000_0000h	23.3.4/1459
10_1190	DMA source attributes register (DMA2_SATR1)	32	R/W	0000_0000h	23.3.5/1461
10_1194	DMA source address register (DMA2_SAR1)	32	R/W	0000_0000h	23.3.6/1462
10_1198	DMA destination attributes register (DMA2_DATR1)	32	R/W	0000_0000h	23.3.7/1463
10_119C	DMA destination address register (DMA2_DAR1)	32	R/W	0000_0000h	23.3.8/1464
10_11A0	DMA byte count register (DMA2_BCR1)	32	R/W	0000_0000h	23.3.9/1465
10_11A4	DMA extended next link descriptor address register (DMA2_ENLNDAR1)	32	R/W	0000_0000h	23.3.10/ 1466
10_11A8	DMA next link descriptor address register (DMA2_NLNDAR1)	32	R/W	0000_0000h	23.3.11/ 1466
10_11B0	DMA extended current list descriptor address register (DMA2_ECLSDAR1)	32	R/W	0000_0000h	23.3.12/ 1467
10_11B4	DMA current list descriptor address register (DMA2_CLSDAR1)	32	R/W	0000_0000h	23.3.13/ 1468
10_11B8	DMA extended next list descriptor address register (DMA2_ENLSDAR1)	32	R/W	0000_0000h	23.3.14/ 1469
10_11BC	DMA next list descriptor address register (DMA2_NLSDAR1)	32	R/W	0000_0000h	23.3.15/ 1469
10_11C0	DMA source stride register (DMA2_SSR1)	32	R/W	0000_0000h	23.3.16/ 1470
10_11C4	DMA destination stride register (DMA2_DSR1)	32	R/W	0000_0000h	23.3.17/ 1471
10_1200	DMA mode register (DMA2_MR2)	32	R/W	0800_0000h	23.3.1/1453
10_1204	DMA status register (DMA2_SR2)	32	R/W	0000_0000h	23.3.2/1457
10_1208	DMA current link descriptor extended address register (DMA2_ECLNDAR2)	32	R/W	0000_0000h	23.3.3/1458
10_120C	DMA current link descriptor address register (DMA2_CLNDAR2)	32	R/W	0000_0000h	23.3.4/1459
10_1210	DMA source attributes register (DMA2_SATR2)	32	R/W	0000_0000h	23.3.5/1461
10_1214	DMA source address register (DMA2_SAR2)	32	R/W	0000_0000h	23.3.6/1462
10_1218	DMA destination attributes register (DMA2_DATR2)	32	R/W	0000_0000h	23.3.7/1463

Table continues on the next page...

DMA memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
10_121C	DMA destination address register (DMA2_DAR2)	32	R/W	0000_0000h	23.3.8/1464
10_1220	DMA byte count register (DMA2_BCR2)	32	R/W	0000_0000h	23.3.9/1465
10_1224	DMA extended next link descriptor address register (DMA2_ENLNDAR2)	32	R/W	0000_0000h	23.3.10/ 1466
10_1228	DMA next link descriptor address register (DMA2_NLNDAR2)	32	R/W	0000_0000h	23.3.11/ 1466
10_1230	DMA extended current list descriptor address register (DMA2_ECLSDAR2)	32	R/W	0000_0000h	23.3.12/ 1467
10_1234	DMA current list descriptor address register (DMA2_CLSDAR2)	32	R/W	0000_0000h	23.3.13/ 1468
10_1238	DMA extended next list descriptor address register (DMA2_ENLSDAR2)	32	R/W	0000_0000h	23.3.14/ 1469
10_123C	DMA next list descriptor address register (DMA2_NLSDAR2)	32	R/W	0000_0000h	23.3.15/ 1469
10_1240	DMA source stride register (DMA2_SSR2)	32	R/W	0000_0000h	23.3.16/ 1470
10_1244	DMA destination stride register (DMA2_DSR2)	32	R/W	0000_0000h	23.3.17/ 1471
10_1280	DMA mode register (DMA2_MR3)	32	R/W	0800_0000h	23.3.1/1453
10_1284	DMA status register (DMA2_SR3)	32	R/W	0000_0000h	23.3.2/1457
10_1288	DMA current link descriptor extended address register (DMA2_ECLNDAR3)	32	R/W	0000_0000h	23.3.3/1458
10_128C	DMA current link descriptor address register (DMA2_CLNDAR3)	32	R/W	0000_0000h	23.3.4/1459
10_1290	DMA source attributes register (DMA2_SATR3)	32	R/W	0000_0000h	23.3.5/1461
10_1294	DMA source address register (DMA2_SAR3)	32	R/W	0000_0000h	23.3.6/1462
10_1298	DMA destination attributes register (DMA2_DATR3)	32	R/W	0000_0000h	23.3.7/1463
10_129C	DMA destination address register (DMA2_DAR3)	32	R/W	0000_0000h	23.3.8/1464
10_12A0	DMA byte count register (DMA2_BCR3)	32	R/W	0000_0000h	23.3.9/1465
10_12A4	DMA extended next link descriptor address register (DMA2_ENLNDAR3)	32	R/W	0000_0000h	23.3.10/ 1466
10_12A8	DMA next link descriptor address register (DMA2_NLNDAR3)	32	R/W	0000_0000h	23.3.11/ 1466
10_12B0	DMA extended current list descriptor address register (DMA2_ECLSDAR3)	32	R/W	0000_0000h	23.3.12/ 1467
10_12B4	DMA current list descriptor address register (DMA2_CLSDAR3)	32	R/W	0000_0000h	23.3.13/ 1468
10_12B8	DMA extended next list descriptor address register (DMA2_ENLSDAR3)	32	R/W	0000_0000h	23.3.14/ 1469
10_12BC	DMA next list descriptor address register (DMA2_NLSDAR3)	32	R/W	0000_0000h	23.3.15/ 1469
10_12C0	DMA source stride register (DMA2_SSR3)	32	R/W	0000_0000h	23.3.16/ 1470

Table continues on the next page...

DMA memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
10_12C4	DMA destination stride register (DMA2_DSR3)	32	R/W	0000_0000h	23.3.17/ 1471
10_1300	DMA general status register 0 (DMA2_DGSR0)	32	R	0000_0000h	23.3.18/ 1472
10_1400	DMA mode register (DMA2_MR4)	32	R/W	0800_0000h	23.3.19/ 1475
10_1404	DMA status register (DMA2_SR4)	32	R/W	0000_0000h	23.3.20/ 1479
10_1408	DMA extended current link descriptor address register (DMA2_ECLNDAR4)	32	R/W	0000_0000h	23.3.21/ 1480
10_140C	DMA current link descriptor address register (DMA2_CLNDAR4)	32	R/W	0000_0000h	23.3.22/ 1481
10_1410	DMA source attributes register (DMA2_SATR4)	32	R/W	0000_0000h	23.3.23/ 1483
10_1414	DMA source address register (DMA2_SAR4)	32	R/W	0000_0000h	23.3.24/ 1484
10_1418	DMA destination attributes register (DMA2_DATR4)	32	R/W	0000_0000h	23.3.25/ 1485
10_141C	DMA destination address register (DMA2_DAR4)	32	R/W	0000_0000h	23.3.26/ 1486
10_1420	DMA byte count register (DMA2_BCR4)	32	R/W	0000_0000h	23.3.27/ 1487
10_1424	DMA next link descriptor extended address register (DMA2_ENLNDAR4)	32	R/W	0000_0000h	23.3.28/ 1488
10_1428	DMA next link descriptor address register (DMA2_NLNDAR4)	32	R/W	0000_0000h	23.3.29/ 1488
10_1430	DMA extended current list descriptor address register (DMA2_ECLSDAR4)	32	R/W	0000_0000h	23.3.30/ 1489
10_1434	DMA current list descriptor address register (DMA2_CLSDAR4)	32	R/W	0000_0000h	23.3.31/ 1490
10_1438	DMA extended next list descriptor address register (DMA2_ENLSDAR4)	32	R/W	0000_0000h	23.3.32/ 1491
10_143C	DMA next list descriptor address register (DMA2_NLSDAR4)	32	R/W	0000_0000h	23.3.33/ 1491
10_1440	DMA source stride register (DMA2_SSR4)	32	R/W	0000_0000h	23.3.34/ 1492
10_1444	DMA destination stride register (DMA2_DSR4)	32	R/W	0000_0000h	23.3.35/ 1493
10_1480	DMA mode register (DMA2_MR5)	32	R/W	0800_0000h	23.3.19/ 1475
10_1484	DMA status register (DMA2_SR5)	32	R/W	0000_0000h	23.3.20/ 1479
10_1488	DMA extended current link descriptor address register (DMA2_ECLNDAR5)	32	R/W	0000_0000h	23.3.21/ 1480

Table continues on the next page...

DMA memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
10_148C	DMA current link descriptor address register (DMA2_CLNDAR5)	32	R/W	0000_0000h	23.3.22/ 1481
10_1490	DMA source attributes register (DMA2_SATR5)	32	R/W	0000_0000h	23.3.23/ 1483
10_1494	DMA source address register (DMA2_SAR5)	32	R/W	0000_0000h	23.3.24/ 1484
10_1498	DMA destination attributes register (DMA2_DATR5)	32	R/W	0000_0000h	23.3.25/ 1485
10_149C	DMA destination address register (DMA2_DAR5)	32	R/W	0000_0000h	23.3.26/ 1486
10_14A0	DMA byte count register (DMA2_BCR5)	32	R/W	0000_0000h	23.3.27/ 1487
10_14A4	DMA next link descriptor extended address register (DMA2_ENLNDAR5)	32	R/W	0000_0000h	23.3.28/ 1488
10_14A8	DMA next link descriptor address register (DMA2_NLNDAR5)	32	R/W	0000_0000h	23.3.29/ 1488
10_14B0	DMA extended current list descriptor address register (DMA2_ECLSDAR5)	32	R/W	0000_0000h	23.3.30/ 1489
10_14B4	DMA current list descriptor address register (DMA2_CLSDAR5)	32	R/W	0000_0000h	23.3.31/ 1490
10_14B8	DMA extended next list descriptor address register (DMA2_ENLSDAR5)	32	R/W	0000_0000h	23.3.32/ 1491
10_14BC	DMA next list descriptor address register (DMA2_NLSDAR5)	32	R/W	0000_0000h	23.3.33/ 1491
10_14C0	DMA source stride register (DMA2_SSR5)	32	R/W	0000_0000h	23.3.34/ 1492
10_14C4	DMA destination stride register (DMA2_DSR5)	32	R/W	0000_0000h	23.3.35/ 1493
10_1500	DMA mode register (DMA2_MR6)	32	R/W	0800_0000h	23.3.19/ 1475
10_1504	DMA status register (DMA2_SR6)	32	R/W	0000_0000h	23.3.20/ 1479
10_1508	DMA extended current link descriptor address register (DMA2_ECLNDAR6)	32	R/W	0000_0000h	23.3.21/ 1480
10_150C	DMA current link descriptor address register (DMA2_CLNDAR6)	32	R/W	0000_0000h	23.3.22/ 1481
10_1510	DMA source attributes register (DMA2_SATR6)	32	R/W	0000_0000h	23.3.23/ 1483
10_1514	DMA source address register (DMA2_SAR6)	32	R/W	0000_0000h	23.3.24/ 1484
10_1518	DMA destination attributes register (DMA2_DATR6)	32	R/W	0000_0000h	23.3.25/ 1485
10_151C	DMA destination address register (DMA2_DAR6)	32	R/W	0000_0000h	23.3.26/ 1486

Table continues on the next page...

DMA memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
10_1520	DMA byte count register (DMA2_BCR6)	32	R/W	0000_0000h	23.3.27/ 1487
10_1524	DMA next link descriptor extended address register (DMA2_ENLNDAR6)	32	R/W	0000_0000h	23.3.28/ 1488
10_1528	DMA next link descriptor address register (DMA2_NLNDAR6)	32	R/W	0000_0000h	23.3.29/ 1488
10_1530	DMA extended current list descriptor address register (DMA2_ECLSDAR6)	32	R/W	0000_0000h	23.3.30/ 1489
10_1534	DMA current list descriptor address register (DMA2_CLSDAR6)	32	R/W	0000_0000h	23.3.31/ 1490
10_1538	DMA extended next list descriptor address register (DMA2_ENLSDAR6)	32	R/W	0000_0000h	23.3.32/ 1491
10_153C	DMA next list descriptor address register (DMA2_NLSDAR6)	32	R/W	0000_0000h	23.3.33/ 1491
10_1540	DMA source stride register (DMA2_SSR6)	32	R/W	0000_0000h	23.3.34/ 1492
10_1544	DMA destination stride register (DMA2_DSR6)	32	R/W	0000_0000h	23.3.35/ 1493
10_1580	DMA mode register (DMA2_MR7)	32	R/W	0800_0000h	23.3.19/ 1475
10_1584	DMA status register (DMA2_SR7)	32	R/W	0000_0000h	23.3.20/ 1479
10_1588	DMA extended current link descriptor address register (DMA2_ECLNDAR7)	32	R/W	0000_0000h	23.3.21/ 1480
10_158C	DMA current link descriptor address register (DMA2_CLNDAR7)	32	R/W	0000_0000h	23.3.22/ 1481
10_1590	DMA source attributes register (DMA2_SATR7)	32	R/W	0000_0000h	23.3.23/ 1483
10_1594	DMA source address register (DMA2_SAR7)	32	R/W	0000_0000h	23.3.24/ 1484
10_1598	DMA destination attributes register (DMA2_DATR7)	32	R/W	0000_0000h	23.3.25/ 1485
10_159C	DMA destination address register (DMA2_DAR7)	32	R/W	0000_0000h	23.3.26/ 1486
10_15A0	DMA byte count register (DMA2_BCR7)	32	R/W	0000_0000h	23.3.27/ 1487
10_15A4	DMA next link descriptor extended address register (DMA2_ENLNDAR7)	32	R/W	0000_0000h	23.3.28/ 1488
10_15A8	DMA next link descriptor address register (DMA2_NLNDAR7)	32	R/W	0000_0000h	23.3.29/ 1488
10_15B0	DMA extended current list descriptor address register (DMA2_ECLSDAR7)	32	R/W	0000_0000h	23.3.30/ 1489
10_15B4	DMA current list descriptor address register (DMA2_CLSDAR7)	32	R/W	0000_0000h	23.3.31/ 1490

Table continues on the next page...

DMA memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
10_15B8	DMA extended next list descriptor address register (DMA2_ENLSDAR7)	32	R/W	0000_0000h	23.3.32/ 1491
10_15BC	DMA next list descriptor address register (DMA2_NLSDAR7)	32	R/W	0000_0000h	23.3.33/ 1491
10_15C0	DMA source stride register (DMA2_SSR7)	32	R/W	0000_0000h	23.3.34/ 1492
10_15C4	DMA destination stride register (DMA2_DSR7)	32	R/W	0000_0000h	23.3.35/ 1493
10_1600	DMA general status register 1 (DMA2_DGSR1)	32	R	0000_0000h	23.3.36/ 1494
10_2100	DMA mode register (DMA3_MR0)	32	R/W	0800_0000h	23.3.1/1453
10_2104	DMA status register (DMA3_SR0)	32	R/W	0000_0000h	23.3.2/1457
10_2108	DMA current link descriptor extended address register (DMA3_ECLNDAR0)	32	R/W	0000_0000h	23.3.3/1458
10_210C	DMA current link descriptor address register (DMA3_CLNDAR0)	32	R/W	0000_0000h	23.3.4/1459
10_2110	DMA source attributes register (DMA3_SATR0)	32	R/W	0000_0000h	23.3.5/1461
10_2114	DMA source address register (DMA3_SAR0)	32	R/W	0000_0000h	23.3.6/1462
10_2118	DMA destination attributes register (DMA3_DATR0)	32	R/W	0000_0000h	23.3.7/1463
10_211C	DMA destination address register (DMA3_DAR0)	32	R/W	0000_0000h	23.3.8/1464
10_2120	DMA byte count register (DMA3_BCR0)	32	R/W	0000_0000h	23.3.9/1465
10_2124	DMA extended next link descriptor address register (DMA3_ENLNDAR0)	32	R/W	0000_0000h	23.3.10/ 1466
10_2128	DMA next link descriptor address register (DMA3_NLNDAR0)	32	R/W	0000_0000h	23.3.11/ 1466
10_2130	DMA extended current list descriptor address register (DMA3_ECLSDAR0)	32	R/W	0000_0000h	23.3.12/ 1467
10_2134	DMA current list descriptor address register (DMA3_CLSDAR0)	32	R/W	0000_0000h	23.3.13/ 1468
10_2138	DMA extended next list descriptor address register (DMA3_ENLSDAR0)	32	R/W	0000_0000h	23.3.14/ 1469
10_213C	DMA next list descriptor address register (DMA3_NLSDAR0)	32	R/W	0000_0000h	23.3.15/ 1469
10_2140	DMA source stride register (DMA3_SSR0)	32	R/W	0000_0000h	23.3.16/ 1470
10_2144	DMA destination stride register (DMA3_DSR0)	32	R/W	0000_0000h	23.3.17/ 1471
10_2180	DMA mode register (DMA3_MR1)	32	R/W	0800_0000h	23.3.1/1453
10_2184	DMA status register (DMA3_SR1)	32	R/W	0000_0000h	23.3.2/1457
10_2188	DMA current link descriptor extended address register (DMA3_ECLNDAR1)	32	R/W	0000_0000h	23.3.3/1458
10_218C	DMA current link descriptor address register (DMA3_CLNDAR1)	32	R/W	0000_0000h	23.3.4/1459

Table continues on the next page...

DMA memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
10_2190	DMA source attributes register (DMA3_SATR1)	32	R/W	0000_0000h	23.3.5/1461
10_2194	DMA source address register (DMA3_SAR1)	32	R/W	0000_0000h	23.3.6/1462
10_2198	DMA destination attributes register (DMA3_DATR1)	32	R/W	0000_0000h	23.3.7/1463
10_219C	DMA destination address register (DMA3_DAR1)	32	R/W	0000_0000h	23.3.8/1464
10_21A0	DMA byte count register (DMA3_BCR1)	32	R/W	0000_0000h	23.3.9/1465
10_21A4	DMA extended next link descriptor address register (DMA3_ENLNDAR1)	32	R/W	0000_0000h	23.3.10/ 1466
10_21A8	DMA next link descriptor address register (DMA3_NLNDAR1)	32	R/W	0000_0000h	23.3.11/ 1466
10_21B0	DMA extended current list descriptor address register (DMA3_ECLSDAR1)	32	R/W	0000_0000h	23.3.12/ 1467
10_21B4	DMA current list descriptor address register (DMA3_CLSDAR1)	32	R/W	0000_0000h	23.3.13/ 1468
10_21B8	DMA extended next list descriptor address register (DMA3_ENLSDAR1)	32	R/W	0000_0000h	23.3.14/ 1469
10_21BC	DMA next list descriptor address register (DMA3_NLSDAR1)	32	R/W	0000_0000h	23.3.15/ 1469
10_21C0	DMA source stride register (DMA3_SSR1)	32	R/W	0000_0000h	23.3.16/ 1470
10_21C4	DMA destination stride register (DMA3_DSR1)	32	R/W	0000_0000h	23.3.17/ 1471
10_2200	DMA mode register (DMA3_MR2)	32	R/W	0800_0000h	23.3.1/1453
10_2204	DMA status register (DMA3_SR2)	32	R/W	0000_0000h	23.3.2/1457
10_2208	DMA current link descriptor extended address register (DMA3_ECLNDAR2)	32	R/W	0000_0000h	23.3.3/1458
10_220C	DMA current link descriptor address register (DMA3_CLNDAR2)	32	R/W	0000_0000h	23.3.4/1459
10_2210	DMA source attributes register (DMA3_SATR2)	32	R/W	0000_0000h	23.3.5/1461
10_2214	DMA source address register (DMA3_SAR2)	32	R/W	0000_0000h	23.3.6/1462
10_2218	DMA destination attributes register (DMA3_DATR2)	32	R/W	0000_0000h	23.3.7/1463
10_221C	DMA destination address register (DMA3_DAR2)	32	R/W	0000_0000h	23.3.8/1464
10_2220	DMA byte count register (DMA3_BCR2)	32	R/W	0000_0000h	23.3.9/1465
10_2224	DMA extended next link descriptor address register (DMA3_ENLNDAR2)	32	R/W	0000_0000h	23.3.10/ 1466
10_2228	DMA next link descriptor address register (DMA3_NLNDAR2)	32	R/W	0000_0000h	23.3.11/ 1466
10_2230	DMA extended current list descriptor address register (DMA3_ECLSDAR2)	32	R/W	0000_0000h	23.3.12/ 1467
10_2234	DMA current list descriptor address register (DMA3_CLSDAR2)	32	R/W	0000_0000h	23.3.13/ 1468
10_2238	DMA extended next list descriptor address register (DMA3_ENLSDAR2)	32	R/W	0000_0000h	23.3.14/ 1469

Table continues on the next page...

DMA memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
10_223C	DMA next list descriptor address register (DMA3_NLSDAR2)	32	R/W	0000_0000h	23.3.15/ 1469
10_2240	DMA source stride register (DMA3_SSR2)	32	R/W	0000_0000h	23.3.16/ 1470
10_2244	DMA destination stride register (DMA3_DSR2)	32	R/W	0000_0000h	23.3.17/ 1471
10_2280	DMA mode register (DMA3_MR3)	32	R/W	0800_0000h	23.3.1/1453
10_2284	DMA status register (DMA3_SR3)	32	R/W	0000_0000h	23.3.2/1457
10_2288	DMA current link descriptor extended address register (DMA3_ECLNDAR3)	32	R/W	0000_0000h	23.3.3/1458
10_228C	DMA current link descriptor address register (DMA3_CLNDAR3)	32	R/W	0000_0000h	23.3.4/1459
10_2290	DMA source attributes register (DMA3_SATR3)	32	R/W	0000_0000h	23.3.5/1461
10_2294	DMA source address register (DMA3_SAR3)	32	R/W	0000_0000h	23.3.6/1462
10_2298	DMA destination attributes register (DMA3_DATR3)	32	R/W	0000_0000h	23.3.7/1463
10_229C	DMA destination address register (DMA3_DAR3)	32	R/W	0000_0000h	23.3.8/1464
10_22A0	DMA byte count register (DMA3_BCR3)	32	R/W	0000_0000h	23.3.9/1465
10_22A4	DMA extended next link descriptor address register (DMA3_ENLNDAR3)	32	R/W	0000_0000h	23.3.10/ 1466
10_22A8	DMA next link descriptor address register (DMA3_NLNDAR3)	32	R/W	0000_0000h	23.3.11/ 1466
10_22B0	DMA extended current list descriptor address register (DMA3_ECLSDAR3)	32	R/W	0000_0000h	23.3.12/ 1467
10_22B4	DMA current list descriptor address register (DMA3_CLSDAR3)	32	R/W	0000_0000h	23.3.13/ 1468
10_22B8	DMA extended next list descriptor address register (DMA3_ENLSDAR3)	32	R/W	0000_0000h	23.3.14/ 1469
10_22BC	DMA next list descriptor address register (DMA3_NLSDAR3)	32	R/W	0000_0000h	23.3.15/ 1469
10_22C0	DMA source stride register (DMA3_SSR3)	32	R/W	0000_0000h	23.3.16/ 1470
10_22C4	DMA destination stride register (DMA3_DSR3)	32	R/W	0000_0000h	23.3.17/ 1471
10_2300	DMA general status register 0 (DMA3_DGSR0)	32	R	0000_0000h	23.3.18/ 1472
10_2400	DMA mode register (DMA3_MR4)	32	R/W	0800_0000h	23.3.19/ 1475
10_2404	DMA status register (DMA3_SR4)	32	R/W	0000_0000h	23.3.20/ 1479
10_2408	DMA extended current link descriptor address register (DMA3_ECLNDAR4)	32	R/W	0000_0000h	23.3.21/ 1480
10_240C	DMA current link descriptor address register (DMA3_CLNDAR4)	32	R/W	0000_0000h	23.3.22/ 1481

Table continues on the next page...

DMA memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
10_2410	DMA source attributes register (DMA3_SATR4)	32	R/W	0000_0000h	23.3.23/ 1483
10_2414	DMA source address register (DMA3_SAR4)	32	R/W	0000_0000h	23.3.24/ 1484
10_2418	DMA destination attributes register (DMA3_DATR4)	32	R/W	0000_0000h	23.3.25/ 1485
10_241C	DMA destination address register (DMA3_DAR4)	32	R/W	0000_0000h	23.3.26/ 1486
10_2420	DMA byte count register (DMA3_BCR4)	32	R/W	0000_0000h	23.3.27/ 1487
10_2424	DMA next link descriptor extended address register (DMA3_ENLNDAR4)	32	R/W	0000_0000h	23.3.28/ 1488
10_2428	DMA next link descriptor address register (DMA3_NLNDAR4)	32	R/W	0000_0000h	23.3.29/ 1488
10_2430	DMA extended current list descriptor address register (DMA3_ECLSDAR4)	32	R/W	0000_0000h	23.3.30/ 1489
10_2434	DMA current list descriptor address register (DMA3_CLSDAR4)	32	R/W	0000_0000h	23.3.31/ 1490
10_2438	DMA extended next list descriptor address register (DMA3_ENLSDAR4)	32	R/W	0000_0000h	23.3.32/ 1491
10_243C	DMA next list descriptor address register (DMA3_NLSDAR4)	32	R/W	0000_0000h	23.3.33/ 1491
10_2440	DMA source stride register (DMA3_SSR4)	32	R/W	0000_0000h	23.3.34/ 1492
10_2444	DMA destination stride register (DMA3_DSR4)	32	R/W	0000_0000h	23.3.35/ 1493
10_2480	DMA mode register (DMA3_MR5)	32	R/W	0800_0000h	23.3.19/ 1475
10_2484	DMA status register (DMA3_SR5)	32	R/W	0000_0000h	23.3.20/ 1479
10_2488	DMA extended current link descriptor address register (DMA3_ECLNDAR5)	32	R/W	0000_0000h	23.3.21/ 1480
10_248C	DMA current link descriptor address register (DMA3_CLNDAR5)	32	R/W	0000_0000h	23.3.22/ 1481
10_2490	DMA source attributes register (DMA3_SATR5)	32	R/W	0000_0000h	23.3.23/ 1483
10_2494	DMA source address register (DMA3_SAR5)	32	R/W	0000_0000h	23.3.24/ 1484
10_2498	DMA destination attributes register (DMA3_DATR5)	32	R/W	0000_0000h	23.3.25/ 1485
10_249C	DMA destination address register (DMA3_DAR5)	32	R/W	0000_0000h	23.3.26/ 1486
10_24A0	DMA byte count register (DMA3_BCR5)	32	R/W	0000_0000h	23.3.27/ 1487

Table continues on the next page...

DMA memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
10_24A4	DMA next link descriptor extended address register (DMA3_ENLNDAR5)	32	R/W	0000_0000h	23.3.28/ 1488
10_24A8	DMA next link descriptor address register (DMA3_NLNDAR5)	32	R/W	0000_0000h	23.3.29/ 1488
10_24B0	DMA extended current list descriptor address register (DMA3_ECLSDAR5)	32	R/W	0000_0000h	23.3.30/ 1489
10_24B4	DMA current list descriptor address register (DMA3_CLSDAR5)	32	R/W	0000_0000h	23.3.31/ 1490
10_24B8	DMA extended next list descriptor address register (DMA3_ENLSDAR5)	32	R/W	0000_0000h	23.3.32/ 1491
10_24BC	DMA next list descriptor address register (DMA3_NLSDAR5)	32	R/W	0000_0000h	23.3.33/ 1491
10_24C0	DMA source stride register (DMA3_SSR5)	32	R/W	0000_0000h	23.3.34/ 1492
10_24C4	DMA destination stride register (DMA3_DSR5)	32	R/W	0000_0000h	23.3.35/ 1493
10_2500	DMA mode register (DMA3_MR6)	32	R/W	0800_0000h	23.3.19/ 1475
10_2504	DMA status register (DMA3_SR6)	32	R/W	0000_0000h	23.3.20/ 1479
10_2508	DMA extended current link descriptor address register (DMA3_ECLNDAR6)	32	R/W	0000_0000h	23.3.21/ 1480
10_250C	DMA current link descriptor address register (DMA3_CLNDAR6)	32	R/W	0000_0000h	23.3.22/ 1481
10_2510	DMA source attributes register (DMA3_SATR6)	32	R/W	0000_0000h	23.3.23/ 1483
10_2514	DMA source address register (DMA3_SAR6)	32	R/W	0000_0000h	23.3.24/ 1484
10_2518	DMA destination attributes register (DMA3_DATR6)	32	R/W	0000_0000h	23.3.25/ 1485
10_251C	DMA destination address register (DMA3_DAR6)	32	R/W	0000_0000h	23.3.26/ 1486
10_2520	DMA byte count register (DMA3_BCR6)	32	R/W	0000_0000h	23.3.27/ 1487
10_2524	DMA next link descriptor extended address register (DMA3_ENLNDAR6)	32	R/W	0000_0000h	23.3.28/ 1488
10_2528	DMA next link descriptor address register (DMA3_NLNDAR6)	32	R/W	0000_0000h	23.3.29/ 1488
10_2530	DMA extended current list descriptor address register (DMA3_ECLSDAR6)	32	R/W	0000_0000h	23.3.30/ 1489
10_2534	DMA current list descriptor address register (DMA3_CLSDAR6)	32	R/W	0000_0000h	23.3.31/ 1490
10_2538	DMA extended next list descriptor address register (DMA3_ENLSDAR6)	32	R/W	0000_0000h	23.3.32/ 1491

Table continues on the next page...

DMA memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
10_253C	DMA next list descriptor address register (DMA3_NLSDAR6)	32	R/W	0000_0000h	23.3.33/ 1491
10_2540	DMA source stride register (DMA3_SSR6)	32	R/W	0000_0000h	23.3.34/ 1492
10_2544	DMA destination stride register (DMA3_DSR6)	32	R/W	0000_0000h	23.3.35/ 1493
10_2580	DMA mode register (DMA3_MR7)	32	R/W	0800_0000h	23.3.19/ 1475
10_2584	DMA status register (DMA3_SR7)	32	R/W	0000_0000h	23.3.20/ 1479
10_2588	DMA extended current link descriptor address register (DMA3_ECLNDAR7)	32	R/W	0000_0000h	23.3.21/ 1480
10_258C	DMA current link descriptor address register (DMA3_CLNDAR7)	32	R/W	0000_0000h	23.3.22/ 1481
10_2590	DMA source attributes register (DMA3_SATR7)	32	R/W	0000_0000h	23.3.23/ 1483
10_2594	DMA source address register (DMA3_SAR7)	32	R/W	0000_0000h	23.3.24/ 1484
10_2598	DMA destination attributes register (DMA3_DATR7)	32	R/W	0000_0000h	23.3.25/ 1485
10_259C	DMA destination address register (DMA3_DAR7)	32	R/W	0000_0000h	23.3.26/ 1486
10_25A0	DMA byte count register (DMA3_BCR7)	32	R/W	0000_0000h	23.3.27/ 1487
10_25A4	DMA next link descriptor extended address register (DMA3_ENLNDAR7)	32	R/W	0000_0000h	23.3.28/ 1488
10_25A8	DMA next link descriptor address register (DMA3_NLNDAR7)	32	R/W	0000_0000h	23.3.29/ 1488
10_25B0	DMA extended current list descriptor address register (DMA3_ECLSDAR7)	32	R/W	0000_0000h	23.3.30/ 1489
10_25B4	DMA current list descriptor address register (DMA3_CLSDAR7)	32	R/W	0000_0000h	23.3.31/ 1490
10_25B8	DMA extended next list descriptor address register (DMA3_ENLSDAR7)	32	R/W	0000_0000h	23.3.32/ 1491
10_25BC	DMA next list descriptor address register (DMA3_NLSDAR7)	32	R/W	0000_0000h	23.3.33/ 1491
10_25C0	DMA source stride register (DMA3_SSR7)	32	R/W	0000_0000h	23.3.34/ 1492
10_25C4	DMA destination stride register (DMA3_DSR7)	32	R/W	0000_0000h	23.3.35/ 1493
10_2600	DMA general status register 1 (DMA3_DGSR1)	32	R	0000_0000h	23.3.36/ 1494

23.3.1 DMA mode register (DMAx_MR*n*)

The mode register allows software to start a DMA transfer and to control various DMA transfer characteristics.

Address: Base address + 100h offset + (128d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved				BWC				Reserved		EMP_EN	Reserved		EMS_EN	DAHTS	
W																
Reset	0	0	0	0	1	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	SAHTS		DAHE	SAHE	Reserved	SRW	EOSIE	EOLNIE	EOLSIE	EIE	XFE	CDSM_SWSM	CA	CTM	CC	CS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMAx_MR*n* field descriptions

Field	Description
0–3 -	This field is reserved. Reserved
4–7 BWC	Bandwidth/pause control. Defined when single and multiple channels are executing or if MRn[EMP_EN] is set in external transfer mode. The value of MRn[BWC] determines how many bytes a given channel is allowed to transfer before the DMA engine pauses the current channel and re-arbitrates (switches to the next channel). In external pause mode, the value of MRn[BWC] dictates how many bytes are allowed to transfer before pausing the channel, after which a new assertion of DREQ_B resumes channel operation. 0000 1 byte 0001 2 bytes 0010 4 bytes 0011 8 bytes 0100 16 bytes

Table continues on the next page...

DMAx_MR*n* field descriptions (continued)

Field	Description
	0101 32 bytes 0110 64 bytes 0111 128 bytes 1000 256 bytes 1001 512 bytes 1010 1024 bytes 1011-1110 Reserved 1111 Disable bandwidth sharing to allow uninterrupted transfers from each channel.
8–9 - Reserved	This field is reserved.
10 EMP_EN	External master pause enable. Valid only if MRn[EMS_EN] is set. 0 Disable the external master pause feature. 1 Enable the external master pause feature. Channel is paused as described by MRn[BWC].
11–12 - Reserved	This field is reserved.
13 EMS_EN	External master start enable. This bit does not apply to single-write start modes (direct or chaining). 0 Disable the channel from being started by an external DMA start pin. 1 Enable the channel to be started by an external DMA start pin, which sets MRn[CS].
14–15 DAHTS	Destination address hold transfer size. Indicates the transfer size used for each transaction while MRn[DAHE] is set. The byte count register must be in multiples of the size and the destination address register must be aligned based on the size. The transfer size assigned to MRn[DAHTS] must be equal to or smaller than that assigned to MRn[BWC] to avoid undefined behavior. 00 1 byte 01 2 bytes 10 4 bytes 11 8 bytes
16–17 SAHTS	Source address hold transfer size. Indicates the transfer size used for each transaction while MRn[SAHE] is set. The byte count register must be in multiples of the size and the source address register must be aligned based on the size. The transfer size assigned to MRn[SAHTS] must be equal to or smaller than that assigned to MRn[BWC] to avoid undefined behavior. 00 1 byte 01 2 bytes 10 4 bytes 11 8 bytes
18 DAHE	Destination address hold enable 0 Disable destination address hold 1 Enable the DMA controller to hold the destination address of a transfer to the size specified by MRn[DAHTS]. Hardware only supports aligned transfers for this feature.
19 SAHE	Source address hold enable 0 Disable source address hold 1 Enable the DMA controller to hold the source address of a transfer to the size specified by MRn[SAHTS]. Hardware only supports aligned transfers for this feature.

Table continues on the next page...

DMAx_MR*n* field descriptions (continued)

Field	Description
20 -	This field is reserved. Reserved
21 SRW	Single register write (Direct mode only; reserved for chaining mode.) 0 Normal operation 1 Enable a write to the source address register to simultaneously set MRn[CS], starting a DMA transfer, when MRn[CDSM_SWSM] is also set. Setting this bit and clearing CDSM_SWSM causes a write to the destination address register to simultaneously set MRn[CS], starting a DMA transfer.
22 EOSIE	End-of-segments interrupt enable 0 Do not generate an interrupt at the completion of a data transfer. CLNDARn[EOSIE] overrides this bit on a link descriptor basis. 1 Generate an interrupt at the completion of a data transfer (That is, SRn[EOSI] is set). This bit overrides the CLNDARn[EOSIE].
23 EOLNIE	End-of-links interrupt enable 0 Do not generate an interrupt at the completion of a list of DMA transfers. 1 Generate an interrupt at the completion of a list of DMA transfers (That is, NLNDARn[EOLND] is set).
24 EOLSIE	End-of-lists interrupt enable 0 Do not generate an interrupt at the completion of all DMA transfers. 1 Generate an interrupt at the completion of all DMA transfers (That is, NLNDARn[EOLND] and NLSDARn[EOLSD] are set).
25 EIE	Error interrupt enable 0 Do not generate an interrupt if a programming or transfer error is detected. 1 Generate an interrupt if a programming or transfer error is detected.
26 XFE	Extended features enable 0 Disable striding feature in direct mode or disable list chaining feature in chaining mode. 1 Enable striding feature in direct mode or enable list chaining feature in chaining mode.
27 CDSM_SWSM	In chaining mode, current descriptor start mode/single-write start mode is as follows: • In basic mode (MRn[XFE] is cleared), setting this bit causes a write to the current link descriptor address register to simultaneously set MRn[CS], starting a DMA transfer. • In extended chaining mode (MRn[XFE] is set), setting this bit causes a write to the current list descriptor address register to simultaneously set MRn[CS], starting a DMA transfer. In direct mode, setting this bit and MRn[SRW] causes a write to the source address register to simultaneously set MRn[CS], starting a DMA transfer. Clearing this bit and setting MRn[SRW] causes a write to the destination address register to simultaneously set MRn[CS], starting a DMA transfer. This bit must be cleared when MRn[SRW] is cleared.
28 CA	Channel abort 0 No effect 1 Cause the current transfer to be aborted and SRn[CB] to be cleared if the channel is busy. The channel remains in the idle state until a new transfer is programmed.
29 CTM	Channel transfer mode 0 Configure the channel in chaining mode. 1 Configure the channel into direct mode. This means that software is responsible for placing all the required parameters into necessary registers to start the DMA process.

Table continues on the next page...

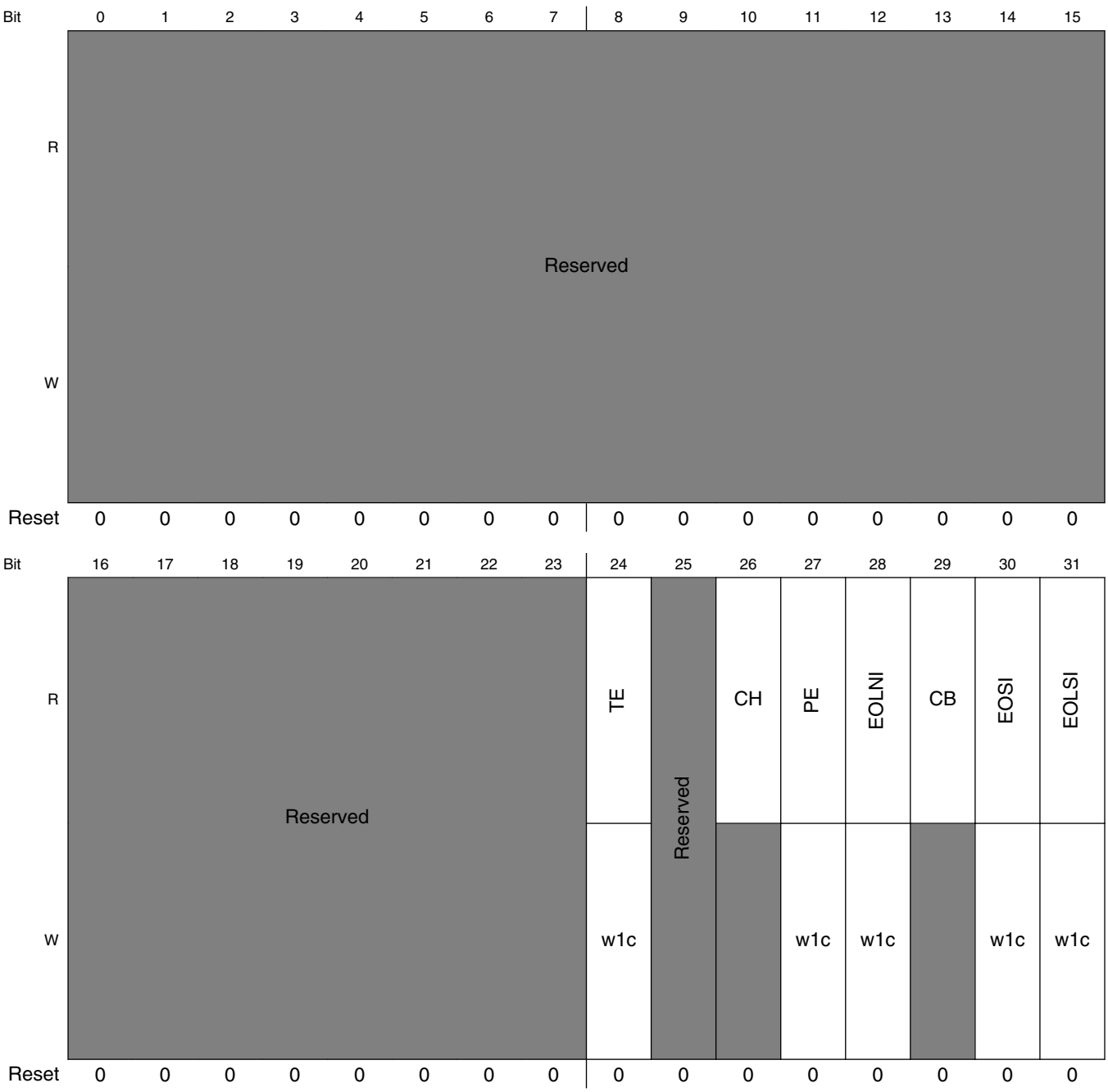
DMAx_MR*n* field descriptions (continued)

Field	Description
30 CC	Channel continue. This bit applies only to chaining mode and is cleared by hardware after the first descriptor read when continuing a transfer. This bit is reserved for external master mode. 0 No effect 1 The DMA transfer restarts the transferring process starting at the current descriptor address.
31 CS	Channel start. This bit is also automatically set by hardware during single-write start mode and external master start enable mode. Note that in external control mode, deasserting DMA_DREQ_B does NOT clear this bit. 0 Halt the DMA process if channel is busy (SRn[CB] is set). No effect if the channel is not busy. 1 Start the DMA process if channel is not busy (CB is cleared). If the channel was halted (CS = 0 and CB = 1), the transfer continues from the point at which it was halted.

23.3.2 DMA status register (DMA*x*_SR*n*)

The status registers report various DMA conditions during and after a DMA transfer.

Address: Base address + 104h offset + (128d × i), where i=0d to 3d



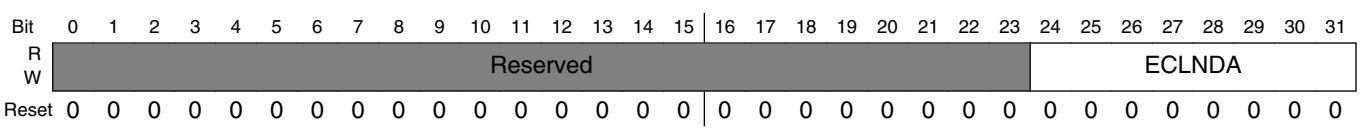
DMAx_SR*n* field descriptions

Field	Description
0–23 -	This field is reserved. Reserved
24 TE	Transfer error (Bit reset, write 1 to clear) 0 No error condition during the DMA transfer 1 Error condition during the DMA transfer. See DMA errors , for additional information.
25 -	This field is reserved. Reserved
26 CH	Channel halted. Cleared automatically by hardware if MR *n* [CS] is set again for resuming a halted transfer 0 Channel is not halted. If software attempts to halt an idle channel (SRn[CB] is cleared), this bit remains 0. 1 DMA transfer was successfully halted by software and can be resumed.
27 PE	Programming error (bit reset, write 1 to clear) 0 No programming error detected 1 A programming error is detected that prevents the DMA transfer from occurring.
28 EOLNI	End-of-links interrupt. After transferring the last block of data in the last link descriptor, if MRn[EOLNIE] is set, then this bit is set and an interrupt is generated. (Bit reset, write 1 to clear)
29 CB	Channel busy 0 DMA transfer is finished, an error occurred, or a channel abort occurred. 1 DMA transfer is currently in progress.
30 EOSI	End-of-segment interrupt. In chaining mode, after finishing a data transfer, if MRn[EOSIE] is set or if CLNDARn[EOSIE] is set, this bit gets set and an interrupt is generated. In direct mode, if MRn[EOSIE] is set, this bit gets set and an interrupt is generated. (Bit reset, write 1 to clear)
31 EOLSI	End-of-list interrupt. After transferring the last block of data in the last list descriptor, if MRn[EOLSIE] is set, then this bit is set and an interrupt is generated. (Bit reset, write 1 to clear)

23.3.3 DMA current link descriptor extended address register (DMAx_ECLNDAR*n*)

These registers contain the upper 8 bits of the address of the current link descriptor. See DMA current link descriptor address register (DMA_CLNDAR*n*)DMA current link descriptor address register (DMA_CLNDAR*n*) for a description of basic chaining mode.

Address: Base address + 108h offset + (128d × i), where i=0d to 3d



DMAx_ECLNDAR*n* field descriptions

Field	Description
0–23 -	This field is reserved. Reserved
24–31 ECLNDA	Current link descriptor extended address (upper 8 bits of 40-bit address)

23.3.4 DMA current link descriptor address register (DMAx_CLNDAR*n*)

Current link descriptor address registers contain the address of the current link descriptor.

In basic chaining mode, shown in , software must initialize these registers to point to the first link descriptors in memory.

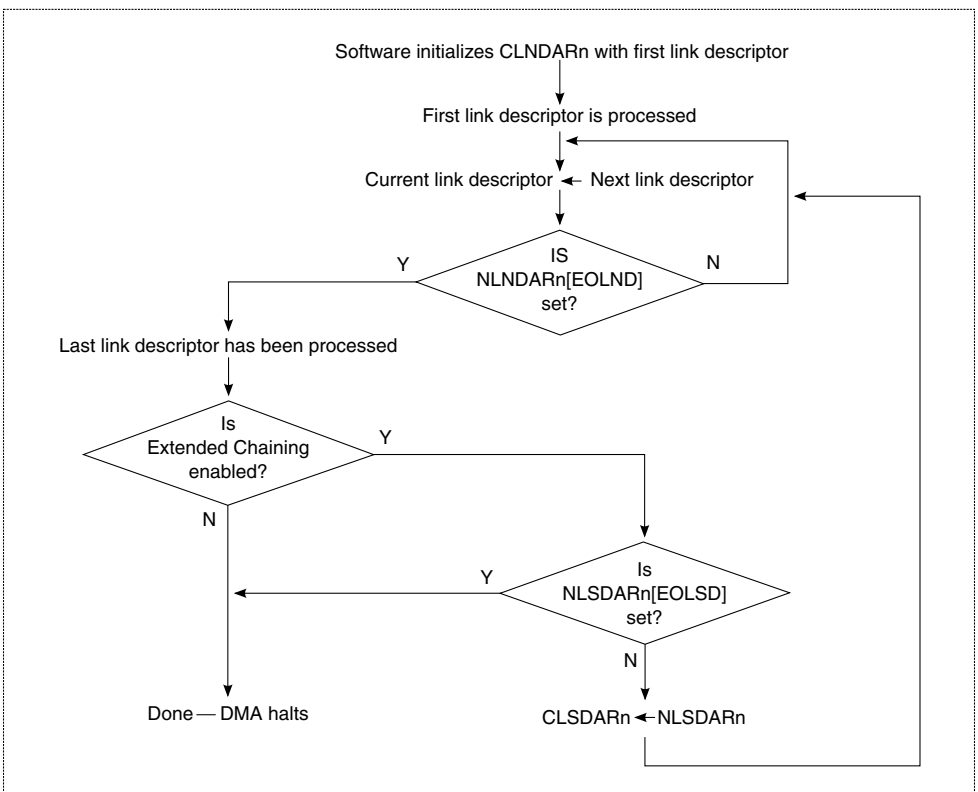
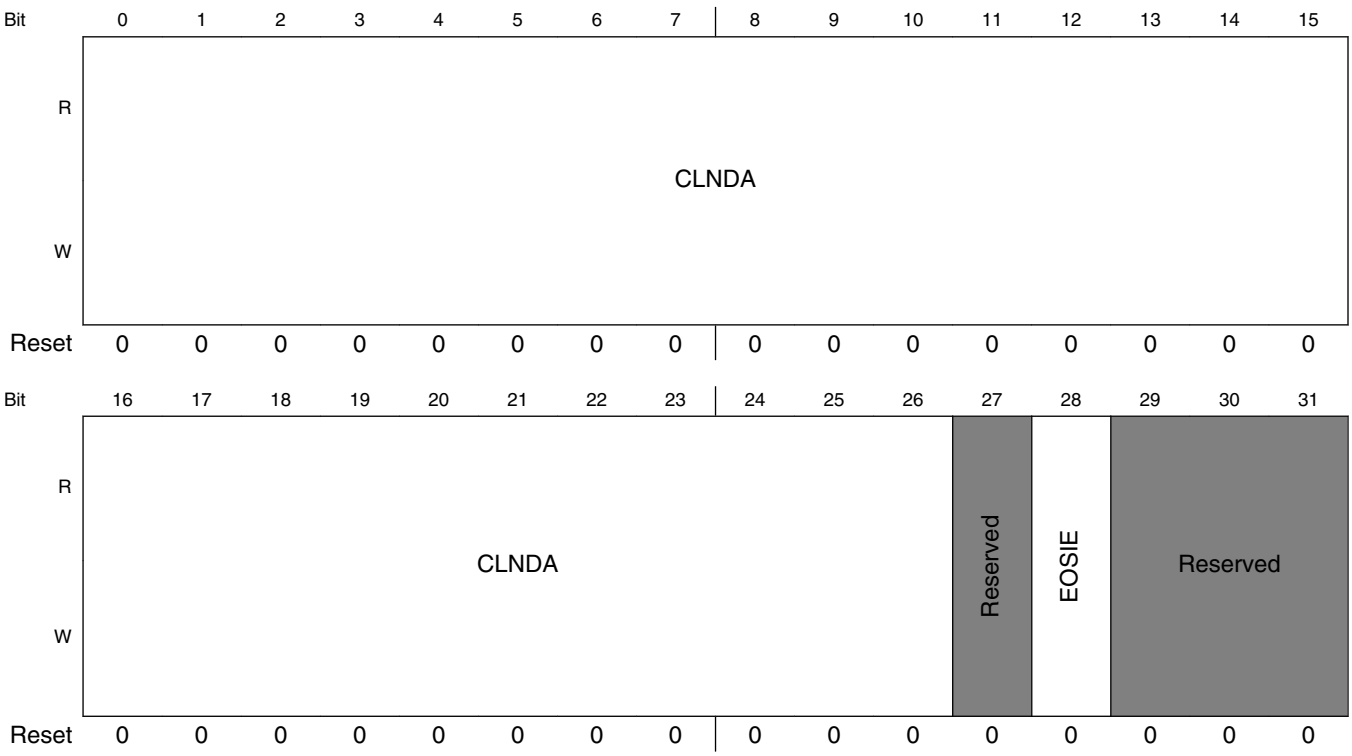


Figure 23-195. Basic Chaining Mode Flow Chart

After the current descriptor is processed, the current link descriptor address register is loaded from the next link descriptor address register and NLNDAR *n* [EOLND] in the next link descriptor address register is examined. If EOLND is zero, the DMA controller reads in the new current link descriptor for processing. If EOLND is set, the last descriptor of the list was just completed. If extended chaining mode is not enabled, all DMA transfers are complete and the DMA controller halts.

If extended chaining mode is enabled, the DMA controller examines the state of NLSDAR *n* [EOLSD] in the next list descriptor address register. If EOLSD is clear, the controller loads the contents of the next list descriptor address register into the current list descriptor address register and reads the new list descriptor from memory. If EOLSD is set, all DMA transfers are complete and the DMA controller halts.

Address: Base address + 10Ch offset + (128d × i), where i=0d to 3d



DMA*x*_CLNDAR*n* field descriptions

Field	Description
0–26 CLNDA	Current link descriptor address. Contains the current descriptor address of the buffer descriptor in memory. The descriptor must be aligned to a 32-byte boundary. (This is the lower portion of the 40 -bit address formed by CLNDARn[CLNDA] and ECLNDARn[ECLNDA].)
27 -	This field is reserved. Reserved
28 EOSIE	End-of-segment interrupt enable 0 Do not generate an interrupt upon completion of the current DMA transfer for the current link descriptor. 1 Generate an interrupt upon completion of the current DMA transfer for the current link descriptor.
29–31 -	This field is reserved. Reserved

23.3.5 DMA source attributes register (DMA*x*_SATR*n*)

The source attributes registers contain the transaction attributes to be used for the DMA operation for channels 0-3. Stride mode is enabled by setting SATR*n*[SSME]. The target interface is derived from the local access window and outbound ATMU mappings and the transaction is obtained from the value specified in SATR*n*[SREADTTYPE].

Address: Base address + 110h offset + (128d × i), where i=0d to 3d

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

DMA*x*_SATR*n* field descriptions

Field	Description
0–6 -	This field is reserved. Reserved
7 SSME	Source stride mode enable. Ignored in basic mode (MRn[XFE] is cleared). Striding on the source address can be accomplished by enabling SATRn[SSME] and setting the desired stride size and distance in the SSRn. 0 Stride mode disabled 1 Stride mode enabled
8–11 -	This field is reserved. Reserved
12–15 SREADTTYPE	DMA source transaction type. Reserved values result in a programming error being detected and logged in SR[PE]. Transaction type to run on local address space . 0000-0011 Reserved 0100 Read, do not snoop local processor 0101 Read, snoop local processor 0110-1011 Reserved 1100 Enhanced read, don't snoop local processor

Table continues on the next page...

DMAx_SATRn field descriptions (continued)

Field	Description
	1101 Enhanced read, snoop local processor 1110-1111 Reserved
16–23 -	This field is reserved. Reserved
24–31 ESAD	Extended source address. ESAD represents the eight high-order bits of the 40-bit source address.

23.3.6 DMA source address register (DMAx_SARn)

The source address registers contain the address from which the DMA controller (channels 0-3) reads data. In direct mode, if MR*n*[CDSM_SWSM] and MR*n*[SRW] are set, a write to this register simultaneously sets MR*n*[CS], starting a DMA transfer. Software must ensure that this is a valid address.

Address: Base address + 114h offset + (128d × i), where i=0d to 3d

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	SAD																															
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

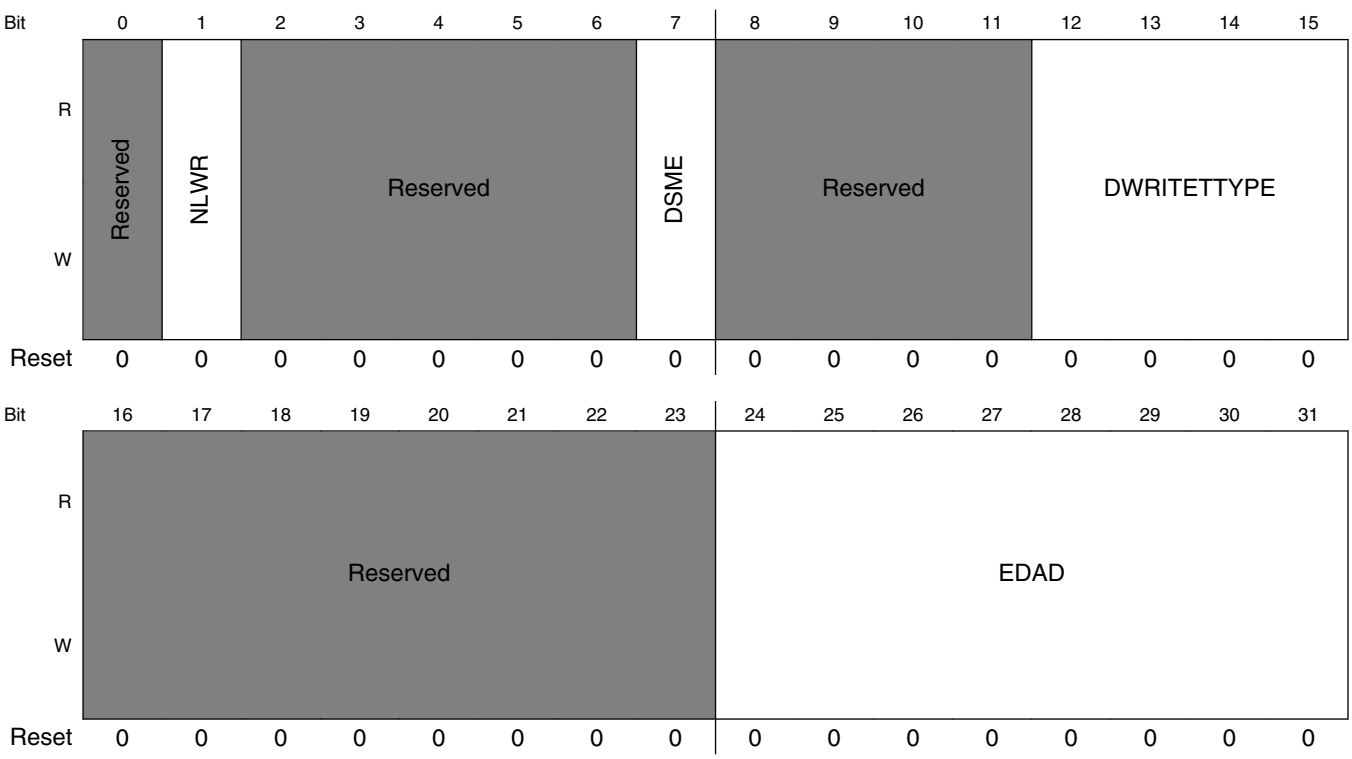
DMAx_SARn field descriptions

Field	Description
0–31 SAD	Source address. This register contains the low-order bits of the 40-bit source address of the DMA transfer for channels 0-3. The contents are updated after every DMA write operation unless the final stride of a striding operation is less than the stride size, in which case it remains equal to the address from which the last stride began.

23.3.7 DMA destination attributes register (DMA*x*_DATR*n*)

The destination attributes registers contain the transaction attributes for the DMA operation for DMA channels 0-3. Stride mode is enabled by setting DATR*n*[DSME]. The target interface is derived from the local access window and outbound ATMU mappings and the transaction is obtained from the value specified in DATR*n*[DWRITETTYPE].

Address: Base address + 118h offset + (128d × i), where i=0d to 3d



DMA*x*_DATR*n* field descriptions

Field	Description
0 - Reserved	This field is reserved.
1 NLWR	No last write with response. Performance impact: • When NLWR is cleared only one "write-with-response" transaction can be processed/in-flight at a time, which prevents pipelining of "write-with-response" transactions. • When NLWR is set "write-without-response" transactions can be issued in a pipelined manner not being stalled by a write response. Error impact:

Table continues on the next page...

DMAx_DATR*n* field descriptions (continued)

Field	Description
	• When NLWR is cleared the returned response includes error status which is used to update the SR[TE] field. • When NLWR is set, since no response is returned, the SR[TE] field is not updated with write transaction error status. 0 "Write-with-response" transactions are generated for the last transaction in a multi-transaction transfer. Single writes will be NWRITE_R. A response is returned from the module targeted by the write transfer (ex. DDR controller, eLBC, PCI-Express,...) informing the DMA of write completion and error status. 1 "Write-without-response" transactions are generated for the last transaction in a multi-transaction transfer. The DMA considers a transfer to be complete when the last write transaction is issued (not necessarily when the transaction has reached the target).
2–6 -	This field is reserved. Reserved
7 DSME	Destination stride mode enable Ignored in basic mode (MRn[XFE] is cleared). Striding on the destination address can be accomplished by setting DSME and setting the desired stride size and distance in DSRn. 0 Stride mode disabled 1 Stride mode enabled
8–11 -	This field is reserved. Reserved
12–15 DWRITETTYPE	DMA destination transaction type. Reserved values result in a programming error being detected and logged in SR[PE]. Transaction type to run on local address space. 0000-0011 Reserved 0100 Write, do not snoop local processor 0101 Write, snoop local processor 0110-1011 Reserved 1100 Enhanced write, don't snoop local processor 1101 Enhanced write, snoop local processor 1110-1111 Reserved
16–23 -	This field is reserved. Reserved
24–31 EDAD	Extended destination address. ESAD represents the eight high-order bits of the 40-bit destination address.

23.3.8 DMA destination address register (DMA*x*_DAR*n*)

The destination address registers contain the addresses to which the DMA controller (channels 0-3) writes data.

In direct mode, if MR*n*[SRW] is set and MR*n*[CDSM_SWSM] is cleared, a write to this register simultaneously sets MR*n*[CS], starting a DMA transfer. Software must ensure that this is a valid address.

Address: Base address + 11Ch offset + (128d × i), where i=0d to 3d

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	DAD																															
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

DMA*x*_DAR*n* field descriptions

Field	Description
0–31 DAD	Destination address. This field contains the low-order bits of the40-bit destination address of the DMA transfer. The contents are updated after every DMA write operation unless the final stride of a striding operation is less than the stride size, in which case it remains equal to the address from which the last stride began.

23.3.9 DMA byte count register (DMA*x*_BCR*n*)

The byte count register contains the number of bytes to transfer.

Address: Base address + 120h offset + (128d × i), where i=0d to 3d

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						BC																									
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

DMA*x*_BCR*n* field descriptions

Field	Description
0–5 -	This field is reserved. Reserved
6–31 BC	Byte count. Contains the number of bytes to transfer. The value in this register is decremented after each DMA read operation. The maximum transfer size is (2^{26}) - 1 bytes.

23.3.10 DMA extended next link descriptor address register (DMA*x*_ENLNDAR*n*)

These registers contain the upper 8 bits of the address of the next link descriptor in memory. See DMA next link descriptor address register (DMA_NLNDAR*n*)DMA next link descriptor address register (DMA_NLNDAR*n*).

Address: Base address + 124h offset + (128d × i), where i=0d to 3d

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																								ENLNDA							
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

DMA*x*_ENLNDAR*n* field descriptions

Field	Description
0–23 -	This field is reserved. Reserved
24–31 ENLNDA	Next link descriptor extended address bits (upper 8 bits of 40-bit address)

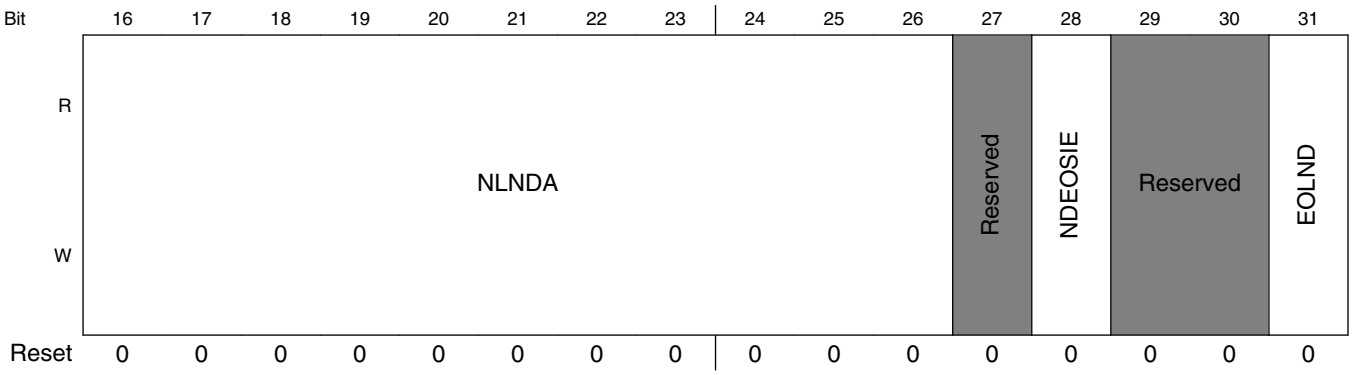
23.3.11 DMA next link descriptor address register (DMA*x*_NLNDAR*n*)

Current link descriptor address registers contain the address for the next link descriptor in memory.

Contents transferred to the current descriptor address registers become effective for the current transfer in basic and extended chaining modes.

Address: Base address + 128h offset + (128d × i), where i=0d to 3d

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



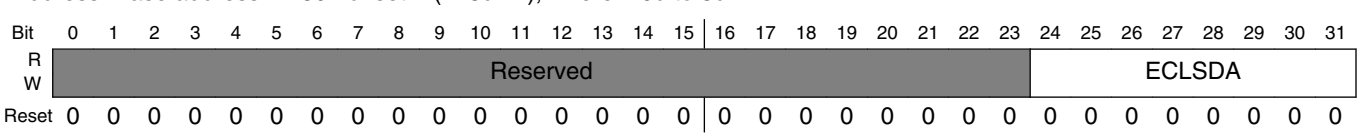
DMAx_NLNDAR*n* field descriptions

Field	Description
0–26 NLNDA	Next link descriptor address. Contains the bits 32-5 of the 40-bit next link descriptor address in memory. The descriptor must be aligned to a 32-byte boundary.
27 -	This field is reserved. Reserved
28 NDEOSIE	Next descriptor end-of-segment interrupt enable 0 Do not generate an interrupt if the current DMA transfer for the current descriptor is finished. 1 Generate an interrupt if the current DMA transfer for the current descriptor is finished.
29–30 -	This field is reserved. Reserved
31 EOLND	End-of-links descriptor. This bit is ignored in direct mode. 0 This descriptor is not the last link descriptor in memory for this list. 1 This descriptor is the last link descriptor in memory for this list. If this bit is set, the DMA controller advances to the next list descriptor in memory if NLSDARn[EOLSD] is also set in extended mode.

23.3.12 DMA extended current list descriptor address register (DMAx_ECLSDAR*n*)

These registers contain the upper 8 bits of the current address of the list descriptor in memory in extended chaining mode. See DMA current list descriptor address register (DMA_CLSDAR*n*)DMA current list descriptor address register (DMA_CLSDAR*n*).

Address: Base address + 130h offset + (128d × i), where i=0d to 3d



DMAx_ECLSDARn field descriptions

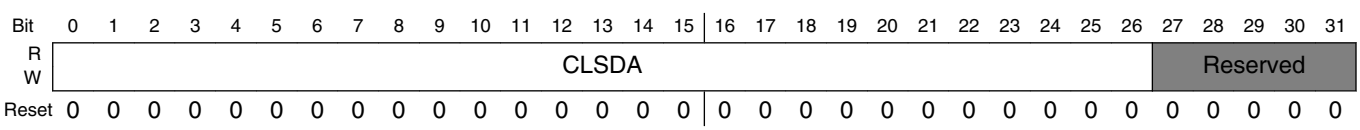
Field	Description
0–23 -	This field is reserved. Reserved
24–31 ECLSDA	Current list descriptor extended address bits (upper 8 bits of 40-bit address)

23.3.13 DMA current list descriptor address register (DMAx_CLSDARn)

The current list descriptor address registers contain the current address of the list descriptor in memory in extended chaining mode.

In extended chaining mode, software must initialize CLSDAR*n* and ECLSDAR *n* to point to the first list descriptor in memory. After finishing the last link descriptor in the current list, the DMA controller loads the contents of the next list descriptor address register into the current list descriptor address register. If NLSDAR *n* [EOLSD] in the next list descriptor address register is set and the last link in the current list is finished, all DMA transfers are complete.

Address: Base address + 134h offset + (128d × i), where i=0d to 3d



DMAx_CLSDARn field descriptions

Field	Description
0–26 CLSDA	Current list descriptor address. Contains the bits 32-5 of the 40-bit current list descriptor address of the buffer descriptor in memory in extended chaining mode. The descriptor must be aligned to a 32-byte boundary.
27–31 -	This field is reserved. Reserved

23.3.14 DMA extended next list descriptor address register (DMAx_ENLSDAR*n*)

These registers contain the upper 8 bits of the address of the next list descriptor in memory. See DMA next list descriptor address register (DMA_NLSDAR*n*)DMA next list descriptor address register (DMA_NLSDAR*n*).

Address: Base address + 138h offset + (128d × i), where i=0d to 3d

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																								ENLSDA							
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

DMAx_ENLSDAR*n* field descriptions

Field	Description
0–23 -	This field is reserved. Reserved
24–31 ENLSDA	Next list descriptor extended address bits (upper 8 bits of 40-bit address)

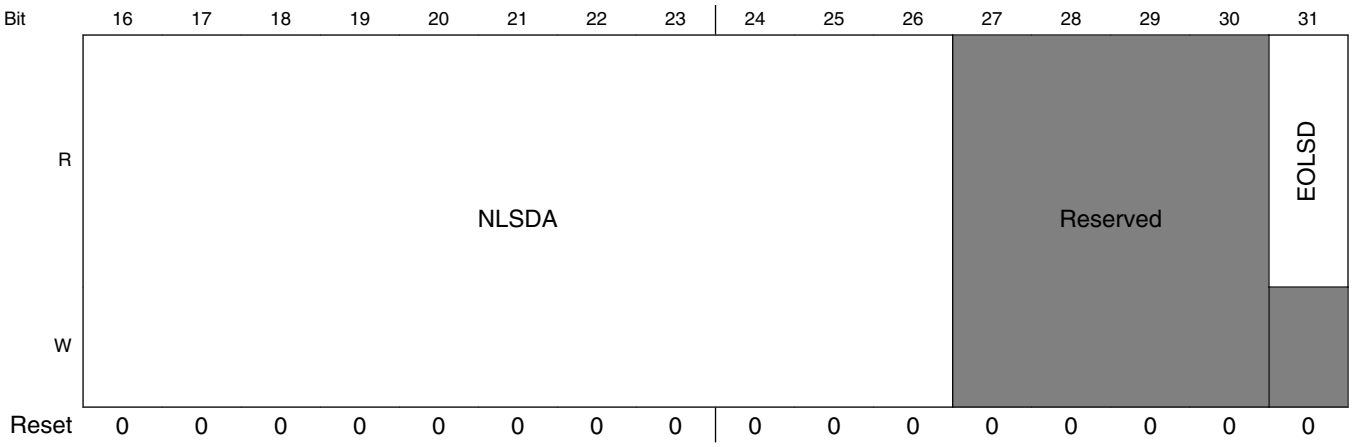
23.3.15 DMA next list descriptor address register (DMAx_NLSDAR*n*)

The next list descriptor address registers contain the address for the next list descriptor in memory.

If the contents are transferred to the current list descriptor address register, they become effective for the current transfer in extended chaining mode.

Address: Base address + 13Ch offset + (128d × i), where i=0d to 3d

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



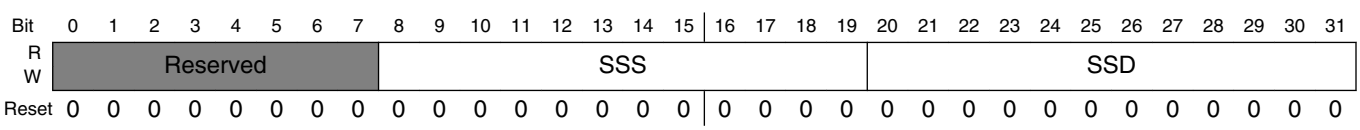
DMAx_NLSDAR*n* field descriptions

Field	Description
0–26 NLSDA	Next list descriptor address. Contains the bits 32-5 of the 40-bit next descriptor address of the buffer descriptor in memory. The descriptor must be aligned on a 32-byte boundary.
27–30 -	This field is reserved. Reserved
31 EOLSD	End-of-lists descriptor. This bit is ignored in direct mode. 0 This list descriptor is not the last list descriptor in memory. 1 This list descriptor is the last list descriptor in memory. If this bit is set, then the DMA controller halts after the last link descriptor transaction is finished.

23.3.16 DMA source stride register (DMAx_SSR*n*)

The source stride register contains the stride size and distance. Note that the source stride information is loaded when a new list descriptor is read from memory. Therefore, the source stride register is applicable for all link descriptors in the new list. Changing the source stride information for a link requires that a new list be generated.

Address: Base address + 140h offset + (128d × i), where i=0d to 3d



DMAx_SSR*n* field descriptions

Field	Description
0–7 -	This field is reserved. Reserved

Table continues on the next page...

DMAx_SSR*n* field descriptions (continued)

Field	Description
8–19 SSS	Source stride size. Number of bytes to transfer before jumping to the next address as specified in the source stride distance field.
20–31 SSD	Source stride distance. The source stride distance in bytes from start byte to start byte.

23.3.17 DMA destination stride register (DMAx_DSR*n*)

The destination stride register contains the stride size, and distance. Note that the destination stride information is loaded when a new list descriptor is read from memory. Therefore, the destination stride register is applicable for all link descriptors in the new list. Changing the destination stride information for a link requires that a new list be generated.

Address: Base address + 144h offset + (128d × i), where i=0d to 3d

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								DSS												DSD											
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

DMAx_DSR*n* field descriptions

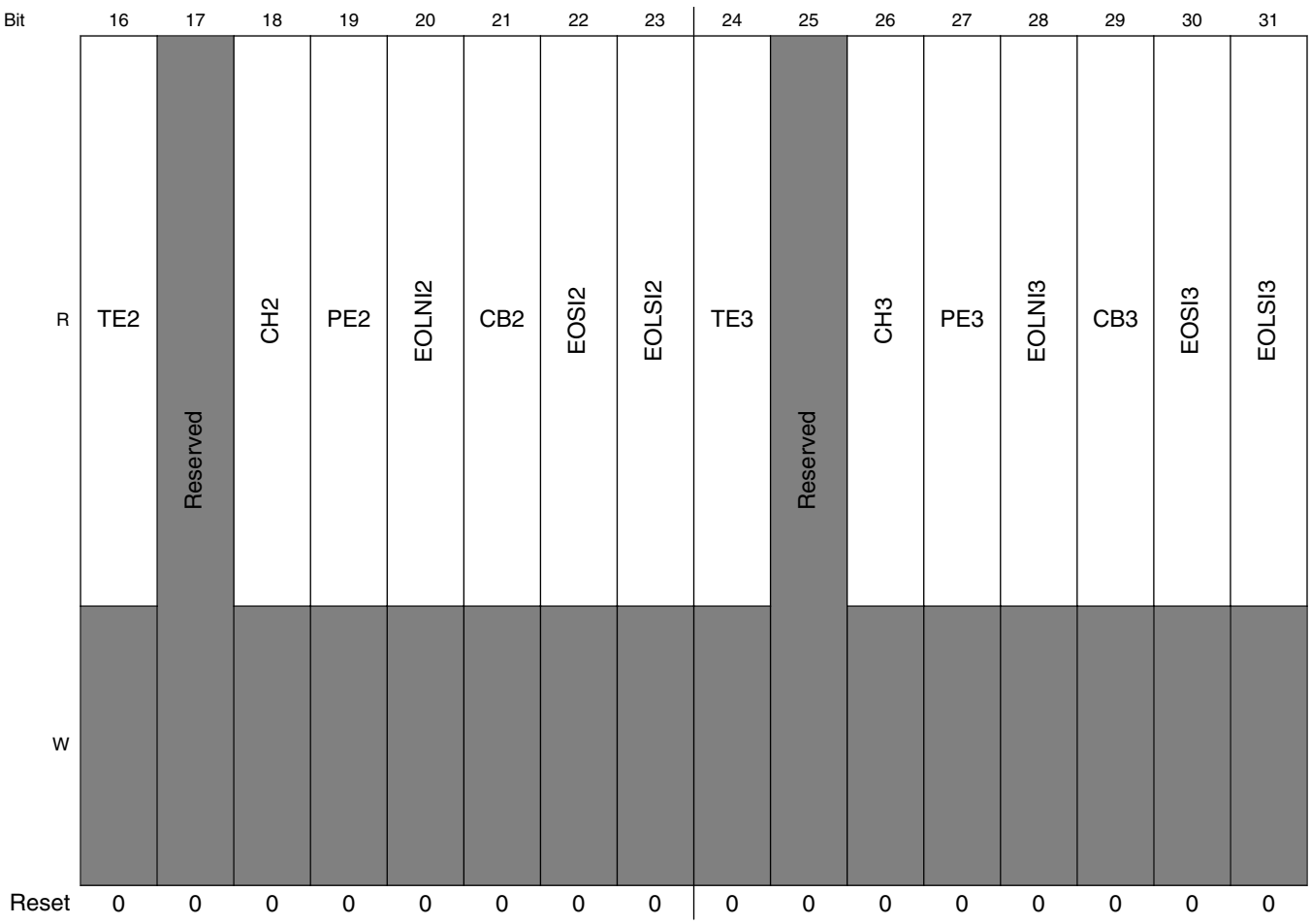
Field	Description
0–7 -	This field is reserved. Reserved
8–19 DSS	Destination stride size. Number of bytes to transfer before jumping to the next address as specified in the destination stride distance field.
20–31 DSD	Destination stride distance. The destination stride distance in bytes from start byte to start byte.

23.3.18 DMA general status register 0 (DMAx_DGSR0)

The DMA general status register 0 combines all of the status bits for channels 0-3 into one register. This register is read-only.

Address: Base address + 300h offset

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TE0	Reserved	CH0	PE0	EOLNI0	CB0	EOSI0	EOLSI0	TE1	Reserved	CH1	PE1	EOLNI1	CB1	EOSI1	EOLSI1
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



DMAx_DGSR0 field descriptions

Field	Description
0 TE0	Transfer error, channel 0 0 Normal operation 1 An error condition occurred during the DMA transfer.
1 -	This field is reserved. Reserved
2 CH0	Channel halted, channel 0
3 PE0	Programming error, channel 0
4 EOLNI0	End-of-links interrupt, channel 0
5 CB0	Channel busy, channel 0
6 EOSI0	End-of-segment interrupt, channel 0
7 EOLSI0	End-of-lists/direct interrupt, channel 0

Table continues on the next page...

DMAx_DGSR0 field descriptions (continued)

Field	Description
8 TE1	Transfer error, channel 1 0 Normal operation 1 An error condition occurred during the DMA transfer.
9 -	This field is reserved. Reserved
10 CH1	Channel halted, channel 1
11 PE1	Programming error, channel 1
12 EOLNI1	End-of-links interrupt, channel 1
13 CB1	Channel busy, channel 1
14 EOSI1	End-of-segment interrupt, channel 1
15 EOLSI1	End-of-lists/direct interrupt, channel 1
16 TE2	Transfer error, channel 2 0 Normal operation 1 An error condition occurred during the DMA transfer.
17 -	This field is reserved. Reserved
18 CH2	Channel halted, channel 2
19 PE2	Programming error, channel 2
20 EOLNI2	End-of-links interrupt, channel 2
21 CB2	Channel busy, channel 2
22 EOSI2	End-of-segment interrupt, channel 2
23 EOLSI2	End-of-lists/direct interrupt, channel 2
24 TE3	Transfer error, channel 3 0 Normal operation 1 An error condition occurred during the DMA transfer.
25 -	This field is reserved. Reserved
26 CH3	Channel halted, channel 3
27 PE3	Programming error, channel 3

Table continues on the next page...

DMAx_DGSR0 field descriptions (continued)

Field	Description
28 EOLNI3	End-of-links interrupt, channel 3
29 CB3	Channel busy, channel 3
30 EOSI3	End-of-segment interrupt, channel 3
31 EOLSI3	End-of-lists/direct interrupt, channel 3

23.3.19 DMA mode register (DMAx_MR*n*)

The mode register allows software to start a DMA transfer and to control various DMA transfer characteristics.

Address: Base address + 400h offset + (128d × i), where i=0d to 3d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved				BWC				Reserved		EMP_EN	Reserved		EMS_EN	DAHTS	
W																
Reset	0	0	0	0	1	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	SAHTS		DAHE	SAHE	Reserved	SRW	EOSIE	EOLNIE	EOLSIE	EIE	XFE	CDSM_SWSM	CA	CTM	CC	CS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DMAx_MR*n* field descriptions

Field	Description
0–3 -	This field is reserved. Reserved

Table continues on the next page...

DMAx_MR*n* field descriptions (continued)

Field	Description
4–7 BWC	Bandwidth/pause control. Defined when single and multiple channels are executing or if MRn[EMP_EN] is set in external transfer mode. The value of MRn[BWC] determines how many bytes a given channel is allowed to transfer before the DMA engine pauses the current channel and re-arbitrates (switches to the next channel). In external pause mode, the value of MRn[BWC] dictates how many bytes are allowed to transfer before pausing the channel, after which a new assertion of DREQ_B resumes channel operation. 0000 1 byte 0001 2 bytes 0010 4 bytes 0011 8 bytes 0100 16 bytes 0101 32 bytes 0110 64 bytes 0111 128 bytes 1000 256 bytes 1001 512 bytes 1010 1024 bytes 1011-1110 Reserved 1111 Disable bandwidth sharing to allow uninterrupted transfers from each channel.
8–9 -	This field is reserved. Reserved
10 EMP_EN	External master pause enable. Valid only if MRn[EMS_EN] is set. 0 Disable the external master pause feature. 1 Enable the external master pause feature. Channel is paused as described by MRn[BWC].
11–12 -	This field is reserved. Reserved
13 EMS_EN	External master start enable. This bit does not apply to single-write start modes (direct or chaining). 0 Disable the channel from being started by an external DMA start pin. 1 Enable the channel to be started by an external DMA start pin, which sets MRn[CS].
14–15 DAHTS	Destination address hold transfer size. Indicates the transfer size used for each transaction while MRn[DAHE] is set. The byte count register must be in multiples of the size and the destination address register must be aligned based on the size. The transfer size assigned to MRn[DAHTS] must be equal to or smaller than that assigned to MRn[BWC] to avoid undefined behavior. 00 1 byte 01 2 bytes 10 4 bytes 11 8 bytes
16–17 SAHTS	Source address hold transfer size. Indicates the transfer size used for each transaction while MRn[SAHE] is set. The byte count register must be in multiples of the size and the source address register must be aligned based on the size. The transfer size assigned to MRn[SAHTS] must be equal to or smaller than that assigned to MRn[BWC] to avoid undefined behavior. 00 1 byte 01 2 bytes

Table continues on the next page...

DMAx_MR*n* field descriptions (continued)

Field	Description
	10 4 bytes 11 8 bytes
18 DAHE	Destination address hold enable 0 Disable destination address hold 1 Enable the DMA controller to hold the destination address of a transfer to the size specified by MRn[DAHTS]. Hardware only supports aligned transfers for this feature.
19 SAHE	Source address hold enable 0 Disable source address hold 1 Enable the DMA controller to hold the source address of a transfer to the size specified by MRn[SAHTS]. Hardware only supports aligned transfers for this feature.
20 -	This field is reserved. Reserved
21 SRW	Single register write (Direct mode only; reserved for chaining mode.) 0 Normal operation 1 Enable a write to the source address register to simultaneously set MRn[CS], starting a DMA transfer, when MRn[CDSM_SWSM] is also set. Setting this bit and clearing CDSM_SWSM causes a write to the destination address register to simultaneously set MRn[CS], starting a DMA transfer.
22 EOSIE	End-of-segments interrupt enable 0 Do not generate an interrupt at the completion of a data transfer. CLNDARn[EOSIE] overrides this bit on a link descriptor basis. 1 Generate an interrupt at the completion of a data transfer (That is, SRn[EOSI] is set). This bit overrides the CLNDARn[EOSIE].
23 EOLNIE	End-of-links interrupt enable 0 Do not generate an interrupt at the completion of a list of DMA transfers. 1 Generate an interrupt at the completion of a list of DMA transfers (That is, NLNDARn[EOLND] is set).
24 EOLSIE	End-of-lists interrupt enable 0 Do not generate an interrupt at the completion of all DMA transfers. 1 Generate an interrupt at the completion of all DMA transfers (That is, NLNDARn[EOLND] and NLSDARn[EOLSD] are set).
25 EIE	Error interrupt enable 0 Do not generate an interrupt if a programming or transfer error is detected. 1 Generate an interrupt if a programming or transfer error is detected.
26 XFE	Extended features enable 0 Disable striding feature in direct mode or disable list chaining feature in chaining mode. 1 Enable striding feature in direct mode or enable list chaining feature in chaining mode.
27 CDSM_SWSM	In chaining mode, current descriptor start mode/single-write start mode is as follows: • In basic mode (MRn[XFE] is cleared), setting this bit causes a write to the current link descriptor address register to simultaneously set MRn[CS], starting a DMA transfer. • In extended chaining mode (MRn[XFE] is set), setting this bit causes a write to the current list descriptor address register to simultaneously set MRn[CS], starting a DMA transfer.

Table continues on the next page...

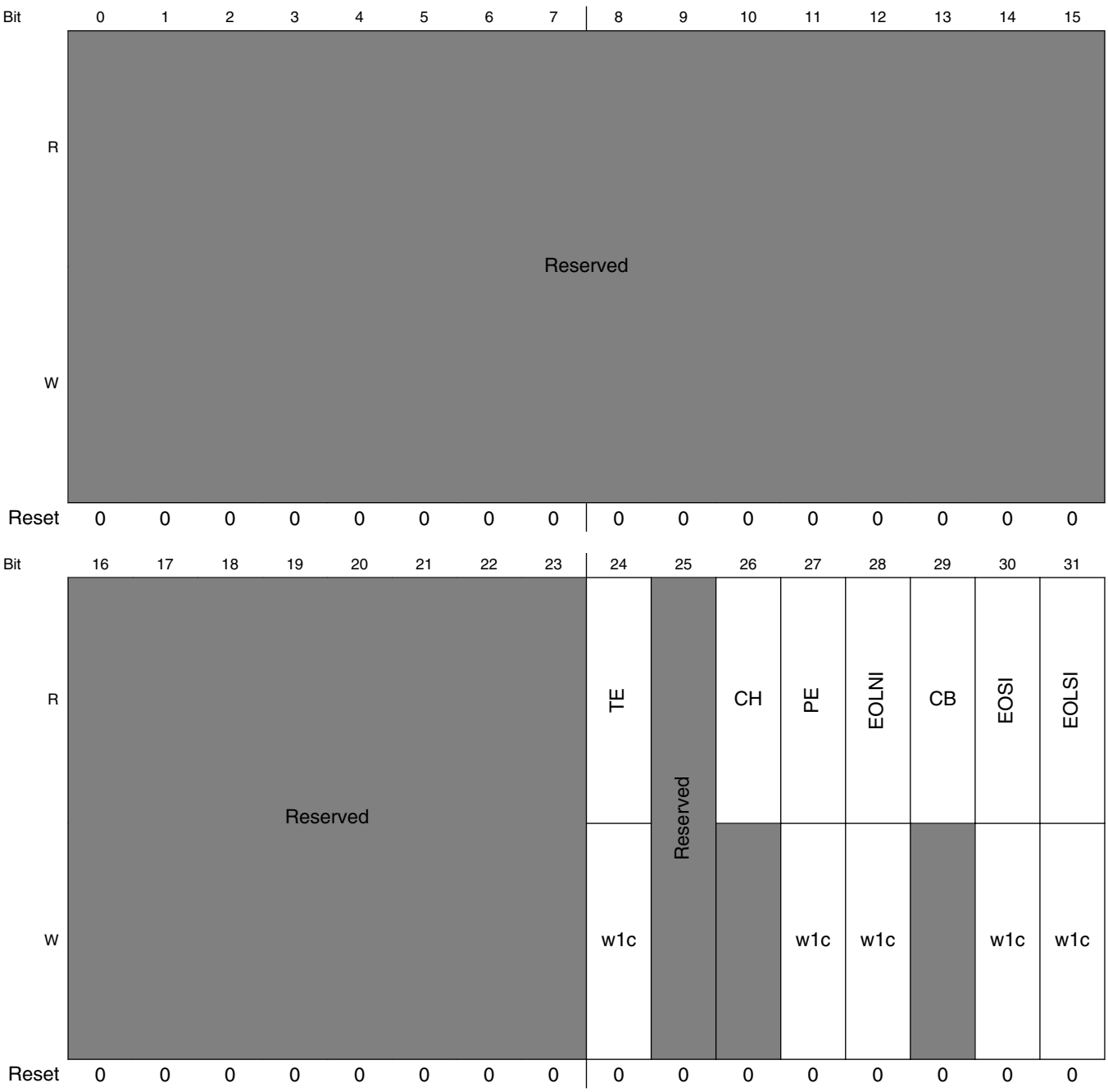
DMAx_MR*n* field descriptions (continued)

Field	Description
	In direct mode, setting this bit and MRn[SRW] causes a write to the source address register to simultaneously set MRn[CS], starting a DMA transfer. Clearing this bit and setting MRn[SRW] causes a write to the destination address register to simultaneously set MRn[CS], starting a DMA transfer. This bit must be cleared when MRn[SRW] is cleared.
28 CA	Channel abort 0 No effect 1 Cause the current transfer to be aborted and SRn[CB] to be cleared if the channel is busy. The channel remains in the idle state until a new transfer is programmed.
29 CTM	Channel transfer mode 0 Configure the channel in chaining mode. 1 Configure the channel into direct mode. This means that software is responsible for placing all the required parameters into necessary registers to start the DMA process.
30 CC	Channel continue. This bit applies only to chaining mode and is cleared by hardware after the first descriptor read when continuing a transfer. This bit is reserved for external master mode. 0 No effect 1 The DMA transfer restarts the transferring process starting at the current descriptor address.
31 CS	Channel start. This bit is also automatically set by hardware during single-write start mode and external master start enable mode. Note that in external control mode, deasserting DMA_DREQ_B does NOT clear this bit. 0 Halt the DMA process if channel is busy (SRn[CB] is set). No effect if the channel is not busy. 1 Start the DMA process if channel is not busy (CB is cleared). If the channel was halted (CS = 0 and CB = 1), the transfer continues from the point at which it was halted.

23.3.20 DMA status register (DMA*x*_SR*n*)

The status registers report various DMA conditions during and after a DMA transfer.

Address: Base address + 404h offset + (128d × i), where i=0d to 3d



DMAx_SRn field descriptions

Field	Description
0–23 -	This field is reserved. Reserved
24 TE	Transfer error (Bit reset, write 1 to clear) 0 No error condition during the DMA transfer 1 Error condition during the DMA transfer. See DMA errors , for additional information.
25 -	This field is reserved. Reserved
26 CH	Channel halted. Cleared automatically by hardware if MR *n* [CS] is set again for resuming a halted transfer 0 Channel is not halted. If software attempts to halt an idle channel (SRn[CB] is cleared), this bit remains 0. 1 DMA transfer was successfully halted by software and can be resumed.
27 PE	Programming error (bit reset, write 1 to clear) 0 No programming error detected 1 A programming error is detected that prevents the DMA transfer from occurring.
28 EOLNI	End-of-links interrupt. After transferring the last block of data in the last link descriptor, if MRn[EOLNIE] is set, then this bit is set and an interrupt is generated. (Bit reset, write 1 to clear)
29 CB	Channel busy 0 DMA transfer is finished, an error occurred, or a channel abort occurred. 1 DMA transfer is currently in progress.
30 EOSI	End-of-segment interrupt. In chaining mode, after finishing a data transfer, if MRn[EOSIE] is set or if CLNDARn[EOSIE] is set, this bit gets set and an interrupt is generated. In direct mode, if MRn[EOSIE] is set, this bit gets set and an interrupt is generated. (Bit reset, write 1 to clear)
31 EOLSI	End-of-list interrupt. After transferring the last block of data in the last list descriptor, if MRn[EOLSIE] is set, then this bit is set and an interrupt is generated. (Bit reset, write 1 to clear)

23.3.21 DMA extended current link descriptor address register (DMAx_ECLNDARn)

These registers contain the upper 8 bits of the address of the current link descriptor. See DMA current link descriptor address register (DMA_CLNDAR*n*)DMA current link descriptor address register (DMA_CLNDAR*n*) for a description of basic chaining mode.

Address: Base address + 408h offset + (128d × i), where i=0d to 3d

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																								ECLNDA							
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

DMAx_ECLNDAR*n* field descriptions

Field	Description
0–23 -	This field is reserved. Reserved
24–31 ECLNDA	Current link descriptor extended address (upper 8 bits of 40-bit address)

23.3.22 DMA current link descriptor address register (DMAx_CLNDAR*n*)

Current link descriptor address registers contain the address of the current link descriptor.

In basic chaining mode, shown in , software must initialize these registers to point to the first link descriptors in memory.

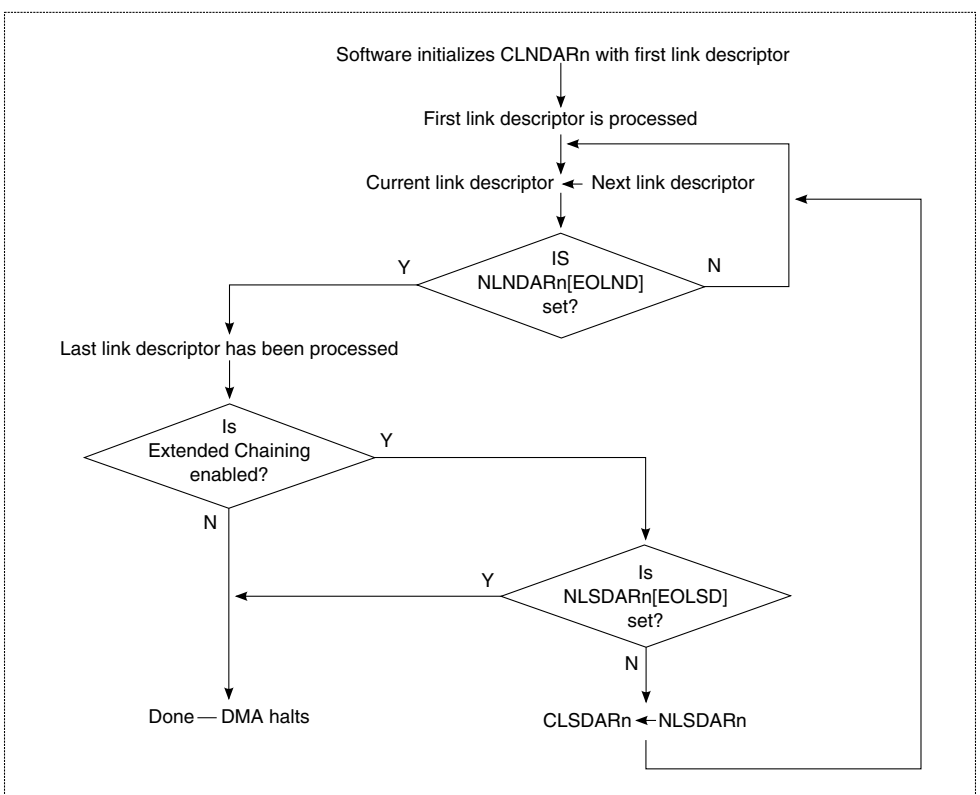
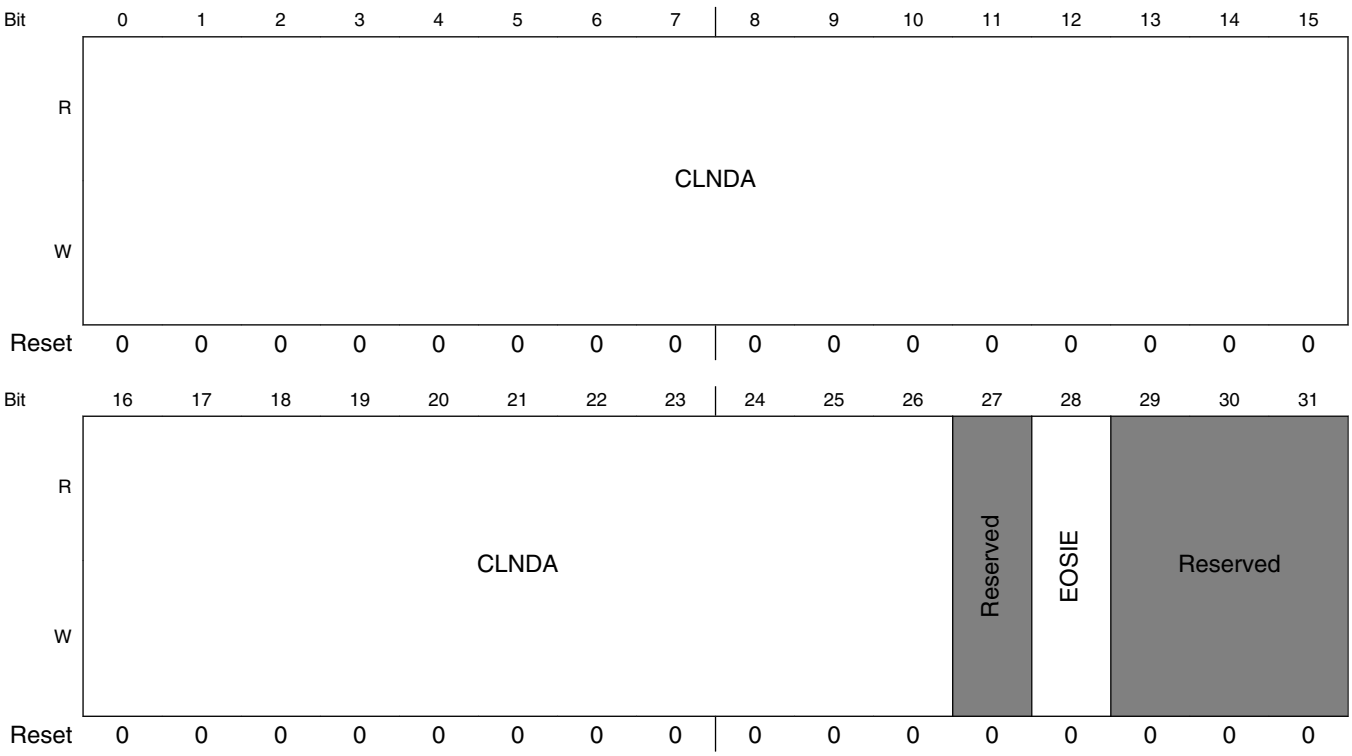


Figure 23-286. Basic Chaining Mode Flow Chart

After the current descriptor is processed, the current link descriptor address register is loaded from the next link descriptor address register and NLNDAR *n* [EOLND] in the next link descriptor address register is examined. If EOLND is zero, the DMA controller reads in the new current link descriptor for processing. If EOLND is set, the last descriptor of the list was just completed. If extended chaining mode is not enabled, all DMA transfers are complete and the DMA controller halts.

If extended chaining mode is enabled, the DMA controller examines the state of NLSDAR *n* [EOLSD] in the next list descriptor address register. If EOLSD is clear, the controller loads the contents of the next list descriptor address register into the current list descriptor address register and reads the new list descriptor from memory. If EOLSD is set, all DMA transfers are complete and the DMA controller halts.

Address: Base address + 40Ch offset + (128d × i), where i=0d to 3d



DMA*x*_CLNDAR*n* field descriptions

Field	Description
0–26 CLNDA	Current link descriptor address. Contains the current descriptor address of the buffer descriptor in memory. The descriptor must be aligned to a 32-byte boundary. (This is the lower portion of the 40 -bit address formed by CLNDARn[CLNDA] and ECLNDARn[ECLNDA].)
27 -	This field is reserved. Reserved
28 EOSIE	End-of-segment interrupt enable 0 Do not generate an interrupt upon completion of the current DMA transfer for the current link descriptor. 1 Generate an interrupt upon completion of the current DMA transfer for the current link descriptor.
29–31 -	This field is reserved. Reserved

23.3.23 DMA source attributes register (DMA*x*_SATR*n*)

The source attributes registers contain the transaction attributes to be used for the DMA operation for channels 4-7. Stride mode is enabled by setting SATR*n*[SSME]. The target interface is derived from the local access window and outbound ATMU mappings and the transaction is obtained from the value specified in SATR*n*[SREADTTYPE].

Address: Base address + 410h offset + (128d × i), where i=0d to 3d

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

DMA*x*_SATR*n* field descriptions

Field	Description
0–6 -	This field is reserved. Reserved
7 SSME	Source stride mode enable. Ignored in basic mode (MRn[XFE] is cleared). Striding on the source address can be accomplished by enabling SATRn[SSME] and setting the desired stride size and distance in the SSRn. 0 Stride mode disabled 1 Stride mode enabled
8–11 -	This field is reserved. Reserved
12–15 SREADTTYPE	DMA source transaction type. Reserved values result in a programming error being detected and logged in SR[PE]. Transaction type to run on local address space . 0000-0011 Reserved 0100 Read, do not snoop local processor 0101 Read, snoop local processor 0110-1011 Reserved 1100 Enhanced read, don't snoop local processor

Table continues on the next page...

DMAx_SATRn field descriptions (continued)

Field	Description
	1101 Enhanced read, snoop local processor 1110-1111 Reserved
16–23 -	This field is reserved. Reserved
24–31 ESAD	Extended source address. ESAD represents the eight high-order bits of the 40-bit source address.

23.3.24 DMA source address register (DMAx_SARn)

The source address registers contain the address from which the DMA controller (channels 4-7) reads data. In direct mode, if MR*n*[CDSM_SWSM] and MR*n* [SRW]are set, a write to this register simultaneously sets MR*n*[CS], starting a DMA transfer. Software must ensure that this is a valid address.

Address: Base address + 414h offset + (128d × i), where i=0d to 3d

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	SAD																															
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

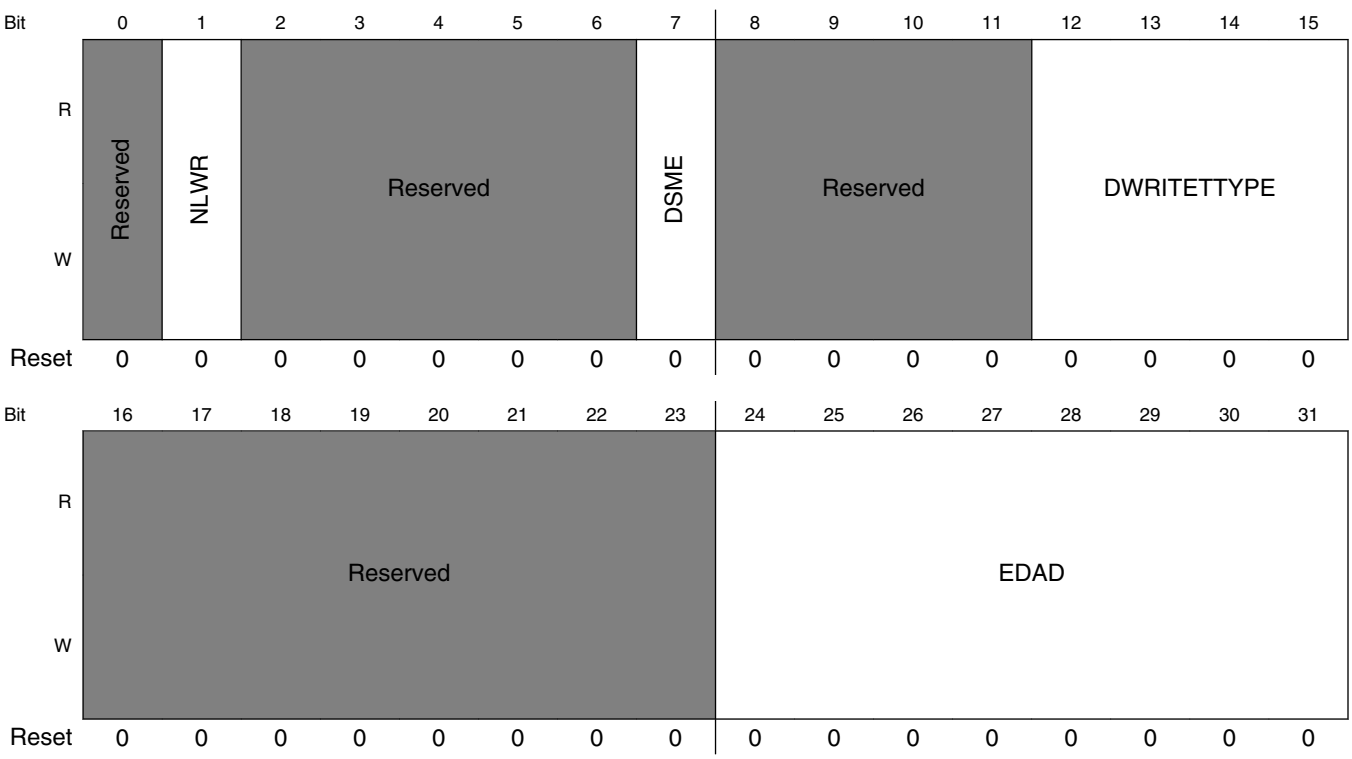
DMAx_SARn field descriptions

Field	Description
0–31 SAD	Source address. This register contains the low-order bits of the 40-bit source address of the DMA transfer for channels 4-7. The contents are updated after every DMA write operation unless the final stride of a striding operation is less than the stride size, in which case it remains equal to the address from which the last stride began.

23.3.25 DMA destination attributes register (DMAx_DATR*n*)

The destination attributes registers contain the transaction attributes for the DMA operation for DMA channels 4-7. Stride mode is enabled by setting DATR*n*[DSME]. The target interface is derived from the local access window and outbound ATMU mappings and the transaction is obtained from the value specified in DATR*n*[DWRITETTYPE].

Address: Base address + 418h offset + (128d × i), where i=0d to 3d



DMAx_DATR*n* field descriptions

Field	Description
0 -	This field is reserved. Reserved
1 NLWR	No last write with response. Performance impact: • When NLWR is cleared only one "write-with-response" transaction can be processed/in-flight at a time, which prevents pipelining of "write-with-response" transactions. • When NLWR is set "write-without-response" transactions can be issued in a pipelined manner not being stalled by a write response. Error impact:

Table continues on the next page...

DMAx_DATR*n* field descriptions (continued)

Field	Description
	• When NLWR is cleared the returned response includes error status which is used to update the SR[TE] field. • When NLWR is set, since no response is returned, the SR[TE] field is not updated with write transaction error status. 0 "Write-with-response" transactions are generated for the last transaction in a multi-transaction transfer. Single writes will be NWRITE_R. A response is returned from the module targeted by the write transfer (ex. DDR controller, eLBC, PCI-Express,...) informing the DMA of write completion and error status. 1 "Write-without-response" transactions are generated for the last transaction in a multi-transaction transfer. The DMA considers a transfer to be complete when the last write transaction is issued (not necessarily when the transaction has reached the target).
2–6 -	This field is reserved. Reserved
7 DSME	Destination stride mode enable Ignored in basic mode (MRn[XFE] is cleared). Striding on the destination address can be accomplished by setting DSME and setting the desired stride size and distance in DSRn. 0 Stride mode disabled 1 Stride mode enabled
8–11 -	This field is reserved. Reserved
12–15 DWRITETTYPE	DMA destination transaction type. Reserved values result in a programming error being detected and logged in SR[PE]. Transaction type to run on local address space. 0000-0011 Reserved 0100 Write, do not snoop local processor 0101 Write, snoop local processor 0110-1011 Reserved 1100 Enhanced write, don't snoop local processor 1101 Enhanced write, snoop local processor 1110-1111 Reserved
16–23 -	This field is reserved. Reserved
24–31 EDAD	Extended destination address. ESAD represents the eight high-order bits of the 40-bit destination address.

23.3.26 DMA destination address register (DMAx_DAR*n*)

The destination address registers contain the addresses to which the DMA controller (channels 4-7) writes data.

In direct mode, if MR*n*[SRW] is set and MR*n*[CDSM_SWSM] is cleared, a write to this register simultaneously sets MR*n*[CS], starting a DMA transfer. Software must ensure that this is a valid address.

Address: Base address + 41Ch offset + (128d × i), where i=0d to 3d

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	DAD																															
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

DMA*x*_DAR*n* field descriptions

Field	Description
0–31 DAD	Destination address. This field contains the low-order bits of the40-bit destination address of the DMA transfer. The contents are updated after every DMA write operation unless the final stride of a striding operation is less than the stride size, in which case it remains equal to the address from which the last stride began.

23.3.27 DMA byte count register (DMA*x*_BCR*n*)

The byte count register contains the number of bytes to transfer.

Address: Base address + 420h offset + (128d × i), where i=0d to 3d

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						BC																									
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

DMA*x*_BCR*n* field descriptions

Field	Description
0–5 -	This field is reserved. Reserved
6–31 BC	Byte count. Contains the number of bytes to transfer. The value in this register is decremented after each DMA read operation. The maximum transfer size is (2^{26}) - 1 bytes.

23.3.28 DMA next link descriptor extended address register (DMA*x*_ENLNDAR*n*)

These registers contain the upper 8 bits of the address of the next link descriptor in memory. See DMA next link descriptor address register (DMA_NLNDAR*n*)DMA next link descriptor address register (DMA_NLNDAR*n*).

Address: Base address + 424h offset + (128d × i), where i=0d to 3d

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																								ENLNDA							
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

DMA*x*_ENLNDAR*n* field descriptions

Field	Description
0–23 -	This field is reserved. Reserved
24–31 ENLNDA	Next link descriptor extended address bits (upper 8 bits of 40-bit address)

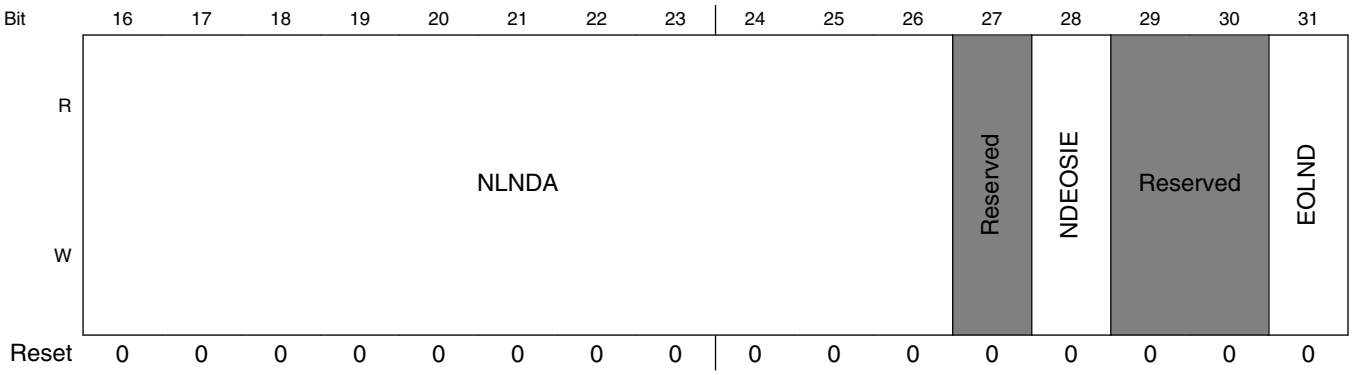
23.3.29 DMA next link descriptor address register (DMA*x*_NLNDAR*n*)

Current link descriptor address registers contain the address for the next link descriptor in memory.

Contents transferred to the current descriptor address registers become effective for the current transfer in basic and extended chaining modes.

Address: Base address + 428h offset + (128d × i), where i=0d to 3d

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



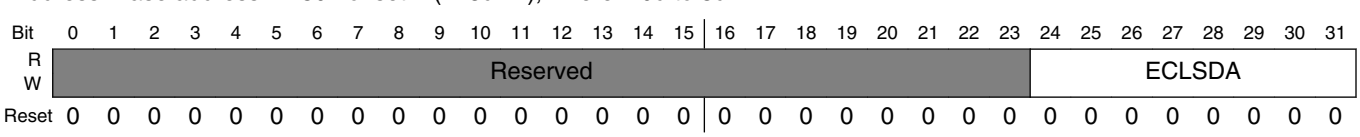
DMAx_NLNDAR*n* field descriptions

Field	Description
0–26 NLNDA	Next link descriptor address. Contains the bits 32-5 of the 40-bit next link descriptor address in memory. The descriptor must be aligned to a 32-byte boundary.
27 -	This field is reserved. Reserved
28 NDEOSIE	Next descriptor end-of-segment interrupt enable 0 Do not generate an interrupt if the current DMA transfer for the current descriptor is finished. 1 Generate an interrupt if the current DMA transfer for the current descriptor is finished.
29–30 -	This field is reserved. Reserved
31 EOLND	End-of-links descriptor. This bit is ignored in direct mode. 0 This descriptor is not the last link descriptor in memory for this list. 1 This descriptor is the last link descriptor in memory for this list. If this bit is set, the DMA controller advances to the next list descriptor in memory if NLSDARn[EOLSD] is also set in extended mode.

23.3.30 DMA extended current list descriptor address register (DMAx_ECLSDAR*n*)

These registers contain the upper 8 bits of the current address of the list descriptor in memory in extended chaining mode. See DMA current list descriptor address register (DMA_CLSDAR*n*)DMA current list descriptor address register (DMA_CLSDAR*n*).

Address: Base address + 430h offset + (128d × i), where i=0d to 3d



DMAx_ECLSDARn field descriptions

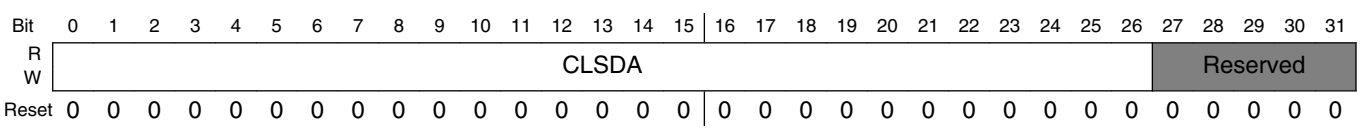
Field	Description
0–23 -	This field is reserved. Reserved
24–31 ECLSDA	Current list descriptor extended address bits (upper 8 bits of 40-bit address)

23.3.31 DMA current list descriptor address register (DMAx_CLSDARn)

The current list descriptor address registers contain the current address of the list descriptor in memory in extended chaining mode.

In extended chaining mode, software must initialize CLSDAR*n* and ECLSDAR *n* to point to the first list descriptor in memory. After finishing the last link descriptor in the current list, the DMA controller loads the contents of the next list descriptor address register into the current list descriptor address register. If NLSDAR *n* [EOLSD] in the next list descriptor address register is set and the last link in the current list is finished, all DMA transfers are complete.

Address: Base address + 434h offset + (128d × i), where i=0d to 3d



DMAx_CLSDARn field descriptions

Field	Description
0–26 CLSDA	Current list descriptor address. Contains the bits 32-5 of the 40-bit current list descriptor address of the buffer descriptor in memory in extended chaining mode. The descriptor must be aligned to a 32-byte boundary.
27–31 -	This field is reserved. Reserved

23.3.32 DMA extended next list descriptor address register (DMAx_ENLSDAR*n*)

These registers contain the upper 8 bits of the address of the next list descriptor in memory. See DMA next list descriptor address register (DMA_NLSDAR*n*)DMA next list descriptor address register (DMA_NLSDAR*n*).

Address: Base address + 438h offset + (128d × i), where i=0d to 3d

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																								ENLSDA							
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

DMAx_ENLSDAR*n* field descriptions

Field	Description
0–23 -	This field is reserved. Reserved
24–31 ENLSDA	Next list descriptor extended address bits (upper 8 bits of 40-bit address)

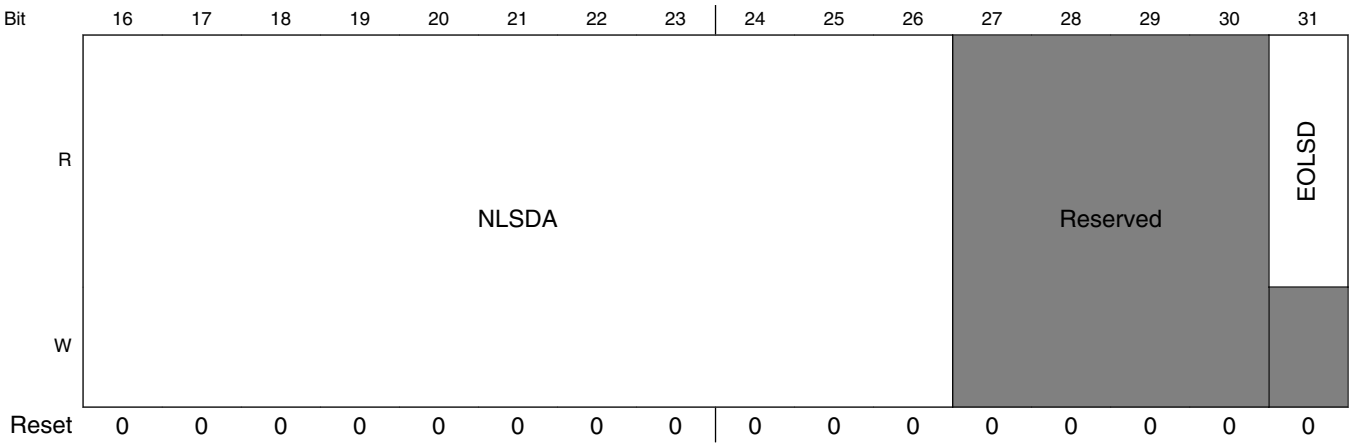
23.3.33 DMA next list descriptor address register (DMAx_NLSDAR*n*)

The next list descriptor address registers contain the address for the next list descriptor in memory.

If the contents are transferred to the current list descriptor address register, they become effective for the current transfer in extended chaining mode.

Address: Base address + 43Ch offset + (128d × i), where i=0d to 3d

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



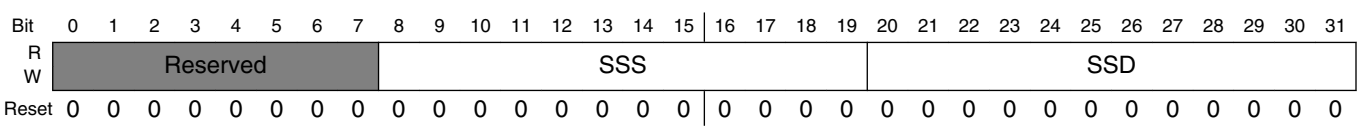
DMAx_NLSDAR*n* field descriptions

Field	Description
0–26 NLSDA	Next list descriptor address. Contains the bits 32-5 of the 40 -bit next descriptor address of the buffer descriptor in memory. The descriptor must be aligned on a 32-byte boundary.
27–30 -	This field is reserved. Reserved
31 EOLSD	End-of-lists descriptor. This bit is ignored in direct mode. 0 This list descriptor is not the last list descriptor in memory. 1 This list descriptor is the last list descriptor in memory. If this bit is set, then the DMA controller halts after the last link descriptor transaction is finished.

23.3.34 DMA source stride register (DMAx_SSR*n*)

The source stride register contains the stride size and distance. Note that the source stride information is loaded when a new list descriptor is read from memory. Therefore, the source stride register is applicable for all link descriptors in the new list. Changing the source stride information for a link requires that a new list be generated.

Address: Base address + 440h offset + (128d × i), where i=0d to 3d



DMAx_SSR*n* field descriptions

Field	Description
0–7 -	This field is reserved. Reserved

Table continues on the next page...

DMAx_SSR*n* field descriptions (continued)

Field	Description
8–19 SSS	Source stride size. Number of bytes to transfer before jumping to the next address as specified in the source stride distance field.
20–31 SSD	Source stride distance. The source stride distance in bytes from start byte to start byte.

23.3.35 DMA destination stride register (DMAx_DSR*n*)

The destination stride register contains the stride size, and distance. Note that the destination stride information is loaded when a new list descriptor is read from memory. Therefore, the destination stride register is applicable for all link descriptors in the new list. Changing the destination stride information for a link requires that a new list be generated.

Address: Base address + 444h offset + (128d × i), where i=0d to 3d

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								DSS												DSD											
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

DMAx_DSR*n* field descriptions

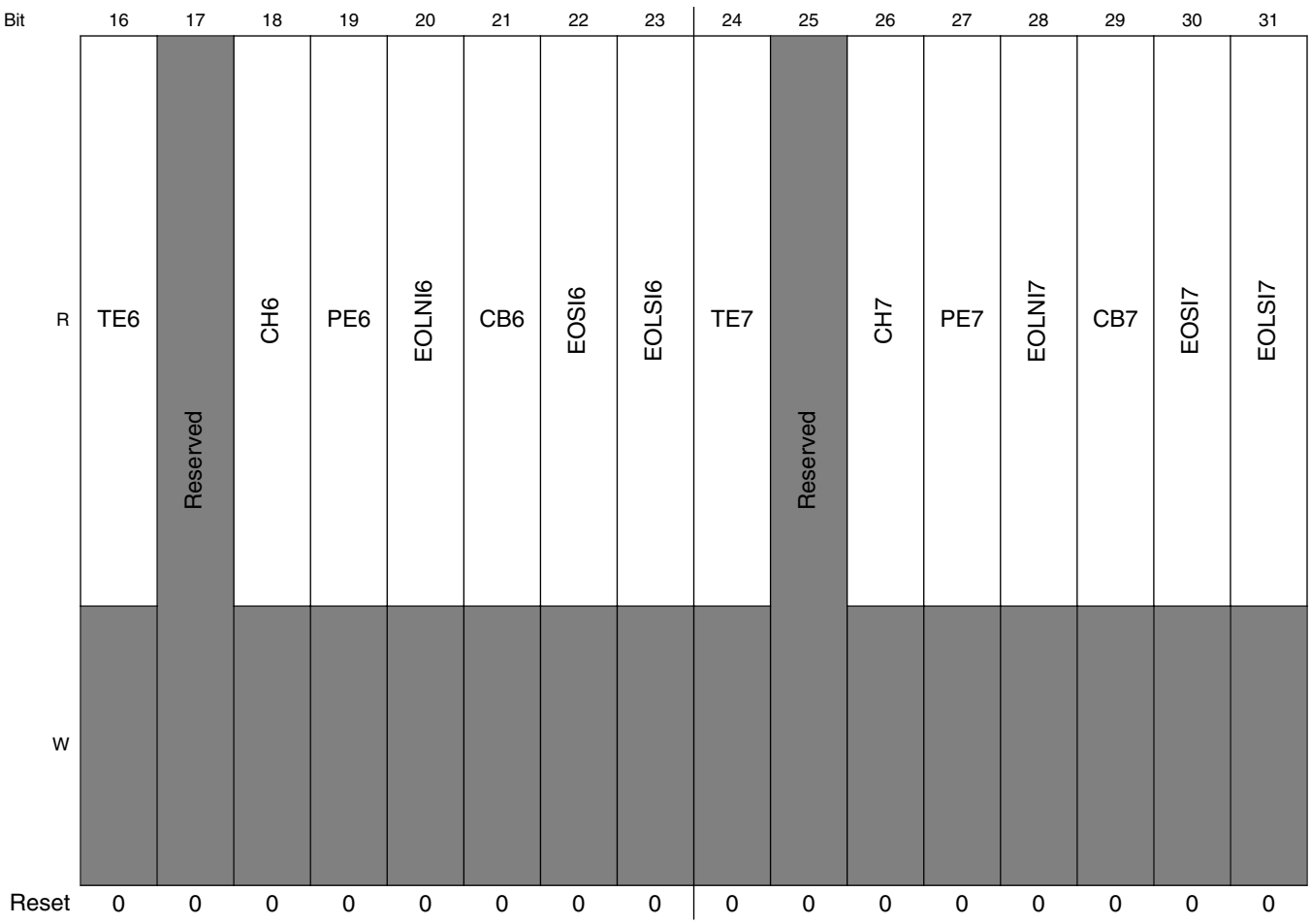
Field	Description
0–7 -	This field is reserved. Reserved
8–19 DSS	Destination stride size. Number of bytes to transfer before jumping to the next address as specified in the destination stride distance field.
20–31 DSD	Destination stride distance. The destination stride distance in bytes from start byte to start byte.

23.3.36 DMA general status register 1 (DMAx_DGSR1)

The DMA general status register 1 combines all of the status bits for channels 4-7 into one register This register is read-only.

Address: Base address + 600h offset

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	TE4	Reserved	CH4	PE4	EOLNI4	CB4	EOSI4	EOLSI4	TE5	Reserved	CH5	PE5	EOLNI5	CB5	EOSI5	EOLSI5
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



DMAx_DGSR1 field descriptions

Field	Description
0 TE4	Transfer error, channel 4 0 Normal operation 1 An error condition occurred during the DMA transfer.
1 -	This field is reserved. Reserved
2 CH4	Channel halted, channel 4
3 PE4	Programming error, channel 4
4 EOLNI4	End-of-links interrupt, channel 4
5 CB4	Channel busy, channel 4
6 EOSI4	End-of-segment interrupt, channel 4
7 EOLSI4	End-of-lists/direct interrupt, channel 4

Table continues on the next page...

DMAx_DGSR1 field descriptions (continued)

Field	Description
8 TE5	Transfer error, channel 5 0 Normal operation 1 An error condition occurred during the DMA transfer.
9 -	This field is reserved. Reserved
10 CH5	Channel halted, channel 5
11 PE5	Programming error, channel 5
12 EOLNI5	End-of-links interrupt, channel 5
13 CB5	Channel busy, channel 5
14 EOSI5	End-of-segment interrupt, channel 5
15 EOLSI5	End-of-lists/direct interrupt, channel 5
16 TE6	Transfer error, channel 6 0 Normal operation 1 An error condition occurred during the DMA transfer.
17 -	This field is reserved. Reserved
18 CH6	Channel halted, channel 6
19 PE6	Programming error, channel 6
20 EOLNI6	End-of-links interrupt, channel 6
21 CB6	Channel busy, channel 6
22 EOSI6	End-of-segment interrupt, channel 6
23 EOLSI6	End-of-lists/direct interrupt, channel 6
24 TE7	Transfer error, channel 7 0 Normal operation 1 An error condition occurred during the DMA transfer.
25 -	This field is reserved. Reserved
26 CH7	Channel halted, channel 7
27 PE7	Programming error, channel 7

Table continues on the next page...

DMAx_DGSR1 field descriptions (continued)

Field	Description
28 EOLNI7	End-of-links interrupt, channel 7
29 CB7	Channel busy, channel 7
30 EOSI7	End-of-segment interrupt, channel 7
31 EOLSI7	End-of-lists/direct interrupt, channel 7

23.4 DMA functional description

This section describes the function of the DMA controller.

23.4.1 DMA channel operation

All DMA channels support two different modes of operation: a basic mode (MR*n*[XFE] is cleared) and an extended mode (MR*n*[XFE] is set). In both modes, a channel can be activated by clearing and setting MR*n*[CS], or through the single-write start mode using MR*n*[CDSM_SWSM] and MR*n*[SRW], or through an external control mode using MR*n*[EMS_EN] and the external DMA_DREQ signal.

In basic mode, the channel can be programmed in basic direct mode or basic chaining mode. In extended mode, the channel can be programmed in extended direct mode or extended chaining mode. Extended mode provides more capabilities, such as extended descriptor chaining, striding capabilities, and a more flexible descriptor structure.

The DMA controller supports misaligned transfers for both the source and destination addresses. In order to maximize performance, the source and destination engines issue one or more transactions to reach the desired alignment based on the rules described in Source/destination transaction size calculations. The DMA always reads/writes the maximum number of bytes for a given transfer as described by the capability inputs of the DMA controller except for globally coherent transactions that use the size of the cache coherence granule as described by the mode select input. Using 256 bytes over the RapidIO interface reduces packet overhead which translates to increased bandwidth utilization through the interface.

The DMA controller supports bandwidth control, which prevents a channel from consuming all the data bandwidth in the controller. Each channel is allowed to consume the bandwidth of the shared resources as specified by the bandwidth control value. After

the channel uses its allotted bandwidth, the arbiter grants the next channel access to the shared resources. The arbitration is round robin between the channels. This feature is also used to implement the external control pause feature. If the external control start and pause are enabled in the MR*n*, the channel enters a paused state after transferring the data described in the bandwidth control. External control can restart the channel from a paused state.

The DMA programming model permits software to program each DMA engine independently to interrupt on completed segment, chain, or error. It also provides the capability for software to resume the DMA engine from a hardware halted condition by setting the channel continue bit, MR*n*[CC]. See Table 23-692 for more complete descriptions of the channel states and state transitions.

23.4.1.1 Source/destination transaction size calculations

The DMA controller may issue smaller transactions from the source and destination address engines in an effort to reach alignment for improved performance.

The flow chart below shows the decision points made in determining the transaction size. The starting *Txfer_Size* is determined by min(BCR[BC],MR[BWC]), *Stride_En* is determined by SATR*n*[SSME] or DATR*n*[DSME], and *Stride_Size* is determined by SSR*n*[SSS] or DSR*n*[DSS].

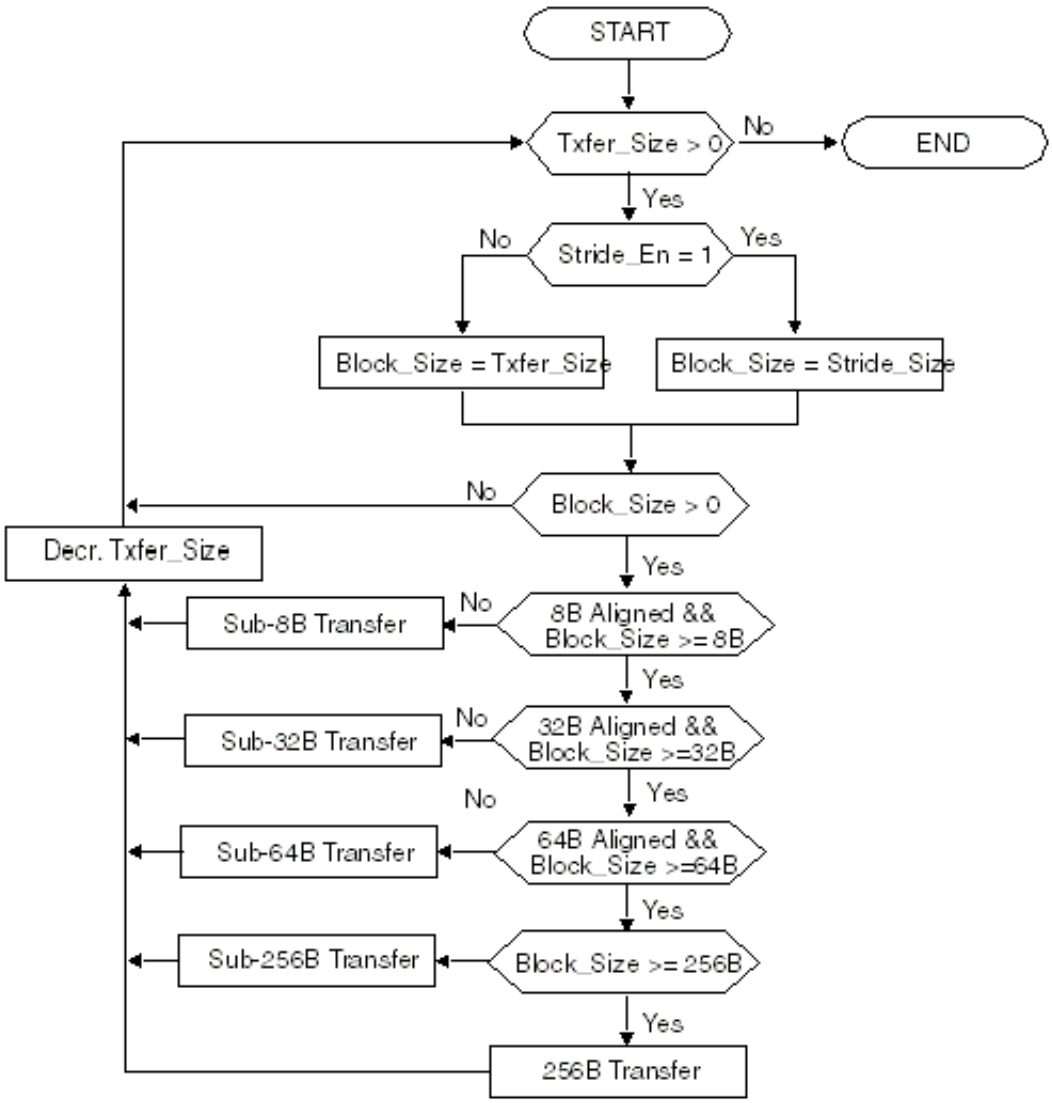


Figure 23-732. Source/destination engine transaction size flow chart

For example, if BCR[BC]=512 bytes and MR[BWC]=256 bytes, reading from starting address 0x5D0 will result in the following transaction sizes:

```
-- Channel arbitration --
0x5D0 - 16 bytes
0x5E0 - 32 bytes
0x600 - 128 bytes
0x680 - 64 bytes
0x6C0 - 16 bytes
-- Channel arbitration --
0x6D0 - 16 bytes
0x6E0 - 32 bytes
0x700 - 128 bytes
0x780 - 64 bytes
0x7C0 - 16 bytes
```

In the example above, the bandwidth control limits the channel from ever reaching the maximum transaction size of 256 bytes. Software should align addresses or increase the available bandwidth for best performance.

23.4.1.2 Basic DMA mode transfer

This mode is primarily included for backward compatibility with existing DMA controllers which use a simple programming model. This is the default mode out of reset.

The different modes of operation under the basic mode are explained in the following sections.

23.4.1.2.1 Basic direct mode

In basic direct mode, the DMA controller does not read descriptors from memory, but instead uses the current parameters programmed in the DMA registers to start the DMA transfer.

Software is responsible for initializing SAR*n*, SATR*n*, DAR*n*, DATR*n*, and BCR*n* registers. The DMA transfer is started when MR*n*[CS] is set. Software is expected to program all the appropriate registers before setting MR*n*[CS] to a 1. The transfer is finished after all the bytes specified in the byte count register have been transferred or if an error condition occurs. The sequence of events to start and complete a transfer in basic direct mode is as follows:

1. Poll the channel state (see Table 23-692) to confirm that the specific DMA channel is idle.
2. Initialize SAR*n*, SATR*n*, DAR*n*, DATR*n* and BCR*n*.
3. Set the mode register channel transfer mode bit, MR*n*[CTM], to indicate direct mode. Other control parameters may also be initialized in the mode register.
4. Clear, then set the mode register channel start bit, MR*n*[CS], to start the DMA transfer.
5. SR*n*[CB] is set by the DMA controller to indicate the DMA transfer is in progress.
6. SR*n*[CB] is automatically cleared by the DMA controller after the transfer is finished, or if the transfer is aborted (MR*n*[CA] transitions from a 0 to 1), or if a transfer error occurs.
7. End of segment interrupt is generated if MR*n*[EOSIE] is set.

23.4.1.2.2 Basic, direct, single-write start mode

In basic direct single-write start mode, the DMA controller does not read descriptors from memory, but instead uses the current parameters programmed in the DMA registers to start the DMA transfer.

Software is responsible for initializing the SATR*n*, DATR*n*, and BCR*n* registers. Setting MR*n*[SRW] configures the DMA controller to begin the DMA transfer either when SAR*n* is written or when DAR*n* is written, determined by the state of MR*n*[CDSM_SWSM]. Writing to SAR*n* initiates the DMA transfer if MR*n*[CDSM_SWSM] is set. Writing to DAR*n* initiates the DMA transfer if MR*n*[CDSM_SWSM] is cleared. The DMA controller automatically sets the channel start bit, MR*n*[CS]. Software is expected to program all the appropriate registers before writing the source or destination address registers. The transfer is finished after all the bytes specified in the byte count register have been transferred or if an error condition occurs. The sequence of events to start and complete a transfer in single-write start basic direct mode is as follows:

1. Poll the channel state (see Table 23-692) to confirm that the specific DMA channel is idle.
2. Initialize the source attributes (SATR*n*), DATR*n*, and BCR*n* registers.
3. Set the mode register channel transfer mode bit, MR*n*[CTM], and the single-write start direct mode bit, MR*n*[SRW]. Other control parameters may also be initialized in the mode register. Set MR*n*[CDSM_SWSM] for transfers started using SAR*n*. Clear MR*n*[CDSM_SWSM] for transfers started using the DAR*n*.
4. A write to the source or destination address register starts the DMA transfer and automatically sets MR*n*[CS].
5. SR*n*[CB] is set by the DMA controller to indicate the DMA transfer is in progress.
6. SR*n*[CB] is automatically cleared by the DMA controller after the transfer is finished, if the transfer is aborted (MR*n*[CA] transitions from a 0 to 1), or if a transfer error occurs.
7. End of segment interrupt is generated if MR*n*[EOSIE] is set.

23.4.1.2.3 Basic chaining mode

In basic chaining mode, software must first build link descriptor segments in memory.

Then the current link descriptor address register must be initialized to point to the first descriptor in memory. The DMA controller loads descriptors from memory prior to a DMA transfer. The DMA controller begins the transfer according to the link descriptor information loaded for the segment. After the current segment is finished, the DMA controller reads the next link descriptor from memory and begins another DMA transfer.

The transfer is finished if the current link descriptor is the last one in memory or if an error condition occurs. The sequence of events to start and complete a transfer in chaining mode is as follows:

1. Build link descriptor segments in memory.
2. Poll the channel state (see Table 23-692) to confirm that the specific DMA channel is idle.
3. Initialize CLNDAR*n* to point to the first link descriptor in memory.
4. Clear the mode register channel transfer mode bit, MR*n*[CTM], as well as MR*n*[XFE], to indicate basic chaining mode. Other control parameters may also be initialized in the mode register.
5. Clear and then set the mode register channel start bit, MR*n*[CS], to start the DMA transfer.
6. SR*n*[CB] is set by the DMA controller to indicate the DMA transfer is in progress.
7. SR*n*[CB] is automatically cleared by the DMA controller after finishing the transfer of the last descriptor segment, if the transfer is aborted (MR*n*[CA] transitions from a 0 to 1), or if an error occurs during any of the transfers.

23.4.1.2.4 Basic chaining, single-write start mode

Basic chaining, single-write start mode allows a chain to be started by writing the current link descriptor address register (CLNDAR*n*).

(Note that ECLNDAR*n* must be written *first* so that the full 40-bit descriptor address is present when the chain starts.) Setting MR*n*[CDSM_SWSM] in the mode register causes MR*n*[CS] to be automatically set when the current link descriptor address register is written. The sequence of events to start and complete a chain using single-write start mode is as follows:

1. Set the mode register current descriptor start mode bit, MR*n*[CDSM_SWSM], and clear the extended features enable bit MR*n*[XFE]. Also, clear the channel transfer mode bit, MR*n*[CTM]. This initialization indicates basic chaining and single-write start mode. Also other control parameters may be initialized in the mode register.
2. Build link descriptor segments in memory.
3. Poll the channel state (see Table 23-692) to confirm that the specific DMA channel is idle.
4. Initialize CLNDAR*n* to point to the first descriptor segment in memory. This write automatically causes the DMA controller to begin the link descriptor fetch and set MR*n*[CS].
5. SR*n*[CB] is set by the DMA controller to indicate the DMA transfer is in progress.
6. SR*n*[CB] is automatically cleared by the DMA controller after finishing the transfer of the last descriptor segment, if the transfer is aborted (MR*n*[CA] transitions from a 0 to 1), or if an error occurs during any of the transfers.

23.4.1.3 Extended DMA mode transfer

The extended DMA mode also operates in chaining and direct mode.

It offers additional capability over the basic mode by supporting striding and a more flexible descriptor structure. This additional functionality also requires a new and more complex programming model. The extended DMA mode is activated by setting MR*n*[XFE].

23.4.1.3.1 Extended direct mode

Extended direct mode has the same functionality as basic direct mode with the addition of stride capabilities.

The bit settings are the same as in direct mode with the exception of the MR*n*[XFE] being set. Striding on the source address can be accomplished by setting SATR*n*[SSME] and setting the desired stride size and distance in SSR*n*. Striding on the destination address can be accomplished by setting DATR*n*[DSME] and setting the desired stride size and distance in DSR*n*.

23.4.1.3.2 Extended direct, single-write start mode

Extended direct, single-write start mode has the same functionality as the basic direct single-write start mode with the addition of stride capabilities.

The bit settings are also the same with the exception of MR*n*[XFE] being set. Striding on the source address can be accomplished by setting SATR*n*[SSME] and setting the desired stride size and distance in SSR*n*. Striding on the destination address can be accomplished by setting DATR*n*[DSME] and setting the desired stride size and distance in DSR*n*.

23.4.1.3.3 Extended chaining mode

In extended chaining mode, the software must first build list and link descriptor segments in memory.

Then CLSDAR*n* must be initialized to point to the first list descriptor in memory. The DMA controller loads list descriptors and link descriptors from memory prior to a DMA transfer. The DMA controller begins the transfer according to the link descriptor information loaded. Once the current link descriptor is finished, the DMA controller reads the next link descriptor from memory and begins another DMA transfer. If the current link descriptor is the last in the list, the DMA controller reads the next list

descriptor in memory. The transfer is finished if the current link descriptor is the last one in the last list in memory or if an error condition occurs. The sequence of events to start and complete a transfer in extended chaining mode is as follows:

1. Build link and list descriptor segments in memory.
2. Poll the channel state (see Table 23-692) to confirm that the specific DMA channel is idle.
3. Initialize CLSDAR*n* to point to the first list descriptor in memory.
4. Clear the mode register channel transfer mode bit, MR*n*[CTM], to indicate chaining mode. MR*n*[XFE] must be set to indicate extended DMA mode. Other control parameters may also be initialized in the mode register.
5. Clear and then set the mode register channel start bit, MR*n*[CS], to start the DMA transfer.
6. SR*n*[CB] is set by the DMA controller to indicate the DMA transfer is in progress.
7. SR*n*[CB] is automatically cleared by the DMA controller after finishing the transfer of the last descriptor segment, if the transfer is aborted (MR*n*[CA] transitions from a 0 to 1), or if an error occurs during any of the transfers.

23.4.1.3.4 Extended chaining, single-write start mode

In the extended mode, the single-write start feature allows a chain to be started by writing the current list descriptor pointer.

Setting MR*n*[CDSM_SWSM] causes MR*n*[CS] to be set automatically when CLSDAR*n* is written. (Note that ECLSDAR*n* must be written *first* so that the full 40-bit descriptor address is present when the chain starts.) The sequence of events to start and complete an extended chain using single-write start mode is as follows:

1. Set MR*n*[CDSM_SWSM] and MR*n*[XFE] and clear MR*n*[CTM] to indicate extended chaining and single-write start mode. Also other control parameters may be initialized in the mode register.
2. Build list and link descriptor segments in local memory.
3. Poll the channel state (see Table 23-692) to confirm that the specific DMA channel is idle.
4. Initialize the current list descriptor address register to point to the first list descriptor segment in memory. This write automatically causes the DMA controller to begin the list descriptor fetch and set MR*n*[CS].
5. SR*n*[CB] is set by the DMA controller to indicate the DMA transfer is in progress.
6. SR*n*[CB] is automatically cleared by the DMA controller after finishing the transfer of the last descriptor segment, or if the transfer is aborted (MR*n*[CA] transitions from a 0 to 1), or if an error occurs during any of the transfers.

23.4.1.4 External control mode transfer

An external control can be used to control DMA1 and DMA2 channels that support external control signals by setting MR*n*[EMS_EN].

The external control can direct the DMA channel in the following transfer modes:

- Basic direct
- Basic chaining
- Extended direct
- Extended chaining

The external control and the DMA controller use a well defined protocol to communicate. The external control can start or restart a paused DMA transfer. The DMA controller acknowledges a DMA transfer in progress and also indicates a transfer completion. Note that external control cannot cause a channel to enter a paused state.

The pause feature can be enabled by setting MR*n*[EMP_EN]. MR*n*[BWC] specifies how much data to allow a specific channel to transfer before entering a paused state by clearing MR*n*[CS]. Note however, that write data for a paused transfer may not have reached the target interface when so indicated. The channel can be restarted from a paused state by the asserted edge of DREQ_B as driven by an external master. In chaining modes, the channel does not pause for descriptor fetch transfer; it only pauses during the actual data transfer.

Note that when operating the DMA in chaining mode the register byte count field, BCR[BC], must be initialized to zero before enabling the pause feature. In chaining modes, the channel does not pause for descriptor fetch transfer.

The following signals are defined for the external control interface:

- DMA_DREQ_B-Asserting edge triggers a DMA transfer start or restart from a pause request. Sets MR*n*[CS]. (Note that negating DMA_DREQ_B does NOT clear MR*n*[CS].)
- DMA_DACK_B-Indicates a DMA transfer currently in progress. SR*n*[CB] is set.
- DMA_DDONE_B-Indicates the completion of the DMA controller's involvement in the transfer and the readiness to accept a new DMA command. SR*n*[CB] is clear. Note however, that write data may still be queued at the target interface or in the process of transfer on an external interface.

Detailed descriptions of the external control interface are in Table 23-3. The timing diagram of the external control interface is shown in this figure.

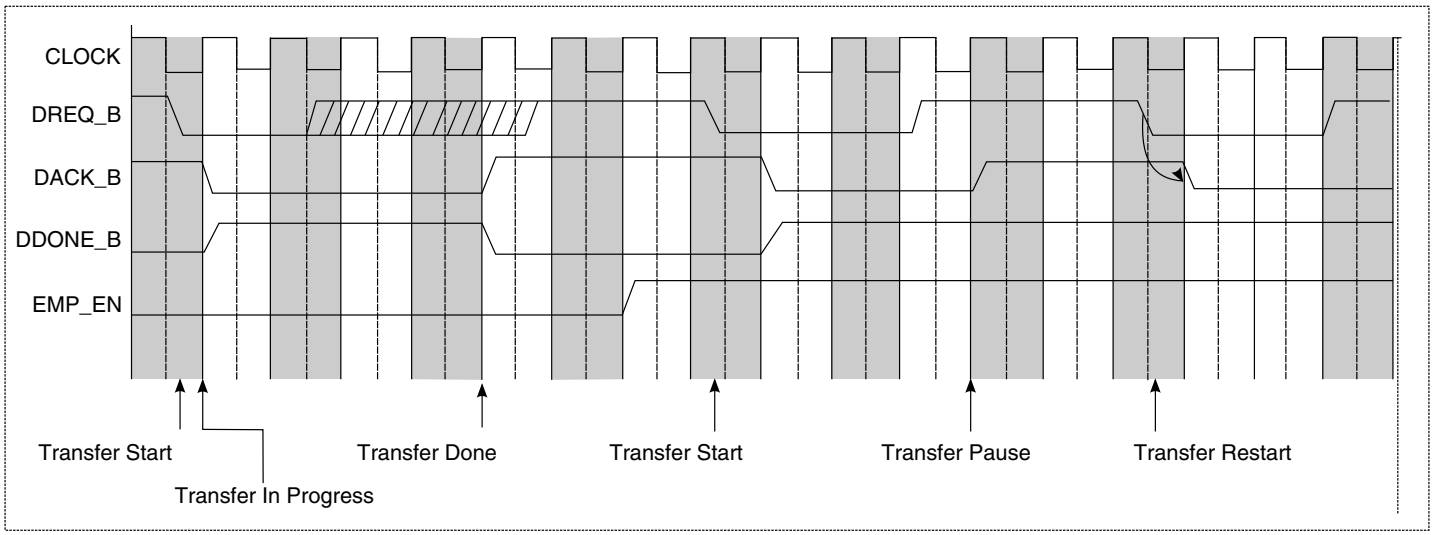


Figure 23-733. External control interface timing

23.4.1.5 Channel continue mode for cascading transfer chains

The channel continue mode (enabled when MR*n*[CC] is set) offers software the flexibility of having the DMA controller start on descriptors that have already been programmed while software continues to build more descriptors in memory.

Software can set the end-of-links descriptor (EOLND) in basic mode, or end-of-lists descriptor (EOLSD) in extended mode, to cause the channel to go into a halted state while software continues to build other descriptors in memory. Software can then set CC to force hardware to continue where it left off. Channel continue is only meaningful for chaining modes, not direct mode.

If CC is set by software while the channel is busy with a transfer, the DMA controller finishes all transfers until it reaches the EOLND in basic mode or EOLSD in extended mode. The DMA controller then refetches the last link descriptor in basic mode or the last list descriptor in extended mode and clears the channel continue bit. If EOLND or EOLSD is still set for their respective modes, the DMA controller remains in the idle state. If EOLND or EOLSD is not set, the DMA controller continues the transfer by refetching the new descriptor. The channel busy (SR*n*[CB]) bit is cleared when the DMA controller reaches EOLND/EOLSD and is set again when it initiates the refetch of the link or list descriptor.

If CC is set by software while the channel is not busy with a transfer, the DMA controller refetches the last link descriptor in basic mode, or the last list descriptor in extended mode and clears the channel continue bit. If EOLND or EOLSD is still set for their

respective modes, the DMA controller remains in the idle state. If the EOLND or EOLSD bits are not set, the DMA controller continues the transfer by refetching the new descriptor.

23.4.1.5.1 Basic mode

On a channel continue, the descriptor at the current link descriptor address register (CLNDAR*n*) is refetched to get the next link descriptor address field as updated by software.

The channel halts if NLNDAR*n*[EOLND] is still set. If EOLND is zero, the next link descriptor address is copied into CLNDAR*n* and the channel continues with another descriptor fetch of the current link descriptor address. As a result, two link descriptor fetches always exist after channel continue before starting the first transfer.

23.4.1.5.2 Extended mode

On a channel continue, the descriptor at the current list descriptor (CLSDAR*n*) address register is refetched to get the next list descriptor address field as updated by software.

The channel halts if NLSDAR*n*[EOLSD] is still set. If not, the next list descriptor address is copied into the CLSDAR*n* register and the channel continues with another descriptor fetch of the current list descriptor address. As a result, two list descriptor fetches always exist after channel continue before the first link descriptor fetch and the first transfer.

23.4.1.6 Channel abort

Software can abort a previously initiated transfer by setting MR*n*[CA].

Once the DMA channel controller detects a zero-to-one transition of MR*n*[CA], it finishes the current sub-block transfer and halts all further activity. The controller then waits for all previously initiated transfers from the specified channel to drain and clears SR*n*[CB]. Successful completion of a software initiated abort request can be recognized by MR*n*[CA] being set and SR*n*[CB] being cleared. Obviously, if the controller was already halted because of an error condition (SR*n*[TE] is set), or the channel has completed all transfers, then SR*n*[CB] being cleared may not signify that the controller entered a halt state due to the abort request.

23.4.1.7 Bandwidth control

MR*n*[BWC] specifies how much data to allow a specific channel to transfer before allowing the next channel to use the shared data transfer hardware.

This promotes equitable bandwidth allocation between channels. However, if only one channel is busy, hardware overrides the specified bandwidth control size value. The DMA controller allows a channel to transfer up to 1 Kbyte at a time when no other channel is active.

The maximum performance can be achieved on a single channel by disabling bandwidth control. This is recommended especially if only a single channel is utilized.

23.4.1.8 Channel state

This table defines the state of a channel based on the values of the channel start (MR*n*[CS]), channel busy (SR*n*[CB]), transfer error (SR*n*[TE]), and channel continue (MR*n*[CC]) bits.

Table 23-692. Channel state table

MRn[CS]	SRn[CB]	SRn[TE]	MRn[CC]	Channel state
0	0	0	0	Idle state. This is the state of the bits out of reset.
0	0	0	1	Channel continue unexpected. Channel remains idle
0	0	1	0	Error occurred after software halted the channel.
0	0	1	1	Channel continue unexpected. Channel remains in error halt state
0	1	0	0	Software halted channel. The channel was busy and software cleared MRn[CS].
0	1	0	1	Channel remains in halt state.
-	1	1	-	The channel has encountered an error condition and it is trying to halt.
1	0	0	0	Ready to start a transfer, or transfer completed
1	0	0	1	Continue transfer (only meaningful in chaining mode, not direct mode). In direct mode, the channel continue has no effect.
1	0	1	0	Error occurred during transfer
1	0	1	1	Channel remains in error halt state
1	1	0	0	Transfer in progress
1	1	0	1	Continue after reaching the end of list/link, or the first descriptor fetch after channel continue

23.4.1.9 Illustration of stride size and stride distance

If operating in stride mode, the stride size defines the amount of data to transfer before jumping to the next quantity of data as specified by the stride distance.

The stride distance is added to the current base address to point to the next quantity of data to be transferred. This figure illustrates the stride size and distance parameters.

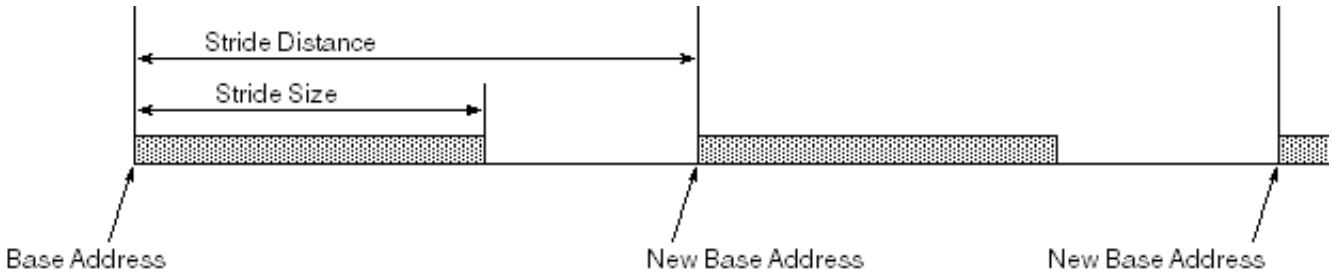


Figure 23-734. Stride size and stride distance

As shown, each time the stride distance is added to the base address, the resulting address becomes the new base address. This sequence repeats until the amount of data transferred equals the transfer size.

23.4.2 DMA transfer interfaces

The DMA can be used to achieve data transfers across the entire memory map.

23.4.3 DMA errors

On a transfer error (for example, non-correctable ECC errors on memory accesses, parity errors on flash controller, address mapping errors, and so on), the DMA halts by setting SR*n*[TE] and generates an interrupt if MR*n*[EIE] is set.

On a programming error, the DMA sets SR*n*[PE] and generates an interrupt if MR*n*[EIE] is set. The DMA controller detects the following programming errors:

- Transfer started with a byte count of zero
- Stride transfer started with a stride size of zero
- Illegal type, defined by SATR*n*[SREADTTYPE] and DATR*n*[DWRITETTYPE], used for the transfer.

23.4.4 DMA descriptors

The DMA engine recognizes the following types of descriptors:

- List descriptors connect lists of link descriptors.
- Link descriptors describe the DMA activity that is to take place.

DMA descriptors are built in either local or remote memory and are connected by the next descriptor fields. Only link descriptors contain information for the DMA controller to transfer data. Software must ensure that each descriptor is 32-byte aligned. The last link descriptor in the last list in memory sets the NLNDAR*n*[EOLND] bit in the next link descriptor and NLSDAR*n*[EOLSD] in the next list descriptor fields indicating that these are the last descriptors in memory. Software initializes the current list descriptor address register to point to the first list descriptor in memory. The DMA controller traverses through the descriptor lists until the last link descriptor is met. For each link descriptor in the chain, the DMA controller starts a new DMA transfer with the control parameters specified by that descriptor. Link and list descriptor fetches always snoop the local memory space.

NOTE

Software must ensure that each descriptor is aligned on a 32-byte boundary.

This table summarizes the DMA list descriptors.

Table 23-693. List DMA descriptor summary

Descriptor field	Description
Next list descriptor extended address	Points to the next list descriptor in memory. After the DMA controller reads the descriptor from memory, this field is loaded into the next list descriptor extended address registers.
Next list descriptor address	Points to the next list descriptor in memory. After the DMA controller reads the descriptor from memory, this field is loaded into the next list descriptor address registers.
First link descriptor extended address	Points to the first link descriptor in memory for this list. After the DMA controller reads the descriptor from memory, this field is loaded into the current link descriptor extended address registers.
First link descriptor address	Points to the first link descriptor in memory for this list. After the DMA controller reads the descriptor from memory, this field is loaded into the current link descriptor address registers.
Source stride	Contains the stride information used for the data source if striding is enabled for a link in the list
Destination stride	Contains the stride information used for the data destination if striding is enabled for a link in the list

This table summarizes the DMA link descriptors.

Table 23-694. Link DMA descriptor summary

Descriptor field	Description
Source attributes register	Contains source transaction attributes
Source address	Contains the source address of the DMA transfer. After the DMA controller reads the descriptor from memory, this field is loaded into the Source address register.
Destination attributes register	Contains destination transaction attributes
Destination address	Contains the destination address of the DMA transfer. After the DMA controller reads the descriptor from memory, this field is loaded into the destination address register.
Next link descriptor extended address	Points to the next link descriptor in memory. After the DMA controller reads the link descriptor from memory, this field is loaded into the extended next link descriptor address registers

Table continues on the next page...

Table 23-694. Link DMA descriptor summary (continued)

Descriptor field	Description
Next link descriptor address	Points to the next link descriptor in memory. After the DMA controller reads the link descriptor from memory, this field is loaded into the next link descriptor address registers.
Byte count	Contains the number of bytes to transfer. After the DMA controller reads the descriptor from memory, this field is loaded into the byte count register.

This figure describes the DMA transaction flow.

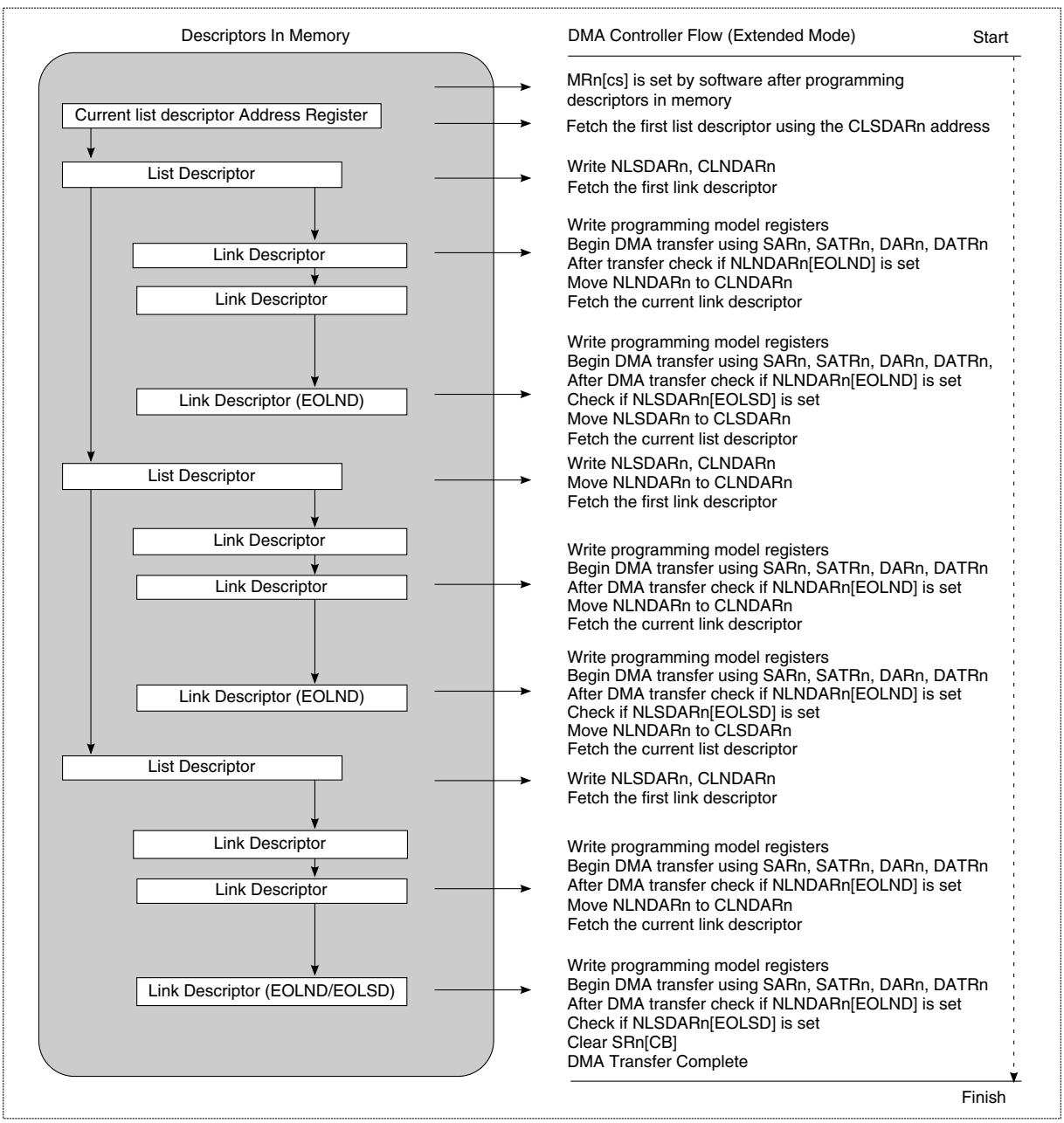


Figure 23-735. DMA transaction flow with DMA descriptors

This figure describes the format of the list descriptors.

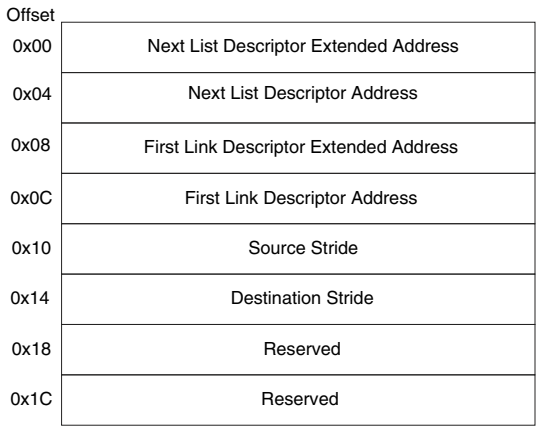


Figure 23-736. List descriptor format

This figure describes the format of the link descriptors.

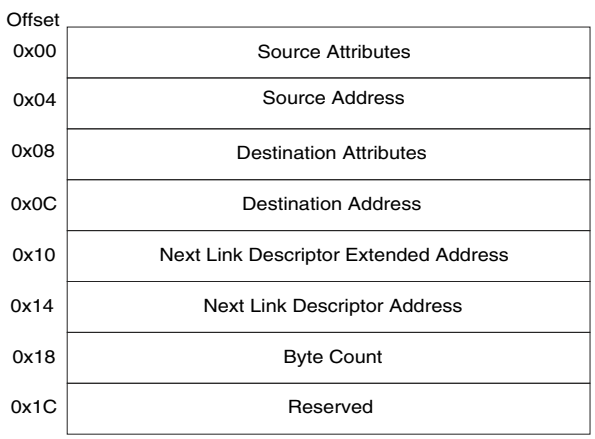


Figure 23-737. Link descriptor format

23.4.5 DMA controller limitations and restrictions

This section is intended to help software maximize the DMA performance and avoid DMA programming errors.

The limitations of the DMA controller are the following:

- Due to the limited number of buffers that the DMA controller can use, avoid stride sizes less than 64 bytes. Maximum utilization is obtained from strides greater than or equal to 256 bytes. However, small stride sizes can be used for scatter-gather functions.
- Coherent reads or writes are broken up into cache line accesses in the DMA.

The DMA controller restrictions are as follows:

- All interface capabilities from where descriptors are being fetched must support read sizes of 32 bytes or greater.
- If MR*n*[SAHE] is set, the source interface transfer size capability must be greater than or equal to MR*n*[SAHTS]. The source address must be aligned to a size specified by SAHTS.
- If MR*n*[DAHE] is set, the destination interface transfer size capability must be greater than or equal to MR*n*[DAHTS]. The destination address must be aligned to the size specified by DAHTS.
- Destination striding is not supported if MR*n*[DAHE] is set and source striding is not supported if MR*n*[SAHE] is set.

23.5 DMA system considerations

This section provides information about how to make most effective use of the DMA channels.

23.5.1 Unusual DMA scenarios

The following is a description of unusual DMA paths including explanations of why some functional modules cannot serve as DMA targets.

The following topics are addressed:

- DMA transaction initiators (masters)
- DMA targets, that is, data sources or destinations
- Transparency of the bus controllers to DMA transactions
- What is useful as opposed to what is possible. For example, any register can be addressed through an internal control bus, which means configuration and control registers can be DMA targets.

23.5.1.1 DMA to core

The L1 cache cannot be a direct DMA target because it cannot be directly addressed by software.

However, DMA access into the L1 cache occurs indirectly if a block of memory that is cached in the L1 is specified as the DMA target. This effect is deterministic if the target memory block was locked into the L1 with cache locking instructions.

23.5.1.2 DMA to configuration, control, and status registers (CCSR)

Because any internal register can be addressed with the four-channel DMA controller, configuration, control, and status registers throughout the device are valid DMA targets.

However, the primary purpose of DMA, to reduce processor load by moving large blocks of data, is not served by DMA transfers of configuration data. For example, while it is possible to DMA into the I^2C controller or programmable interrupt controller (PIC), doing so is extremely inefficient and is seldom beneficial in normal operation. The overhead of creating DMA descriptors far exceeds any savings in CPU cycles.

23.5.1.3 DMA to I^2C

The I^2C controller is not transparent to DMA transfers.

Observe the caveats listed in DMA to configuration, control, and status registers (CCSR) when accessing any I^2C register, including the data register (I2CDR).

23.5.1.4 DMA to DUART

The DUART provides complete and sophisticated DMA support, which is described in FIFO Mode.

23.5.1.5 DMA in multi-partition systems

The DMA controller uses a privileged copy of the LAWs to resolve source and destination targets.

This means that when a logical address is passed to the DMA controller, it compares this logical address with true physical addresses in the LAWs. Because of this behavior, the DMA controller has the capability to resolve its target without the need for a PAMU, providing optimization when targeting outbound I/O transactions. Because the PAMU only authorizes inbound transactions, it does not authorize transactions that resolve to outbound I/O. Note that this behavior only applies to outbound I/O transactions from the DMA controller; only the DMA controller has access to the privileged copy of the LAWs.

In a multi-partitioned system (such as a system partitioned and controlled under a Hypervisor), where logical addresses are not mapped one-to-one to physical addresses, one of the following three techniques must be implemented to help ensure absolute partitioning:

1. The programming of the DMA controller from a guest OS must be virtualized in software so that the guest OS must make Hypervisor calls (hcalls) in order to program the DMA. The Hypervisor is aware of the global system memory map and can block any OS from trying to target an address it deems invalid for that OS.
2. The Hypervisor must enforce the following logical address restrictions on all partitions:
   a. All I/O logical addresses must be mapped one-to-one using their true physical addresses.
   b. All guest OSs must map their I/O logical address space in the same globally unique system-wide range, which does not overlap other non-I/O addresses. All OSs use this space only for I/O.

   For example, the following represents a valid setup:

   - OS-A is aware of logical addresses A-C, that are mapped one-to-one with physical addresses A-C.
   - OS-B is aware of logical addresses D-F, that are mapped one-to-one with physical addresses D-F.
   - A law entry for DDR maps the entire SDRAM space to physical addresses A-F.
   - The PCI express controller 1 outbound window is mapped to physical addresses 2-3.

   In this example all logical addresses are mapped one-to-one with true physical addresses, and the outbound PCI express space does not overlap either OS's view of logical space.

   As a counter-example, the following represents an invalid setup:

   - OS-A is aware of logical addresses 0-2, that are mapped to physical addresses A-C.
   - OS-B is aware of logical addresses D-F, that are mapped one-to-one with physical addresses D-F.
   - A LAW entry for DDR maps the entire SDRAM space to physical addresses A-F.
   - The PCI express controller 1 outbound window is mapped from 1-2.

In this example, the setup is invalid because the logical addresses of OS-A overlap the outbound PCI express physical I/O window. If the DMA targets logical address 1, it will use its privileged copy of the LAWs to resolve the target, and because address 1 overlaps with the outbound PCI express window, it will target PCI express instead of DDR, which may not be intended.

3. Do not grant direct DMA controller access to a guest OS which cannot meet one of the previous two constraints.

Chapter 24
Data Path Acceleration Architecture (DPAA) Overview and T2080 DPAA Implementation

24.1 DPAA Introduction and Terms

The data path processing subsystem is an implementation of the QorIQ Data Path Acceleration Architecture (DPAA). The T2080 QorIQ Data Path Acceleration Architecture (DPAA) Appendices describe the superset of DPAA functionality. The chip implements a unique subset of this functionality, as described in the T2080-Specific DPAA Implementation Details.

As shown in the following figure, the DPAA consists of a parse, classify, and distribute module (PCD) and an accelerator module (FMan), along with related network interfaces (10GEC100Mbps/1G/2.5G/10G multirate ethernet MAC (mEMAC)), hardware offload accelerators (SEC, and PME), DCE, RMan and the infrastructure blocks that support these accelerators (QMan and BMan). Each of these accelerators provides its own unique set of functionality and thus each has its own unique requirements for how software must program and interact with that accelerator.

The DPAA's common infrastructure modules and the desire to be able to pass data received from one accelerator to another led to some common requirements, restrictions, and conventions. This chapter outlines requirements and conventions that are common to multiple accelerators within the DPAA.

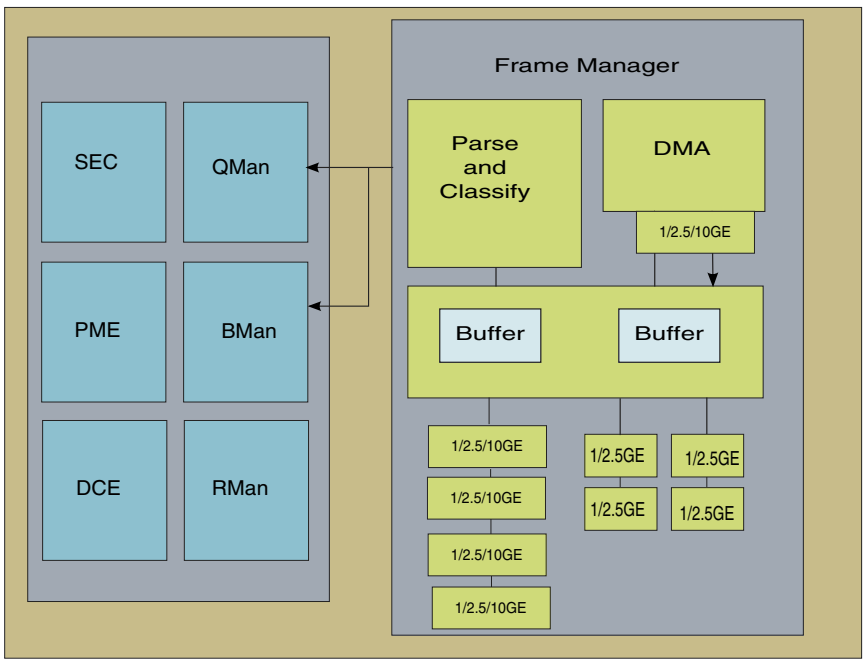


Figure 24-1. T2080 DPAA Components

The following list describes common DPAA terms and definitions.

Buffer

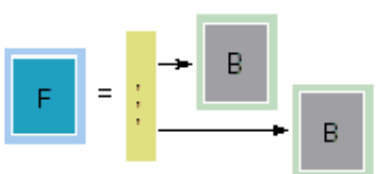
Region of contiguous memory, allocated by software, may be managed by the DPAA Buffer Manager (BMan).

Buffer pool

Set of buffers with common characteristics (mainly size, alignment, access control)

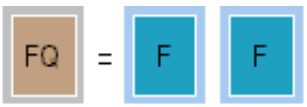
Frame

Contents of a single buffer or multiple buffers referenced by a table that hold data. For example, packet payload, header, and other control information.



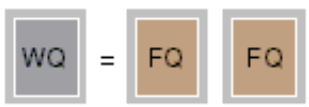
Frame queue (FQ)

FIFO of frames



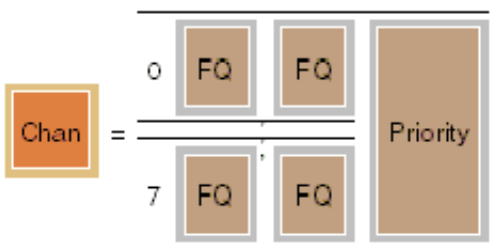
Work queue (WQ)

FIFO of frame queues (FQs)



Channel

Set of eight work queues (WQs), with hardware provided prioritized access



Dedicated channel

Channel statically assigned to a particular end point from which that end point can dequeue frames. End-point may be a CPU, FMan, PME, or SEC.

Pool channel

A channel statically assigned to a group of end points from which any of the end points may dequeue frames.

24.2 Data Formats Used in the DPAA

Data-to-be-processed is passed within the DPAA in distinct units known as “frames.” The actual data is held in buffers in memory and a description of the frame is what is passed within the DPAA.

24.2.1 Frame Descriptor (FD)

A frame descriptor (FD) is the basic element queued by the QMan. See Chapter 8, "Frame Manager" in the *QorIQ Data Path Acceleration Architecture (DPAA) Reference Manual* for a description of the QMan's usage of the FD. The use and interpretation of some fields is specific to the producer or consumer of the FD. Such usage is described in the chapter/section covering that module.

24.2.1.1 FD Format

Table 24-1. Frame Descriptor (FD)

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
DD		LIODN_OFFSET						BPID								ELIODN_OFFSET				Reserved				ADDR							
32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
ADDR																															
64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
FORMAT			OFFSET									LENGTH																			
			LENGTH or CONGESTION WEIGHT																												
96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
STATUS/CMD																															

Table 24-2. FD Field Descriptions

FD Bits	FD Name	FD Field Description
0-1	DD	Dynamic Debug marking code point A frame with a non-zero debug code point can generate a trace event at various enqueue and/or dequeue points within the QMan. Individual QMan users have different mechanisms and criteria for setting non-zero DD values.
2-7	LIODN_ OFFSET	Frame LIODN Offset (6 lsbs of complete LIODN_OFFSET) The LIODN_OFFSET is set when the FD is enqueued. It is used by a hardware module that dequeues the FD as part of the memory access control mechanism supported by the PAMU (see Role of the PAMU in the DPAA) See Accessing Memory Using Logical IO Device Numbers (LIODNs) for more information on the use of this field.
8-15	BPID	Buffer Pool ID The Buffer Pool ID is the number of the BMan buffer pool to which the buffer at ADDR should be released, if it is to be released to BMan.
16-19	ELIODN_ OFFSET	Frame Extended LIODN offset (4 msbs of complete LIODN_OFFSET) The complete 10-bit LIODN_OFFSET value is a concatenation of {ELIODN_OFFSET, LIODN_OFFSET}. See Accessing Memory Using Logical IO Device Numbers (LIODNs) for more information on the use of this field.

Table continues on the next page...

Table 24-2. FD Field Descriptions (continued)

FD Bits		FD Name	FD Field Description
20-23		-	Reserved
24-63		ADDR	Address The memory address of the start of the buffer holding the frame data or the buffer containing the scatter/gather list describing the frame
64-66		FORMAT	A coded value that defines the format of the OFFSET and LENGTH fields, and of the frame referenced by this FD. See Frame Format Codes for a description of these codes.
Optional fields	67-75	OFFSET	This field is valid only when FORMAT bits 65-66 are 00; otherwise, a frame offset value of 0 is implied. Frame offset is the number of bytes from the frame address (ADDR) at which actual data begins. Note that this field is not present in all FDs
	76-95	LENGTH	Total number of valid bytes of data in the frame. Note that this field is not present in all FDs.
	67-95	CONGESTION WEIGHT	A value used by the QMan for certain congestion management/avoidance calculations. Note that this field is not present in all FDs.
96-127		STATUS/CMD	Status or Command This field allows the sender of a frame to communicate some out-of-band information to the receiver of the frame. For example, this field can be used to communicate a command describing what processing is to be performed to a hardware accelerator or the error/output status after a hardware accelerator has finished processing a frame.

24.2.1.2 FD Considerations

To support the diverse needs of the various accelerators in the DPAA, multiple frame formats are defined. FD[FORMAT] designates the frame format described by the FD. Depending on the situation, one of the following must be stored:

- Data beginning at an offset from the actual start of the buffer, saving room at the beginning of the buffer (for adding new headers, for example)
- Large amount of data in contiguous memory

Because these cases tend to be mutually exclusive, formats are defined that trade off between the size of FD[LENGTH] and FD[OFFSET]. Depending on the FD[FORMAT], one of the following is present in the FD:

- FD[LENGTH] and FD[OFFSET]
- Just FD[LENGTH]
- Just FD[CONGESTION WEIGHT]

These fields occupy the same bits within the FD.

24.2.2 Multi-Buffer Frames

The actual data in a frame is held in one or more memory buffers. If multiple buffers are required, a scatter/gather table (also known as a block vector or IO vector table) is used to describe the buffers in a multi-buffer frame. See Scatter/Gather Entry Format for a description of this data structure.

24.2.2.1 Scatter/Gather Entry Format

Table 24-3. Scatter/Gather Table Entry Format

	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
0	Reserved (must be 0)																								ADDR							
	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
1	ADDR																															
	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
2	E	F	LENGTH																													
	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
3	Reserved (must be 0)								BPID								Reserved (Must be 0)			OFFSET												

Table 24-4. Scatter Gather Table Entry Field Description

Bits	Name	Description
0-23	-	Reserved
24-63	ADDR	Address of the buffer referenced by this table entry. This buffer may contain data or it may contain another scatter/gather table.
64	E	Extension bit. If set, this table entry points to a buffer that contains another scatter/gather table.
65	F	Final bit. If set, this is the last table entry for this frame. Note that the E (extension) bit takes precedence over the F (final) bit. If both are set, the F bit is ignored and processing continues with the entries in the scatter/gather table in referenced buffer.
66-95	LENGTH	Depending on the context, this field either specifies the number of valid bytes of data in the referenced buffer or the size of the referenced buffer (in the case that empty buffers are being provided to an accelerator). Note that if the E (extension) bit is set, LENGTH is ignored.
96-103	-	Reserved
104-111	BPID	Buffer Pool ID: The number of the BMan buffer pool to which this buffer should be released, if it is released to BMan.
112-114	-	Reserved
115-127	OFFSET	An offset in bytes from the ADDR at which valid data starts Note that if the E (extension) bit is set, "valid data" is a scatter/gather table.

24.2.2.2 Multi-Buffer Frame Considerations

Important multi-buffer frame considerations are as follows:

- The first scatter/gather table in a multi-buffer frame is held in the buffer referenced by the FD. In the case of a short, multi-buffer frame, the table starts at OFFSET bytes from the beginning of the buffer.
- “Trees” or hierarchy are not supported, because for simple, multi-buffer frames, processing never returns to the table in the original buffer.
- If the E bit is set, it indicates that the table entry points to another buffer containing more table entries. When a scatter/gather table entry with the E bit set is encountered, processing proceeds from the first entry in the new table found in the new buffer. This new table may begin at some offset from the start of the buffer as defined by the OFFSET field in the entry with the E bit set.
- In simple, multi-buffer frames, the LENGTH field in a table entry with its E bit set can be ignored.
- The E bit takes precedence over the F bit (that is, if both are set in an entry, the F bit is ignored).

This figure shows a simplified representation of a "long" simple frame using a scatter/gather table. Note that this diagram does not show the use of the Extension bit.

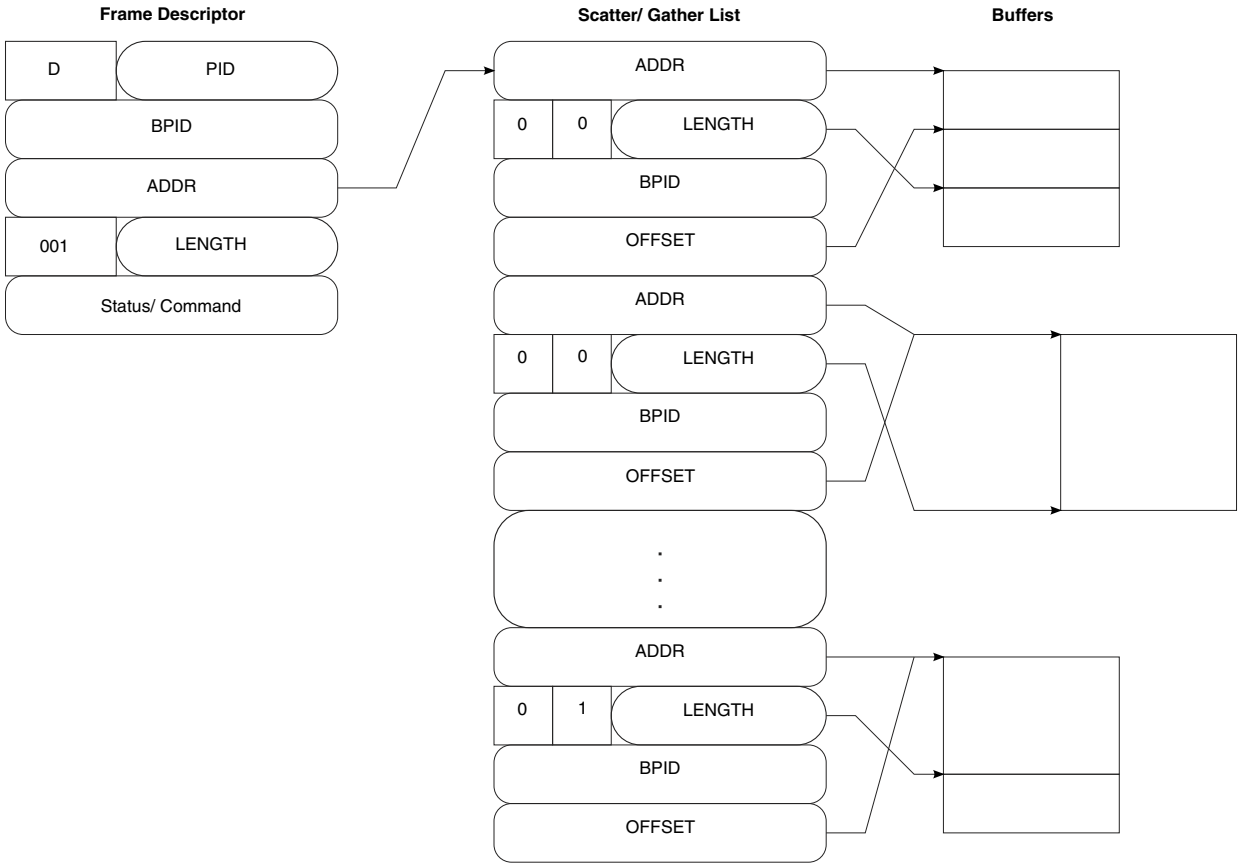


Figure 24-2. Simplified Representation of a Long, Multi-Buffer, Simple Frame

24.2.2.3 Situations Where Multi-Frame Buffering Stops

Multi-buffer frame processing stops in the following situations:

- When the data referenced by an entry with the F bit is processed regardless of the LENGTH indicated in the FD for the frame. In some cases, a mismatch (less data found in the frame compared to the FD length) may be an error condition for the accelerator module that is performing the processing, and it is reported as such.
- Processing also stops when the number of bytes specified by the overall length in the frame's FD have been processed. In this case, it is not necessary to have encountered an entry with the F bit set and it is not considered an error when this occurs.

24.2.3 Single-Buffer Frames

Most frame formats consist of a single delineated unit of data such as a single packet or Protocol Data Unit (PDU). FD[LENGTH] fields for these formats represent the total amount of valid data in the frame regardless of whether it is held in a single or in multiple buffers using scatter/gather. The OFFSET, on the other hand, represents the offset to the start of valid data or the scatter/gather table in the first buffer.

24.2.4 Compound Frames

Frames may also consist of multiple, related, distinct units such as the encrypted form of a packet along with the decrypted form of the same packet. These are known as compound frames, which consist of two or more simple frames. Each simple frame in a compound frame can in turn be stored in a single buffer or in multiple buffers. Compound frames exist so that all of the related data can be passed in a single unit within the DPAA. If an FD has a FORMAT code that identifies it as a compound frame then the FD[CONGESTION WEIGHT] is used to by the QMan for active queue management calculations such as WRED.

24.2.4.1 When to Use Compound Frames

Compound frames are used for the following purposes:

- To pass multiple, distinct pieces of data either to or from a hardware accelerator. For instance, it may be required to pass frames containing both encrypted and decrypted forms of a packet.
- To supply empty buffers to a hardware accelerator into which it may place its output. In this case, it is not desirable to use the BMan. Any LENGTH and OFFSET fields that refer to a specific buffer have special meaning; OFFSET specifies the byte offset from ADDR where the accelerator should start storing data and LENGTH gives the number of bytes of data which can be stored in the remainder of the buffer.

Consider the following when using compound frames:

- When multiple input or output frames are described by a compound frame, their order is consumer-dependent.
- If a compound frame is used to pass empty buffers to a consumer for its output, those buffers are in the first frame of the compound frame.
- If a module uses a compound frame to return the input frame as well as its output frame, the output frame is the first frame of the compound frame and the input frame is the second frame.

24.2.4.2 Compound Frame Considerations

Important compound frame considerations are as follows:

- Compound frames use the same scatter/gather table format as simple frames, but with slightly different semantics for the fields in the table entries.
- The buffer referenced by FD[ADDR] of a compound frame must contain a scatter/gather table (the compound scatter/gather table). A compound frame contains FD[CONGESTION WEIGHT] but does not contain FD[LENGTH] or FD[OFFSET]. The compound scatter/gather table must start at FD[ADDR].
- Each entry in the compound scatter/gather table references a simple frame. ADDR, LENGTH, BPID, and OFFSET fields in the compound scatter/gather table entry replace the corresponding fields in an FD. A compound frame contains FD[CONGESTION WEIGHT] but does not contain FD[LENGTH] or FD[OFFSET]. The compound scatter/gather table must start at FD[ADDR].
- The E bit is set if the simple frame occupies multiple buffers. The E bits in multiple entries in the compound scatter/gather table may be set, because each simple frame in the compound frame may occupy multiple buffers. In this case, the buffer at ADDR contains a scatter/gather table.
- The scatter/gather table(s) that make up a simple frame that is part of a compound frame follow the same semantics as described in Scatter/Gather Entry Format.
- There is no equivalent in the compound scatter/gather table for the FORMAT field in the FD. The entries in a compound scatter/gather table cannot themselves be compound frames; the E bit is used to indicate a multi-buffer frame, and the LENGTH and OFFSET fields supported in the scatter/gather entry are a superset of the fields in an FD.
- The F bit must be set in the last entry of a compound frame scatter/gather table.

This figure shows a representation of a compound frame.

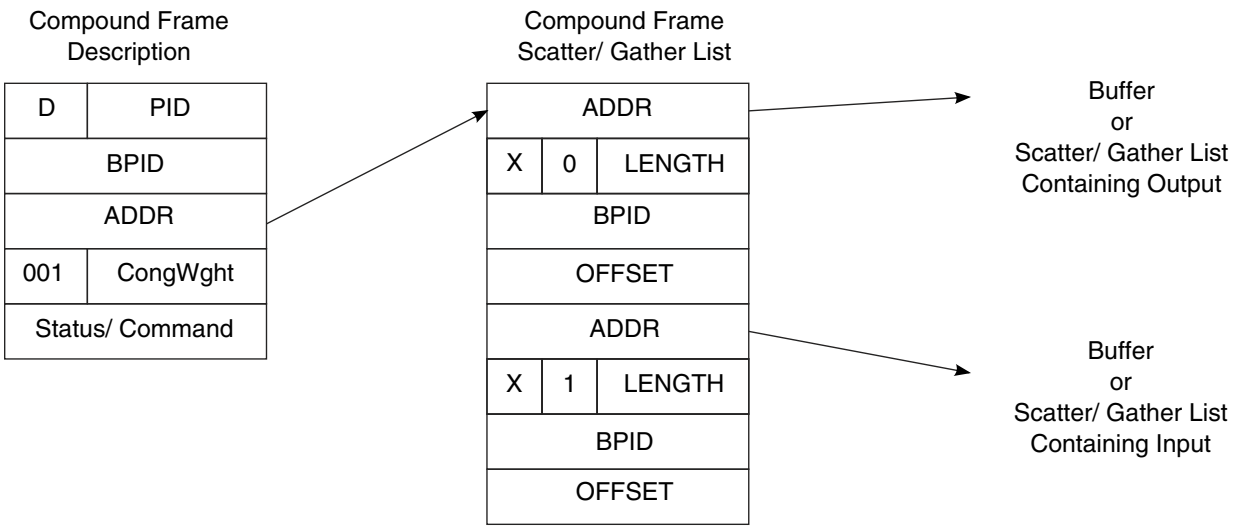


Figure 24-3. Simplified Representation of a Compound Frame

24.2.5 Simple Frames

Simple frames can be either short or long. Regardless of the length of the frame, data is stored in a single-buffer frame, whereas scatter/gather tables are stored in a multi-buffer frame.

A short frame can be no more than 1 Mbyte - 1 byte-long but the data (in the case of a single buffer frame) or scatter/gather table (in the case of a multi-buffer frame) can be stored starting at an offset from the beginning of the buffer referenced by the FD. Long frames can be as much as 512 Mbytes - 1 byte-long, but the data or scatter/gather table must be stored starting at the beginning of the buffer referenced by the FD.

Because network data (packets) are typically much less than 64 Kbytes in size, the short format is useful for most network processing needs.

24.2.6 Frame Format Codes

This table defines the frame format codes and provides a brief description of the frame format.

Table 24-5. Frame Format Codes

Value	Mnemonic	Definition	Size of LENGTH, OFFSET, and CONGESTION WEIGHT Fields
'000'	Short single buffer simple frame	Simple frame Single buffer, Offset and "small" length	9b OFFSET, 20b LENGTH
'010'	Long single buffer simple frame	Simple frame, single buffer, "big" length	29b LENGTH No OFFSET
'100'	Short multi-buffer simple frame	Simple frame, Scatter Gather table, Offset and "small" length	9b OFFSET, 20b LENGTH
'110'	Long multi-buffer simple frame	Simple frame, Scatter Gather table, "big" length	29b LENGTH No OFFSET
'001'	Compound frame	Compound Frame	29b <CONGESTION WEIGHT> No LENGTH or OFFSET
'011'	NA	Reserved	Not defined
'101'	NA	Reserved	Not defined
'111'	NA	Reserved	Not defined

24.2.7 Frame Formats Supported by Accelerators

Hardware accelerators (FMan, PME, and SEC) support a subset of the DPAA frame formats. This table summarizes the frame formats supported by various accelerators.

Table 24-6. Frame Format Support Matrix

	Short, Single Buffer		Long, Single Buffer		Short Multi-Buffer		Long Multi-Buffer		Compound Frames	
	Input	Output	Input	Output	Input	Output	Input	Output	Input	Output
FM	Yes	Yes	No	No	16-entry table E bit not supported	16-entry table E bit not supported	No	No	No	No
PME	Yes	Return input Frame only	Yes	Return input frame only	Yes	Return input frame only	Yes	Return input frame only	Yes	Yes
SEC	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
DCE	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
RMan	Yes	Yes	No	No	Yes	Yes	No	No	No	No

24.2.8 Special Values and Exceptions

The ADDR, BPID, and LENGTH fields can have special meanings under certain conditions, as follows:

- FDs
- Compound frame scatter/gather tables
- Multi-buffer scatter/gather tables

Exceptions and special behaviors are as follows:

- When the LENGTH field that describes a specific buffer is zero, this indicates that the buffer contains no data. If an accelerator finds a LENGTH of zero it should not access (read or write) the memory at ADDR.
- The LENGTH field in a scatter/gather table entry of a simple multi-buffer frame with its E (extension) bit set is an exception, because the value of this field is ignored.
- If an ADDR field that describes a specific buffer is zero, it indicates that there is no buffer. Regardless of the value of LENGTH, the “buffer” at ADDR==0 should not be accessed (read or written). Note that this means that buffers starting at address 0 are not valid buffers and this memory should not be used for buffers.

24.2.9 Releasing Buffers to the BMan

Accelerators only release buffers to the BMan that they have processed or, in the case of a buffer with a zero LENGTH, that they have passed over during processing. The exact configuration is dependent on the specific accelerator:

- Buffers with a non-zero ADDR are released.
- Buffers with a zero ADDR, BPID, and LENGTH are not released.

REQUIREMENT: Because processing stops when the frame’s total length of data is processed or after the data referenced by a scatter/gather entry with its F (final) bit set is processed, ensure that all buffers in a frame are released. For example, software must ensure that there are no entries in a scatter/gather table for a multi-buffer frame that are beyond the frame’s total length or described in entries after the entry with the F bit set.

24.3 Accessing Memory Using Logical IO Device Numbers (LIODNs)

This topic describes use of LIODNs and peripheral access management units (PAMUs).

24.3.1 Role of the PAMU in the DPAA

QorIQ multicore SoCs include peripheral access management units (PAMUs), which provide a number of functions including memory access control. When the PAMU is enabled, any hardware module in the SoC other than the cores access memory through PAMU. The PAMU can authorize the access, remap the memory address, and remap the transaction type (for example, a simple "write" to a "write with allocate to cache").

The PAMU determines what action to take and whether to authorize the action on the basis of the memory address, a logical IO device number (LIODN), and a configured table (logically) indexed by a LIODN and address. Hardware devices that need to access memory must provide a LIODN in addition to the memory address.

24.3.2 Using Multiple LIODNs

This topic describes the benefits and requirements of using multiple LIODNs.

24.3.2.1 Benefit of Using Multiple LIODNs

By using different LIODNs, a single hardware module can perform memory accesses on behalf of different requestors with the mapping and access controls appropriate to that requestor.

24.3.2.2 LIODN Requirements

Some important LIODN requirements are as follows:

- The requestor must communicate the LIODN to the hardware blockmodule. In the DPAA, this is done using the “<LIODN OFFSET>”. <LIODN OFFSET> is formed by concatenating LIODN_OFFSET and ELIODN_OFFSET from the FD with the latter used as the 4 msbs.
- Hardware module determine the LIODN to use in accessing memory by adding <LIODN OFFSET> to a configured <LIODN BASE>. Because <LIODN OFFSET> is in the FD, the module can use a different LIODN for each frame.
- Some hardware module may make different types of memory accesses as a result of dequeuing a frame. For instance, they may access per queue context/state/descriptors as well as reading and writing frame data. These modules may have different <LIODN BASE> values for these different types of access. These modules compute

different LIODNs using the <LIODN OFFSET> from the FD and the appropriate <LIODN BASE> configured for the type of access.

- The <LIODN OFFSET> must be set by the producer (enqueuer) of the FD. For software portals, this is done by the QMan from values configured for the portal to ensure that accesses by hardware module on behalf of software are controlled. In other words, software running on a core cannot get a hardware blockmodule to make accesses to memory, because it is not permitted to make these accesses by setting the <LIODN OFFSET> value in an FD.

24.4 Packet Walk-Through Example

The following example walks a packet through a DPAA-enabled, QorIQ device to illustrate how the DPAA offloads and accelerates packet forwarding while leaving key decisions under software control:

1. Datapath processing begins when Ethernet frames arrive at a network interface, assumed to be one of the Ethernet MACs (and 10GECmEMAC) within a Frame Manager (FMan). Alternatively, packets can arrive across a peripheral bus from an external network interface, in which case, the same packet walk-through stages can be applied with the help of a CPU acting as an I/O processor.
2. After FMan receives the Ethernet frame, it requests one or more buffers from the hardware Buffer Manager (BMan) to store the frame. BMan maintains pools of buffers, each with software-defined characteristics, and FMan is initialized to request a buffer from the most appropriate pool. If a sufficiently large buffer cannot be found for the incoming frame, FMan stores the frame across several smaller buffers and create a scatter/gather list for these buffers.
3. FMan's configurable parsing and filing capabilities can perform initial classification, sufficient to steer a packet toward a control processor, or toward one of the datapath processors for flow-specific processing (or additional classification by means of software). The steering of a packet towards a processor or a group of processors could be also based on differentiating flows by means of the Quality of Service attributes of the flow (for example, DSCP, IP precedence, or by user-defined proprietary headers).
4. Steering is accomplished by FMan issuing an enqueue command to QMan with the designated Frame Queue ID, along with frame parameters such as Frame Queue ID and Color Marking. In this example FMan's initial classification causes it to select a Frame Queue ID which the Queue Manager uses to steer the Frame Queue to the dedicated channel of logical CPU#3, a CPU operating in a datapath role. Potentially, the Frame Queue could be selected based on the QoS requirements of the flow with a policing profile associated with the selection.

5. Continuing the example, QMan places the Frame Queue onto Work Queue#5 of CPU#3's dedicated channel. The Frame Queue was queued to CPU#3 due to user configuration decisions that all packets belonging to this specific flow should be processed by CPU#3, possibly to take advantage of special processing instructions locked in CPU#3's private cache, or due to user-defined load balancing. The Frame Queue was placed in Work Queue#5 for QoS reasons, as the amount of traffic processed from each Work Queue relative to other work queues is also user-configurable. All Frame Queues on a Work Queue have equal priority, but the amount of traffic that the CPU draws from each is user-configurable to allow fairness and appropriate bandwidth allocation.
6. Once configured, QMan can take care of the packet/frame level scheduling requirements of the CPU, that is, QMan would appropriately schedule the Work Queue and Frame Queue within the Work Queue. QMan can be configured to perform a stashing operation in response to a CPU (or co-processor) accessing a Frame Queue.
7. CPU#3 performs protocol processing on the packet, including modification of the packet data in the buffer. Any modifications which change the length of the packet would be noted by means of changes to the frame. The CPU determines that the packet requires IPsec ESP processing, and enqueues the frame back to the QMan using the FQ ID associated with the packet's specific ESP tunnel.
8. The QMan uses this Frame Queue ID to determine that the next consumer of the Frame Queue is the SEC, and places the Frame Queue onto Work Queue#5 of the SEC's dedicated channel. The SEC pulls Frame Queues from the Work Queues in its dedicated channel according to user-defined weights in a WFQ model.
9. The SEC dequeues and processes data from the Frame Queue, adding the tunnel IP header, ESP header, IV, trailer, and HMAC, and writing encrypted data to either the original buffers, or new buffers which the SEC requests from BMan. The SEC updates the frame description (length change due to the addition of the headers, trailers, and HMAC) and enqueues the FQ back to QMan using a configured FQ ID. In this case, the FQ ID causes the QMan to enqueue the FQ to Work Queue #5 of logical CPU#4's dedicated channel. CPU#4 dequeues the FQ, determines the outbound interface for the newly encrypted packet, updates the tunnel IP header, and enqueues the frame back to QMan on a new FQ ID. This FQ ID causes the QMan to enqueue the FQ to a channel serviced by FMan, which dequeues the FQ, transmits one or more packets from that FQ out the appropriate mEMAC, and releases the buffers back to BMan.

The processing pipeline used in this example is not required by the QorIQ DPAA. The initial classification could have caused the packet to be steered toward a CPU dedicated to fine-grained classification, or to a pool channel of CPUs, any of which could have performed the operations described. CPU#3 could have added the ESP header and trailer to the packet and sent it to the SEC for crypto-only processing. Following SEC

processing, the flow was steered to CPU#4, however it could have just as easily been steered back to CPU#3, or to a pool channel. At any FQ ID transition, the relative priority of the flow could have been elevated or reduced by enqueuing it to a different work queue.

24.5 T2080-Specific DPAA Implementation Details

The *QorIQ Data Path Acceleration Architecture (DPAA) Reference Manual* describes the superset of DPAA functionality. The T2080 implements a unique subset of this functionality, as described in the following sections.

24.5.1 T2080 Queue Manager (QMan) Implementation

The QMan of the T2080 is implemented as described in the *QorIQ Data Path Acceleration Architecture (DPAA) Reference Manual* with the following implementation parameters:

- QMan block base address: 0x31_8000 - 0x31_9FFF
- 2-Kbyte frame queue (FQ) cache
- 2-Kbyte SFDRs
- 256 congestion groups

24.5.2 T2080 Buffer Manager (BMan) Implementation

The BMan of the T2080 is implemented as described in the *QorIQ Data Path Acceleration Architecture (DPAA) Reference Manual* with the following implementation parameters:

- BMan block base address: 0x31_A000 - 0x31_AFFF
- 64 buffer pools

24.5.3 T2080 Frame Manager (FMan) Implementation

The FMan of the T2080 is implemented as described in the *QorIQ Data Path Acceleration Architecture (DPAA) Reference Manual* with the following implementation parameters:

- FMan block base address: 0x40_0000 - 0x4F_FFFF

- Up to eight Ethernet MACs
  - Up to four 1 Gbps Ethernet MACs. Block base addresses are as follows:
    - MAC3: 4E_4000h
    - MAC4: 4E_6000h
    - MAC5: 4E_8000h
    - MAC6: 4E_A000h
  - Up to four 1/2.5/10 Gbps Ethernet MACs. Block base addresses are as follows:
    - MAC1: 4E_0000h (best effort)
    - MAC2: 4E_2000h (best effort)
    - MAC9: 4F_0000h
    - MAC10: 4F_2000h
- Two dedicated external Ethernet management interfaces (EMI1 and EMI2). Block base addresses are as follows:
  - MDIO1: 4F_C000h
  - MDIO2: 4F_D000h
- Up to eight SGMII interfaces and two parallel interfaces. See signals description and configuration chapter for valid interface combinations and corresponding signalling.
- Six Offline/Host Command ports (PortIDs: 02h...07h)
- 384-Kbyte internal FMan memory
- 64-Kbyte FMan Controller configuration data

24.5.4 T2080 Security and Encryption Engine (SEC) Implementation

The Security and Encryption Engine of the T2080 is implemented as described in the *T2080 SEC Reference Manual* with the following implementation parameters:

- SEC block base address: 0x30_0000 - 0x30_FFFF

Chapter 25
Multicore Programmable Interrupt Controller (MPIC)

This chapter describes how the multicore programmable interrupt controller (MPIC) manages the various types of interrupt sources managed by or originating from within the MPIC. It describes how to configure the MPIC to manage arbitration of those interrupt sources to ensure that the highest-priority interrupts are handled as quickly as possible.

NOTE

In this chapter, the mapping of the referenced MPIC core *n* to the T2080 e6500 core threads is as follows:

Table 25-1. T2080 MPIC Core *n* Mapping

MPIC Core *n*	T2080 Core/Thread
0	e6500 core 0, thread 0
1	e6500 core 0, thread 1
2	e6500 core 1, thread 0
3	e6500 core 1, thread 1
4	e6500 core 2, thread 0
5	e6500 core 2, thread 1
6	e6500 core 3, thread 0
7	e6500 core 3, thread 1

References to other MPIC cores should be disregarded.

25.1 Introduction

The MPIC conforms to the OpenPIC architecture, but with many enhancements. The interrupt controller provides multiprocessor interrupt management, and is responsible for receiving hardware-generated interrupts from different sources (both internal and external), prioritizing them in the context of interrupts that are generated from within the MPIC (such as messaging, timer, and interprocessor interrupts), and delivering them to the appropriate destination for servicing.

25.1.1 Features

The MPIC supports the following interrupt sources:

- External—Off-chip signals, IRQ[0:11]

- Internal—Up to 256 internal interrupt sources. On-chip sources from peripheral logic within the integrated device, signalling error conditions that need to be addressed by software. It is implementation specific which of these sources are used and how they are assigned; refer to the Interrupt Assignments chapter for specific assignments for this chip.
- Shared message signaled registers—Defined within the MPIC, used for cross-program communication, triggered on a register write, and cleared on a register read.
  - Four interprocessor interrupt channels.
  - Two groups of eight 32-bit message interrupt channels.
  - Four blocks of eight or sixteen shared message signaled interrupt sources with up to 32 sharers per shared interrupt register with interrupt coalescing intended to support up to four PCI Express host ports on a device.
  - Two groups of four global 32-bit timers clocked with the MPIC input clock or the RTC input. Timers within each group can be concatenated to time longer durations.
- Interrupts generated from within the MPIC itself, which are as follows:
  - Global timers A and B internal to the MPIC
  - Interprocessor interrupts (IPI)—Intended for communication between different processor cores on the same device. (Can be used for self-interrupt.)
  - Message registers—From within the MPIC. Triggered on register write, cleared on read. Used for interprocessor communication.

The MPIC has the following features:

- Three different modes of operation:
  - Supports the external proxy facility of the e6500 core
  - Legacy int*n* interrupt delivery to the processor core(s)
  - Pass-through mode (MPIC disabled) in which the MPIC directs interrupts off-chip for external servicing.
- Selected processor core(s) can be programmed to run in Legacy mode when MPIC is programmed to run in External proxy facility mode.
- The following types of programmable interrupt outputs:
  - The *int* outputs (*intn*). Any of the MPIC interrupt sources can be programmed to direct interrupt requests to *intn*. Handling of such interrupt requests follows the OpenPIC specification, which guarantees that the highest priority interrupts can supersede lower priority ones. Section 25.4.1.2, "Interrupts Routed to *int*," describes how the MPIC logic handles these interrupts.
  - The *cint* outputs (*cintn*). These are intended to be connected to the critical interrupt pin on the processor core(s).
  - The *mcp* outputs (*mcpn*). These are intended to be connected to the machine check pin on the processor core(s).
  - Interrupt output signal, $\overline{\text{IRQ\_OUT}}$. Section 25.4.1.1, "Interrupts Routed to cint, mcp, sie, or IRQ_OUT," describes how the MPIC logic supports these interrupts.
  - SoC Interrupt Event outputs (*sien*). These are used to signal events to the SoC.

- Programming model compatible with the OpenPIC architecture.
  - Message, interprocessor and global timer interrupts. (Note that the interprocessor and global timer interrupts can only be routed to *intn*.)
  - The following OpenPIC-defined features support only interrupts routed to the *int* signals:
    - Fully-nested interrupt delivery, guaranteeing that the interrupt source with the highest priority is given precedence over lower priority interrupts, including any that are in service.
    - 16 programmable interrupt priority levels
    - Support for identifying and handling spurious interrupts
- Support for up to 8 processors.
  - Interrupts can be routed to any processor core, depending on the configuration
  - Multicast delivery mode for interprocessor, global timer, and edge-triggered interrupts allowing these interrupts to be routed any or all of the cores
- Processor core initialization control
- Processor core reset control
- Processor non-maskable interrupt control
- Programmable resetting of the MPIC unit through the global configuration register
- Support for connection of external interrupt controller device such as an 8259 programmable interrupt controller. In 8259 mode, an interrupt causes assertion of a local (that is, internal to the integrated device) interrupt output signal $\overline{\text{IRQ\_OUT}}$.

25.1.2 Block Diagram

Figure 25-1 is a block diagram showing the relationship of the various functional blocks and how the signals external to the MPIC are connected to other blocks on the device, including the cores.

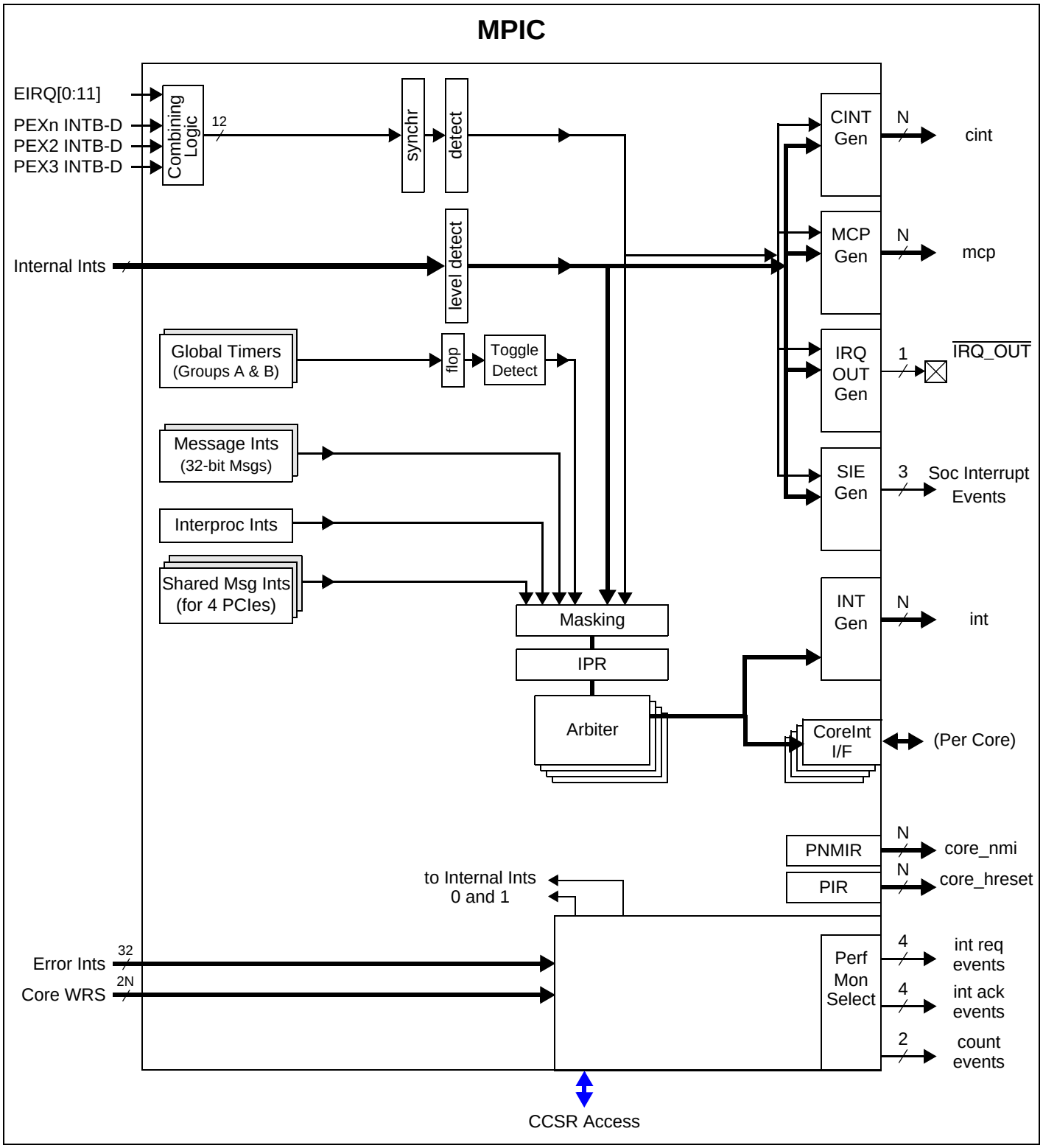


Figure 25-1. MPIC Block Diagram

25.1.3 The MPIC in Multiple- and Single-Core Implementations

In a multiprocessor system, it is possible for the MPIC to direct interrupts to the another processor. This functionality is supported by certain multiprocessor aspects to the programming model.

Note that while they are part of the programming model, such resources are reserved in a single-processor device.

25.1.4 Modes of Operation

The MPIC operates in one of the following modes:

- External Proxy Facility Mode
- Mixed Mode
- Pass-Through Mode

These are described below.

25.1.4.1 External Proxy Facility Mode (GCR[CI] = 1)

In external proxy facility mode, the vector is directly forwarded to the processor without the need for a software fetch. This feature is supported to all the interrupts. Detailed description of the external proxy facility is described in the *e6500 Core Reference Manual*.

NOTES

For processor core n to operate in external proxy facility mode, the GCR1[PIn] must be 0.

For MPIC to operate in external proxy facility mode, the GCR[M] must be 1. It is an invalid configuration to have MPIC in pass-through mode (GCR[M]=0) and in external proxy facility mode (GCR[CI]=1).

25.1.4.2 Mixed Mode (GCR[M] = 1)

In mixed mode, external and internal interrupts are delivered using the normal priority and delivery mechanisms detailed in Section 25.4.1, “Flow of Interrupt Control.”

25.1.4.3 Pass-Through Mode (GCR[M] = 0)

The MPIC provides a mechanism to support alternate external interrupt controllers such as the PC/AT-compatible 8259 interrupt controller architecture. After a hard reset, the MPIC defaults to pass-through mode, in which active-high interrupts from external source IRQ0 are passed directly to core 0 as shown in Figure 25-2; all other external interrupt signals are ignored. Thus, the interrupt signal from an external interrupt controller can be connected to IRQ0 and cause direct interrupts to the processor core 0. The MPIC does not perform a vector fetch from an 8259 interrupt controller.

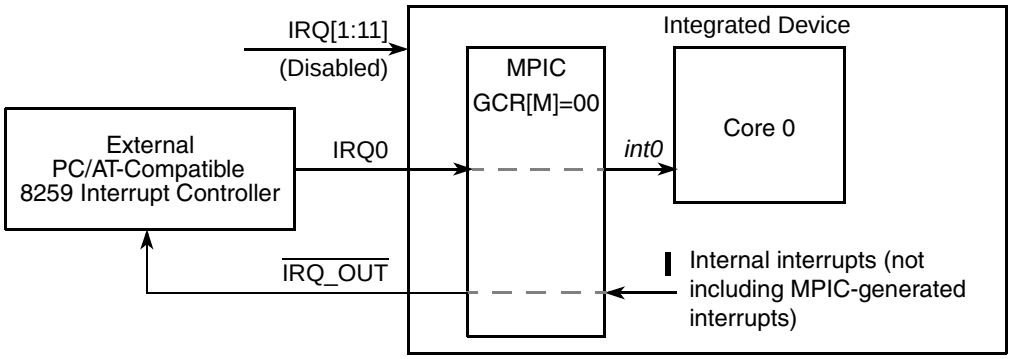


Figure 25-2. Pass-Through Mode Example

When pass-through mode is enabled, the internally-generated interrupts are not forwarded to core 0. Instead, the MPIC passes the raw interrupts from the internal sources to the interrupt output signal, $\overline{\text{IRQ\_OUT}}$.

NOTES

In pass-through mode, interrupts generated within the MPIC (global timers, interprocessor, and message register interrupts) are disabled. If internal or MPIC-generated interrupts must be reported internally to the processor, mixed mode must be used.

In pass-through mode, the target for each interrupt must be set to *int* (xILRn[INTTGT] = 0x00). This is the default setting of each register; therefore, no programming should be necessary to comply with this requirement.

25.1.5 Interrupts to the Processor Core

The *intn* signals, which signal an interrupt to the respective cores, are the main interrupt outputs from the MPIC unit to the processor cores. The global utilities block monitors the *intn* signals and wakes the corresponding core when one of these asserts and the core is in a low-power state (see Figure 25-1).

The interrupt sources can also specify the critical interrupt, mcp, or external interrupt outputs if the corresponding destination bits are set. For internal and external interrupts, the interrupt destination is decoded based on the Interrupt Level Registers (see Figure 25-75 and Figure 25-78) and xIDR*n*[P*n*]. These interrupt sources must be level sensitive.

The MPIC also defines the PIR, described in Section 25.3.1.7, "Processor Core Initialization Register (PIR)," which can be used to assert *hresetn* and trigger a hard reset. It also defines the PNMIR, described in Section 25.3.1.8, "Processor NMI Register (PNMIR)," which can be used to assert *coren_nmi* and to trigger a non-maskable interrupt. Processor core interrupts generated by the MPIC are described in Table 25-2.

Table 25-2. Processor Core Interrupts Generated by the MPIC—Types and Sources

Source	Internal, External, Message, Shared MSI, Global Timer, Interprocessor	Internal, External	Internal, External	Internal, External	PIR	PNMIR
MPIC output	*intn*	*cintn*	*mcpn*	$\overline{\text{IRQ\_OUT}}$, *sien*	*core0_hreset* through *core7_hreset*	*core0_nmi* through *core7_nmi*
Description	MPIC signals an interrupt in corresponding core.	Causes critical interrupt condition in correspondi ng core.	Causes machine check condition in correspondi ng core.	$\overline{\text{IRQ\_OUT}}$, *sien* assertion	One of the following: • External HRESET*n* assertion (and negation) • hreset*n* output from MPIC due to PIR update.	*core0_nmi* through *core7_nmi* outputs due to PNMIR update.

25.1.6 Interrupt Sources

The MPIC can receive separate interrupts from the following sources:

- External—Off-chip signals, IRQ[0:11]
- Internal—On-chip sources from peripheral logic within the integrated device. See the Interrupt Assignments chapter for a detailed list of all internal interrupt sources.
- Shared message signaled registers—Supports PCI Express Message Signalled Interrupts. Triggered on register write, cleared on read. Used for cross-program communication.
- Sources inside the MPIC:
  - Global timers A and B internal to the MPIC
  - Interprocessor interrupts (IPI)—Intended for communication between different processor cores on the same device. (Can be used for self-interrupt.)
  - Message registers—From within the MPIC. Triggered on register write, cleared on read. Used for interprocessor communication.
- Internal Error - internal errors from datapath etc. steered to processor interrupt, critical, or machine check of any core. It has a shared vector and destination (Internal Interrupt 0) supporting one bank for 32 error types.
- Watchdog timers—watchdog timer events that is steered to processor interrupt, critical, or machine check of any core. It has a shared vector and destination (Internal Interrupt 1).

25.1.6.1 Interrupt Destinations

When changed from Pass-Through mode to any of the other modes (external proxy or mixed mode), the MPIC directs all timer, shared message signaled, and interrupts from external and internal sources to the *int0* output (which is connected to the *int* signal of processor core 0). An interrupt can be routed to another core by explicitly setting the destination register for that source.

The interprocessor, global timers, and edge-triggered external interrupts can be programmed to be routed to one or more core's *int* signal. When more than one core is routed in destination, it is refered to as multi-casting. Level-sensitive external interrupts are not permitted for multi-casting.

Internal and external interrupts have more destination options, but only one destination can be chosen for a single non–multicasting interrupt. Instead of being routed to the *int* input of one of the processor cores, these interrupts can be alternatively routed to $\overline{\text{IRQ\_OUT}}$, *sie*, *mcp*, or *cint* with respect to the MPIC and routed to any core. These options are selected by writing to the *x*ILR*n*[INTTGT] field in the appropriate interrupt level register and xIDR*n*[P*n*] in the appropriate destination register.

Message and Shared MSI interrupts can only be routed to one of *intn* by writing to xIDR*n*[P*n*] fields in the appropriate destination register.

Table 25-3 summarizes the valid interrupt destinations for the various interrupt sources handled by the MPIC. It also indicates whether multicasting (MC) is supported or if only single destination (SD) is allowed for the interrupt. Single destination is used in the $\overline{\text{IRQ\_OUT}}$ interrupt output column because there is only one $\overline{\text{IRQ\_OUT}}$ signal—multicasting would not make sense for this selection.

Table 25-3. Destinations for MPIC Interrupt, NMI, and Reset Sources

Category	Interrupt Source	MPIC Interrupt Outputs[1] (Only one output type per source)					NMI & Core Reset		Debug[2]	Comments
		intn	*cintn*	*mcpn*	$\overline{\text{IRQ\_OUT}}$	*sien*	*coren_nmi*	*coren_hreset*	Perf-Mon	
Standard Interrupt Sources										
External Sources	EIRQ[0:m]	SD	MC[1]	MC[3],	SD[1]	SD[1]	—		SD	—
SoC Internal Sources[4]	IRQ[0:n]	SD	MC	MC	SD	SD			SD	—
Shared Message Signalled Interrupts	Initiated by Inbound PCIe write transactions	SD	—	—	—	—			SD	This set has no level registers.
MPIC Sources	Global Timers	MC	—	—	—	—			SD	This set has no level registers.
	Interprocessor Interrupts (IPI)	MC	—	—	—	—			SD	This set has no level registers.
	Message Interrupts	SD	—	—	—	—			SD	This set has no level registers.
	Shared Message Signalled Interrupts	SD	—	—	—	—			SD	This set has no level registers.
Register-Driven NMI and Core Resets										
MPIC Register Driven	PNMIR	—					MC	—	—	—
	PIR						—	MC		—

[1] Interrupt outputs are mutually exclusive and selected as follows:
intn: *x*ILR[INTTGT] = 0x00
cintn: *x*ILR[INTTGT] = 0x01
mcpn: *x*ILR[INTTGT] = 0x02
sien: *x*ILR[INTTGT] = 0xF0 through 0xF2
IRQ_OUT: *x*ILR[INTTGT] = 0xFF

[2] Perfmon events can operate in parallel with selected MPIC interrupt outputs.

[3] EIRQ source must be programmed as level-sensitive in EIVPRn[S] when used for this interrupt output. Note that all other sources are inherently level-sensitive.

[4] Includes the following: ORed SoC error interrupt as IRQ[0] and ORed WDT interrupt as IRQ[1].

For each interrupt source, only one MPIC interrupt output type can be used as programmed in its corresponding *x*ILR*n*[INTTGT]. For example, it is not possible to generate both an *intn* and IRQ_OUT in response to EIRQ[3] but it is possible to generate a *cint0* and *cint3* to cores 0 and 3 in response to EIRQ[3] since these share the same MPIC interrupt output type. Performance monitor events can be programmed to operate in parallel with a selected MPIC interrupt output.

25.1.6.2 Interrupt Routing—Mixed Mode

When an interrupt request is delivered to the MPIC, the corresponding interrupt destination register is checked to determine where the request should be routed, as follows:

- If Interrupt Level Register *x*ILR*n*[INTTGT] = 0xFF, the interrupt is routed off-chip to the interrupt output signal, IRQ_OUT.
- If Interrupt Level Register *x*ILRn[INTTGT] = 0xF0 to 0xF2, the interrupt is routed to the corresponding SoC interrupt event output signals, *sien*. See the Interrupt Assignments chapter for *sien* routing assignments.
- If Interrupt Level Register *x*ILR*n*[INTTGT] = 0x01, and one, but not multiple, *x*IDR*n*[P*n*] is set, the interrupt is routed to the appropriate *cintn*.
- If Interrupt Level Register *x*ILR*n*[INTTGT] = 0x02, and one, but not multiple, *x*IDR*n*[P*n*] is set, the interrupt is routed to the appropriate *mcpn*.
- If Interrupt Level Register *x*ILR*n*[INTTGT] = 0x00, and one, but not multiple, *x*IDR*n*[P*n*] is set, the interrupt is routed to the appropriate *intn* signal. In this case, the interrupt is latched by the interrupt pending register (IPR) and the interrupt flow is as described in Section 25.4.1, "Flow of Interrupt Control." Note that multicasting interrupts (global timer, interprocessor, and edge-triggered external interrupts) can set to any or all of P0 through P7; other interrupt sources cannot.

25.2 External Signal Descriptions

The following sections describe the MPIC signals.

25.2.1 Device External Pin Descriptions

The MPIC device external interface signals are described in Table 25-4. There are 12 distinct external interrupt request input signals and 1 interrupt request output pin IRQ_OUT.

Table 25-4 provides detailed descriptions of the external MPIC signals.

Table 25-4. External Interrupt Signals—Detailed Signal Descriptions

Signal	I/O	Description	
IRQ[0:11]	I	External Interrupt request 0–11. The polarity and sense of each of these signals are programmable. All of these inputs can be driven asynchronously. **Note:** Refer to the Interrupt Assignments chapter for information on how external interrupt request signals and PCI virtual INTx signals are shared.	
		State Meaning	Asserted—When an external interrupt signal is asserted (according to the programmed polarity), the MPIC checks its priority and the interrupt is conditionally passed to the processor designated in the corresponding destination register. In pass-through mode, only interrupts detected on IRQ0 are passed directly to core 0. Negated—There is no incoming interrupt from that source.
		Timing	Assertion—All of these inputs can be asserted asynchronously. Negation—Interrupts programmed as level-sensitive must remain asserted until serviced. Timing requirements for edge-sensitive interrupts can be found in the *Hardware Specifications*.
IRQ_OUT	O	Interrupt request out. Active-high out of the IP block. When the MPIC is programmed in pass-through mode, this output reflects the raw interrupts generated by on-chip sources. See Section 25.1.4, "Modes of Operation."	
		State Meaning	Asserted—At least one interrupt is currently being signalled to the external system. Negated—Indicates no interrupt source currently routed to $\overline{\text{IRQ\_OUT}}$.
		Timing	Because external interrupts are asynchronous with respect to the system clock, both assertion and negation of $\overline{\text{IRQ\_OUT}}$ occurs asynchronously with respect to the interrupt source. All timing given here is approximate. Assertion—Internal interrupt source: 2 clock cycles after interrupt occurs. External interrupt source: 4 cycles after interrupt occurs. Negation—Follows interrupt source negation with the following delay: Internal interrupt: 2 clock cycles External interrupt: 4 cycles.

25.3 Memory Map/Register Definition

The MPIC programmable register map occupies 256 Kbytes of memory-mapped space. Reading undefined portions of the memory map returns all zeros; writing has no effect. All MPIC registers are 32 bits wide and, although (mostly) located on 128-bit address boundaries, should be accessed only as 32-bit quantities.

The MPIC address offset map is divided into the following areas:

- 0x*nn*4_0020–0x*nn*4_0FFF—Global registers and Per-CPU private-access registers (Table 25-5)
- 0x*nn*5_0000–0x*nn*5_FFF0—Interrupt source configuration registers (Table 25-6)
- 0x*nn*6_0000–0x*nn*7_FFF0—Per-CPU global-access registers (Table 25-7)

The difference between Per-CPU private-access and global access registers is discussed in Section 25.3.8, "Per-CPU (Private Access) Registers.

Table 25-5. MPIC Register Address Map

Offset	Register	Access	Reset	Section/Page
Global Revision Registers				
0x4_0000	BRR1—Block revision register 1	R	imp-specific	25.3.1.1/25-42
0x4_0010	BRR2—Block revision register 2	R	imp-specific	25.3.1.2/25-43
Per-CPU Private-Access Registers				
0x4_0020–0x4_0030	Reserved	—	—	—
0x4_0040	IPIDR0—Interprocessor interrupt dispatch register 0	W	All zeros	25.3.8.1/25-93
0x4_0050	IPIDR1—Interprocessor interrupt dispatch register 1	W	All zeros	25.3.8.1/25-93
0x4_0060	IPIDR2—Interprocessor interrupt dispatch register 2	W	All zeros	25.3.8.1/25-93
0x4_0070	IPIDR3—Interprocessor interrupt dispatch register 3	W	All zeros	25.3.8.1/25-93
0x4_0080	CTPR—Current task priority register	R/W	0x0000_000F	25.3.8.2/25-95
0x4_0090	WHOAMI—Who am I register	R	n/a	25.3.8.3/25-96
0x4_00A0	IACK—Interrupt acknowledge register	R	All zeros	25.3.8.4/25-96
0x4_00B0	EOI—End of interrupt register	W	All zeros	25.3.8.5/25-97
0x4_00C0–0x4_0FF0	Reserved	—	—	—
Global Services Registers				
0x4_1000	FRR—Feature reporting register	R	imp-specific	25.3.1.3/25-43
0x4_1010	Reserved	—	—	—
0x4_1020	GCR—Global configuration register	R/W	All zeros	25.3.1.4/25-44
0x4_1024	GCR1—Global configuration register 1	R/W	All zeros	25.3.1.5/25-44
0x4_1030–0x4_1070	Reserved	—	—	—
0x4_1080	VIR—Vendor identification register	R	All zeros	25.3.1.5/25-44
0x4_1090	PIR—Processor initialization register	R/W	All zeros	25.3.1.7/25-45
0x4_1098	PNMIR—Processor NMI register	R/W	All zeros	25.3.1.8/25-46
0x4_10A0	IPIVPR0—IPI vector/priority register 0	R/W	0x8000_0000	25.3.1.9/25-47
0x4_10B0	IPIVPR1—IPI vector/priority register 1	R/W	0x8000_0000	25.3.1.9/25-47
0x4_10C0	IPIVPR2—IPI vector/priority register 2	R/W	0x8000_0000	25.3.1.9/25-47
0x4_10D0	IPIVPR3—IPI vector/priority register 3	R/W	0x8000_0000	25.3.1.9/25-47
0x4_10E0	SVR—Spurious vector register	R/W	0x0000_FFFF	25.3.1.10/25-48
Global Timer Group A Registers				
0x4_10F0	TFRRA—Timer frequency reporting register (Group A)	R/W	All zeros	25.3.2.1/25-48

Table 25-5. MPIC Register Address Map

Offset	Register	Access	Reset	Section/Page
0x4_1100	GTCCRA0—Global timer current count register (Group A) 0	R	All zeros	25.3.2.2/25-49
0x4_1110	GTBCRA0—Global timer base count register (Group A) 0	R/W	0x8000_0000	25.3.2.3/25-49
0x4_1120	GTVPRA0—Global timer vector/priority register (Group A) 0	R/W	0x8000_0000	25.3.2.4/25-50
0x4_1130	GTDRA0—Global timer destination register (Group A) 0	R/W	0x0000_0001	25.3.2.5/25-51
0x4_1140	GTCCRA1—Global timer current count register (Group A) 1	R	All zeros	25.3.2.2/25-49
0x4_1150	GTBCRA1—Global timer base count register (Group A) 1	R/W	0x8000_0000	25.3.2.3/25-49
0x4_1160	GTVPRA1—Global timer vector/priority register (Group A) 1	R/W	0x8000_0000	25.3.2.4/25-50
0x4_1170	GTDRA1—Global timer destination register (Group A) 1	R/W	0x0000_0001	25.3.2.5/25-51
0x4_1180	GTCCRA2—Global timer current count register (Group A) 2	R	All zeros	25.3.2.2/25-49
0x4_1190	GTBCRA2—Global timer base count register (Group A) 2	R/W	0x8000_0000	25.3.2.3/25-49
0x4_11A0	GTVPRA2—Global timer vector/priority register (Group A) 2	R/W	0x8000_0000	25.3.2.4/25-50
0x4_11B0	GTDRA2—Global timer destination register (Group A) 2	R/W	0x0000_0001	25.3.2.5/25-51
0x4_11C0	GTCCRA3—Global timer current count register (Group A) 3	R	All zeros	25.3.2.2/25-49
0x4_11D0	GTBCRA3—Global timer base count register (Group A) 3	R/W	0x8000_0000	25.3.2.3/25-49
0x4_11E0	GTVPRA3—Global timer vector/priority register (Group A) 3	R/W	0x8000_0000	25.3.2.4/25-50
0x4_11F0	GTDRA3—Global timer destination register (Group A) 3	R/W	0x0000_0001	25.3.2.5/25-51
0x4_1200- 0x4_12F0	Reserved	—	—	—
0x4_1300	TCRA—Timer control register (Group A)	R/W	All zeros	25.3.2.6/25-52
0x4_1310– 0x4_13F0	Reserved	—	—	—
Global Message Group A Registers				
0x4_1400	MSGRA0—Message register (Group A) 0	R/W	All zeros	25.3.5.1/25-75
0x4_1410	MSGRA1—Message register (Group A) 1	R/W	All zeros	25.3.5.1/25-75
0x4_1420	MSGRA2—Message register (Group A) 2	R/W	All zeros	25.3.5.1/25-75
0x4_1430	MSGRA3—Message register (Group A) 3	R/W	All zeros	25.3.5.1/25-75
0x4_1440	MSGRA4—Message register (Group A) 4	R/W	All zeros	25.3.5.1/25-75
0x4_1450	MSGRA5—Message register (Group A) 5	R/W	All zeros	25.3.5.1/25-75
0x4_1460	MSGRA6—Message register (Group A) 6	R/W	All zeros	25.3.5.1/25-75
0x4_1470	MSGRA7—Message register (Group A) 7	R/W	All zeros	25.3.5.1/25-75
0x4_1480– 0x4_14F0	Reserved	—	—	—
0x4_1500	MERA—Message enable register (Group A)	R/W	All zeros	25.3.5.2/25-75
0x4_1510	MSRA—Message status register (Group A)	R/W	All zeros	25.3.5.3/25-76

Table 25-5. MPIC Register Address Map

Offset	Register	Access	Reset	Section/Page
0x4_1520-0x4_15F0	Reserved	—	—	—
Bank A Shared Message Signaled Interrupt Registers				
0x4_1600	MSIRA0—Shared message signaled interrupt register (Bank A) 0	RC	All zeros	25.3.6.1/25-77
0x4_1610	MSIRA1—Shared message signaled interrupt register (Bank A) 1	RC	All zeros	25.3.6.1/25-77
0x4_1620	MSIRA2—Shared message signaled interrupt register (Bank A) 2	RC	All zeros	25.3.6.1/25-77
0x4_1630	MSIRA3—Shared message signaled interrupt register (Bank A) 3	RC	All zeros	25.3.6.1/25-77
0x4_1640	MSIRA4—Shared message signaled interrupt register (Bank A) 4	RC	All zeros	25.3.6.1/25-77
0x4_1650	MSIRA5—Shared message signaled interrupt register (Bank A) 5	RC	All zeros	25.3.6.1/25-77
0x4_1660	MSIRA6—Shared message signaled interrupt register (Bank A) 6	RC	All zeros	25.3.6.1/25-77
0x4_1670	MSIRA7—Shared message signaled interrupt register (Bank A) 7	RC	All zeros	25.3.6.1/25-77
0x4_1680	MSIRA8—Shared message signaled interrupt register (Bank A) 8	RC	All zeros	25.3.6.1/25-77
0x4_1690	MSIRA9—Shared message signaled interrupt register (Bank A) 9	RC	All zeros	25.3.6.1/25-77
0x4_16A0	MSIRA10—Shared message signaled interrupt register (Bank A) 10	RC	All zeros	25.3.6.1/25-77
0x4_16B0	MSIRA11—Shared message signaled interrupt register (Bank A) 11	RC	All zeros	25.3.6.1/25-77
0x4_16C0	MSIRA12—Shared message signaled interrupt register (Bank A) 12	RC	All zeros	25.3.6.1/25-77
0x4_16D0	MSIRA13—Shared message signaled interrupt register (Bank A) 13	RC	All zeros	25.3.6.1/25-77
0x4_16E0	MSIRA14—Shared message signaled interrupt register (Bank A) 14	RC	All zeros	25.3.6.1/25-77
0x4_16F0	MSIRA15—Shared message signaled interrupt register (Bank A) 15	RC	All zeros	25.3.6.1/25-77
0x4_1720	MSISRA—Shared message signaled interrupt status register (Bank A)	R	All zeros	25.3.6.2/25-78
0x4_1740	MSIIRA—Shared message signaled interrupt index register (Bank A)	W	All zeros	25.3.6.3/25-78
0x4_1750–0x4_17F0	Reserved	—	—	—
Bank B Shared Message Signaled Interrupt Registers				
0x4_1800	MSIRB0—Shared message signaled interrupt register (Bank B) 0	RC	All zeros	25.3.6.1/25-77
0x4_1810	MSIRB1—Shared message signaled interrupt register (Bank B) 1	RC	All zeros	25.3.6.1/25-77
0x4_1820	MSIRB2—Shared message signaled interrupt register (Bank B) 2	RC	All zeros	25.3.6.1/25-77
0x4_1830	MSIRB3—Shared message signaled interrupt register (Bank B) 3	RC	All zeros	25.3.6.1/25-77
0x4_1840	MSIRB4—Shared message signaled interrupt register (Bank B) 4	RC	All zeros	25.3.6.1/25-77
0x4_1850	MSIRB5—Shared message signaled interrupt register (Bank B) 5	RC	All zeros	25.3.6.1/25-77
0x4_1860	MSIRB6—Shared message signaled interrupt register (Bank B) 6	RC	All zeros	25.3.6.1/25-77
0x4_1870	MSIRB7—Shared message signaled interrupt register (Bank B) 7	RC	All zeros	25.3.6.1/25-77

Table 25-5. MPIC Register Address Map

Offset	Register	Access	Reset	Section/Page
0x4_1880	MSIRB8—Shared message signaled interrupt register (Bank B) 8	RC	All zeros	25.3.6.1/25-77
0x4_1890	MSIRB9—Shared message signaled interrupt register (Bank B) 9	RC	All zeros	25.3.6.1/25-77
0x4_18A0	MSIRB10—Shared message signaled interrupt register (Bank B) 10	RC	All zeros	25.3.6.1/25-77
0x4_18B0	MSIRB11—Shared message signaled interrupt register (Bank B) 11	RC	All zeros	25.3.6.1/25-77
0x4_18C0	MSIRB12—Shared message signaled interrupt register (Bank B) 12	RC	All zeros	25.3.6.1/25-77
0x4_18D0	MSIRB13—Shared message signaled interrupt register (Bank B) 13	RC	All zeros	25.3.6.1/25-77
0x4_18E0	MSIRB14—Shared message signaled interrupt register (Bank B) 14	RC	All zeros	25.3.6.1/25-77
0x4_18F0	MSIRB15—Shared message signaled interrupt register (Bank B) 15	RC	All zeros	25.3.6.1/25-77
0x4_1920	MSISRB—Shared message signaled interrupt status register (Bank B)	R	All zeros	25.3.6.2/25-78
0x4_1940	MSIIRB—Shared message signaled interrupt index register (Bank B)	W	All zeros	25.3.6.3/25-78
0x4_1950–0x4_19F0	Reserved	—	—	—
Bank C Shared Message Signaled Interrupt Registers				
0x4_1A00	MSIRC0—Shared message signaled interrupt register (Bank C) 0	RC	All zeros	25.3.6.1/25-77
0x4_1A10	MSIRC1—Shared message signaled interrupt register (Bank C) 1	RC	All zeros	25.3.6.1/25-77
0x4_1A20	MSIRC2—Shared message signaled interrupt register (Bank C) 2	RC	All zeros	25.3.6.1/25-77
0x4_1A30	MSIRC3—Shared message signaled interrupt register (Bank C) 3	RC	All zeros	25.3.6.1/25-77
0x4_1A40	MSIRC4—Shared message signaled interrupt register (Bank C) 4	RC	All zeros	25.3.6.1/25-77
0x4_1A50	MSIRC5—Shared message signaled interrupt register (Bank C) 5	RC	All zeros	25.3.6.1/25-77
0x4_1A60	MSIRC6—Shared message signaled interrupt register (Bank C) 6	RC	All zeros	25.3.6.1/25-77
0x4_1A70	MSIRC7—Shared message signaled interrupt register (Bank C) 7	RC	All zeros	25.3.6.1/25-77
0x4_1A80	MSIRC8—Shared message signaled interrupt register (Bank C) 8	RC	All zeros	25.3.6.1/25-77
0x4_1A90	MSIRC9—Shared message signaled interrupt register (Bank C) 9	RC	All zeros	25.3.6.1/25-77
0x4_1AA0	MSIRC10—Shared message signaled interrupt register (Bank C) 10	RC	All zeros	25.3.6.1/25-77
0x4_1AB0	MSIRC11—Shared message signaled interrupt register (Bank C) 11	RC	All zeros	25.3.6.1/25-77
0x4_1AC0	MSIRC12—Shared message signaled interrupt register (Bank C) 12	RC	All zeros	25.3.6.1/25-77
0x4_1AD0	MSIRC13—Shared message signaled interrupt register (Bank C) 13	RC	All zeros	25.3.6.1/25-77
0x4_1AE0	MSIRC14—Shared message signaled interrupt register (Bank C) 14	RC	All zeros	25.3.6.1/25-77
0x4_1AF0	MSIRC15—Shared message signaled interrupt register (Bank C) 15	RC	All zeros	25.3.6.1/25-77
0x4_1B20	MSISRC—Shared message signaled interrupt status register (Bank C)	R	All zeros	25.3.6.2/25-78
0x4_1B40	MSIIRC—Shared message signaled interrupt index register (Bank C)	W	All zeros	25.3.6.3/25-78

Table 25-5. MPIC Register Address Map

Offset	Register	Access	Reset	Section/Page
0x4_1B50–0x4_1FF0	Reserved	—	—	—
Bank D Shared Message Signaled Interrupt Registers				
0x4_1C00	MSIRD0—Shared message signaled interrupt register (Bank D) 0	RC	All zeros	25.3.6.1/25-77
0x4_1C10	MSIRD1—Shared message signaled interrupt register (Bank D) 1	RC	All zeros	25.3.6.1/25-77
0x4_1C20	MSIRD2—Shared message signaled interrupt register (Bank D) 2	RC	All zeros	25.3.6.1/25-77
0x4_1C30	MSIRD3—Shared message signaled interrupt register (Bank D) 3	RC	All zeros	25.3.6.1/25-77
0x4_1C40	MSIRD4—Shared message signaled interrupt register (Bank D) 4	RC	All zeros	25.3.6.1/25-77
0x4_1C50	MSIRD5—Shared message signaled interrupt register (Bank D) 5	RC	All zeros	25.3.6.1/25-77
0x4_1C60	MSIRD6—Shared message signaled interrupt register (Bank D) 6	RC	All zeros	25.3.6.1/25-77
0x4_1C70	MSIRD7—Shared message signaled interrupt register (Bank D) 7	RC	All zeros	25.3.6.1/25-77
0x4_1C80	MSIRD8—Shared message signaled interrupt register (Bank D) 8	RC	All zeros	25.3.6.1/25-77
0x4_1C90	MSIRD9—Shared message signaled interrupt register (Bank D) 9	RC	All zeros	25.3.6.1/25-77
0x4_1CA0	MSIRD10—Shared message signaled interrupt register (Bank D) 10	RC	All zeros	25.3.6.1/25-77
0x4_1CB0	MSIRD11—Shared message signaled interrupt register (Bank D) 11	RC	All zeros	25.3.6.1/25-77
0x4_1CC0	MSIRD12—Shared message signaled interrupt register (Bank D) 12	RC	All zeros	25.3.6.1/25-77
0x4_1CD0	MSIRD13—Shared message signaled interrupt register (Bank D) 13	RC	All zeros	25.3.6.1/25-77
0x4_1CE0	MSIRD14—Shared message signaled interrupt register (Bank D) 14	RC	All zeros	25.3.6.1/25-77
0x4_1CF0	MSIRD15—Shared message signaled interrupt register (Bank D) 15	RC	All zeros	25.3.6.1/25-77
0x4_1D20	MSISRD—Shared message signaled interrupt status register (Bank D)	R	All zeros	25.3.6.2/25-78
0x4_1D40	MSIIRD—Shared message signaled interrupt index register (Bank D)	W	All zeros	25.3.6.3/25-78
0x4_1D50–0x4_1FF0	Reserved	—	—	—
Global Timer Group B Registers				
0x4_20F0	TFRRB —Timer frequency reporting register (Group B)	R/W	All zeros	25.3.2.1/25-48
0x4_2100	GTCCRB0—Global timer current count register (Group B) 0	R	All zeros	25.3.2.2/25-49
0x4_2110	GTBCRB0 —Global timer base count register (Group B) 0	R/W	0x8000_0000	25.3.2.3/25-49
0x4_2120	GTVPRB0—Global timer vector/priority register (Group B) 0	R/W	0x8000_0000	25.3.2.4/25-50
0x4_2130	GTDRB0—Global timer destination register (Group B) 0	R/W	0x0000_0001	25.3.2.5/25-51
0x4_2140	GTCCRB1—Global timer current count register (Group B) 1	R	All zeros	25.3.2.2/25-49
0x4_2150	GTBCRB1—Global timer base count register (Group B) 1	R/W	0x8000_0000	25.3.2.3/25-49
0x4_2160	GTVPRB1—Global timer vector/priority register (Group B) 1	R/W	0x8000_0000	25.3.2.4/25-50

Table 25-5. MPIC Register Address Map

Offset	Register	Access	Reset	Section/Page
0x4_2170	GTDRB1—Global timer destination register (Group B) 1	R/W	0x0000_0001	25.3.2.5/25-51
0x4_2180	GTCCRB2—Global timer current count register (Group B) 2	R	All zeros	25.3.2.2/25-49
0x4_2190	GTBCRB2—Global timer base count register (Group B) 2	R/W	0x8000_0000	25.3.2.3/25-49
0x4_21A0	GTVPRB2—Global timer vector/priority register (Group B) 2	R/W	0x8000_0000	25.3.2.4/25-50
0x4_21B0	GTDRB2—Global timer destination register (Group B) 2	R/W	0x0000_0001	25.3.2.5/25-51
0x4_21C0	GTCCRB3—Global timer current count register (Group B) 3	R	All zeros	25.3.2.2/25-49
0x4_21D0	GTBCRB3—Global timer base count register (Group B) 3	R/W	0x8000_0000	25.3.2.3/25-49
0x4_21E0	GTVPRB3—Global timer vector/priority register (Group B)	R/W	0x8000_0000	25.3.2.4/25-50
0x4_21F0	GTDRB3—Global timer destination register (Group B) 3	R/W	0x0000_0001	25.3.2.5/25-51
0x4_2200-0x4_22F0	Reserved	—	—	—
0x4_2300	TCRB—Timer control register (Group B)	R/W	All zeros	25.3.2.6/25-52
0x4_2310–0x4_23F0	Reserved	—	—	—
Global Message Group B Registers				
0x4_2400	MSGRB0—Message register (Group B) 0	R/W	All zeros	25.3.5.1/25-75
0x4_2410	MSGRB1—Message register (Group B) 1			
0x4_2420	MSGRB2—Message register (Group B) 2			
0x4_2430	MSGRB3—Message register (Group B) 3			
0x4_2440	MSGRB4—Message register (Group B) 4			
0x4_2450	MSGRB5—Message register (Group B) 5			
0x4_2460	MSGRB6—Message register (Group B) 6			
0x4_2470	MSGRB7—Message register (Group B) 7			
0x_2480–0x_24F0	Reserved	—	—	—
0x4_2500	MERB—Message enable register (Group B)	R/W	All zeros	25.3.5.2/25-75
0x4_2510	MSRB—Message status register (Group B)	R/W	All zeros	25.3.5.3/25-76
0x4_2520–0x4_2FF0	Reserved	—	—	—
Global Performance Monitor Registers				
0x4_3000	PM0MR0—Performance monitor 0 mask register 0	R/W	0x00FF_FFFF	25.3.3.1/25-54
0x4_3010	PM0MR1—Performance monitor 0 mask register 1	R/W	0x0000_F000	25.3.3.2/25-54
0x4_3020	PM0MR2—Performance monitor 0 mask register 2	R/W	0xFFFF_FFFF	25.3.3.3/25-55
0x4_3030	PM0MR3—Performance monitor 0 mask register 3	R/W	0xFFFF_FFFF	25.3.3.4/25-56

Table 25-5. MPIC Register Address Map

Offset	Register	Access	Reset	Section/Page
0x4_3040	PM0MR4—Performance monitor 0 mask register 4	R/W	0xFFFF_FFFF	25.3.3.5/25-56
0x4_3050	PM0MR5—Performance monitor 0 mask register 5	R/W	0xFFFF_FFFF	25.3.3.6/25-57
0x4_3060	PM0MR6—Performance monitor 0 mask register 6	R/W	0xFFFF_FFFF	25.3.3.7/25-57
0x4_3070	PM0MR7—Performance monitor 0 mask register 7	R/W	0xFFFF_FFFF	25.3.3.8/25-57
0x4_3080	PM0MR8—Performance monitor 0 mask register 8	R/W	0xFFFF_FFFF	25.3.3.9/25-58
0x4_3090	PM0MR9—Performance monitor 0 mask register 9	R/W	0xFFFF_FFFF	25.3.3.10/25-58
0x4_30A0	PM0MR10—Performance monitor 0 mask register 10	R/W	0xFFFF_FFFF	25.3.3.11/25-59
0x4_30B0	PM0MR11—Performance monitor 0 mask register 11	R/W	0xFFFF_FFFF	25.3.3.12/25-59
0x4_30C0–0x4_30F0	Reserved	—	—	—
0x4_3100	PM1MR0—Performance monitor 1 mask register 0	R/W	0x00FF_FFFF	25.3.3.1/25-54
0x4_3110	PM1MR1—Performance monitor 1 mask register 1	R/W	0x0000_F000	25.3.3.2/25-54
0x4_3120	PM1MR2—Performance monitor 1 mask register 2	R/W	0xFFFF_FFFF	25.3.3.3/25-55
0x4_3130	PM1MR3—Performance monitor 1 mask register 3	R/W	0xFFFF_FFFF	25.3.3.4/25-56
0x4_3140	PM1MR4—Performance monitor 1 mask register 4	R/W	0xFFFF_FFFF	25.3.3.5/25-56
0x4_3150	PM1MR5—Performance monitor 1 mask register 5	R/W	0xFFFF_FFFF	25.3.3.6/25-57
0x4_3160	PM1MR6—Performance monitor 1 mask register 6	R/W	0xFFFF_FFFF	25.3.3.7/25-57
0x4_3170	PM1MR7—Performance monitor 1 mask register 7	R/W	0xFFFF_FFFF	25.3.3.8/25-57
0x4_3180	PM1MR8—Performance monitor 1 mask register 8	R/W	0xFFFF_FFFF	25.3.3.9/25-58
0x4_3190	PM1MR9—Performance monitor 1 mask register 9	R/W	0xFFFF_FFFF	25.3.3.10/25-58
0x4_31A0	PM1MR10—Performance monitor 1 mask register 10	R/W	0xFFFF_FFFF	25.3.3.11/25-59
0x4_31B0	PM1MR11—Performance monitor 1 mask register 11	R/W	0xFFFF_FFFF	25.3.3.12/25-59
0x4_31C0–0x4_31F0	Reserved	—	—	—
0x4_3200	PM2MR0—Performance monitor 2 mask register 0	R/W	0x00FF_FFFF	25.3.3.1/25-54
0x4_3210	PM2MR1—Performance monitor 2 mask register 1	R/W	0x0000_F000	25.3.3.2/25-54
0x4_3220	PM2MR2—Performance monitor 2 mask register 2	R/W	0xFFFF_FFFF	25.3.3.3/25-55
0x4_3230	PM2MR3—Performance monitor 2 mask register 3	R/W	0xFFFF_FFFF	25.3.3.4/25-56
0x4_3240	PM2MR4—Performance monitor 2 mask register 4	R/W	0xFFFF_FFFF	25.3.3.5/25-56
0x4_3250	PM2MR5—Performance monitor 2 mask register 5	R/W	0xFFFF_FFFF	25.3.3.6/25-57
0x4_3260	PM2MR6—Performance monitor 2 mask register 6	R/W	0xFFFF_FFFF	25.3.3.7/25-57
0x4_3270	PM2MR7—Performance monitor 2 mask register 7	R/W	0xFFFF_FFFF	25.3.3.8/25-57
0x4_3280	PM2MR8—Performance monitor 2 mask register 8	R/W	0xFFFF_FFFF	25.3.3.9/25-58

Table 25-5. MPIC Register Address Map

Offset	Register	Access	Reset	Section/Page
0x4_3290	PM2MR9—Performance monitor 2 mask register 9	R/W	0xFFFF_FFFF	25.3.3.10/25-58
0x4_32A0	PM2MR10—Performance monitor 2 mask register 10	R/W	0xFFFF_FFFF	25.3.3.11/25-59
0x4_32B0	PM2MR11—Performance monitor 2 mask register 11	R/W	0xFFFF_FFFF	25.3.3.12/25-59
0x4_32C0–0x4_32F0	Reserved	—	—	—
0x4_3300	PM3MR0—Performance monitor 3 mask register 0	R/W	0x00FF_FFFF	25.3.3.1/25-54
0x4_3310	PM3MR1—Performance monitor 3 mask register 1	R/W	0x0000_F000	25.3.3.2/25-54
0x4_3320	PM3MR2—Performance monitor 3 mask register 2	R/W	0xFFFF_FFFF	25.3.3.3/25-55
0x4_3330	PM3MR3—Performance monitor 3 mask register 3	R/W	0xFFFF_FFFF	25.3.3.4/25-56
0x4_3340	PM3MR4—Performance monitor 3 mask register 4	R/W	0xFFFF_FFFF	25.3.3.5/25-56
0x4_3350	PM3MR5—Performance monitor 3 mask register 5	R/W	0xFFFF_FFFF	25.3.3.6/25-57
0x4_3360	PM3MR6—Performance monitor 3 mask register 6	R/W	0xFFFF_FFFF	25.3.3.7/25-57
0x4_3370	PM3MR7—Performance monitor 3 mask register 7	R/W	0xFFFF_FFFF	25.3.3.8/25-57
0x4_3380	PM3MR8—Performance monitor 3 mask register 8	R/W	0xFFFF_FFFF	25.3.3.9/25-58
0x4_3390	PM3MR9—Performance monitor 3 mask register 9	R/W	0xFFFF_FFFF	25.3.3.10/25-58
0x4_33A0	PM3MR10—Performance monitor 3 mask register 10	R/W	0xFFFF_FFFF	25.3.3.11/25-59
0x4_33B0	PM3MR11—Performance monitor 3 mask register 11	R/W	0xFFFF_FFFF	25.3.3.12/25-59
0x4_33C0–0x4_37F0	Reserved	—	—	—
Global Interrupt Summary Registers				
0x4_3800	ERQSR—External interrupt summary register	R	All zeros	25.3.4.1/25-60
0x4_3810–0x4_38F0	Reserved	—	—	—
0x4_3900	EISR—Error interrupt summary register	R	All zeros	25.3.4.2/25-60
0x4_3910	EIMR0—Error interrupt mask register 0	R/W	All zeros	25.3.4.2/25-60
0x4_3920–0x4_39F0	Reserved	—	—	—
0x4_3A00	WSRSR—Watchdog status register summary register	R/W1C	All zeros	25.3.4.4/25-61
0x4_3A20–0x4_3AF0	Reserved	—	—	—
0x4_3B00	CISR0—Critical interrupt summary register 0	R	All zeros	25.3.4.5.1/25-62
0x4_3B10–0x4_3B30	Reserved	—	—	—
0x4_3B40	CISR1—Critical interrupt summary register 1	R	All zeros	25.3.4.5.2/25-63
0x4_3B50	CISR2—Critical interrupt summary register 2	R	All zeros	25.3.4.5.3/25-63

Table 25-5. MPIC Register Address Map

Offset	Register	Access	Reset	Section/Page
0x4_3B60	CISR3—Critical interrupt summary register 3	R	All zeros	25.3.4.5.4/25-64
0x4_3B70	CISR4—Critical interrupt summary register 4	R	All zeros	25.3.4.5.5/25-64
0x4_3B80	CISR5—Critical interrupt summary register 5	R	All zeros	25.3.4.5.6/25-65
0x4_3B90	CISR6—Critical interrupt summary register 6	R	All zeros	25.3.4.5.7/25-65
0x4_3BA0	CISR7—Critical interrupt summary register 7	R	All zeros	25.3.4.5.8/25-66
0x4_3BB0	CISR8—Critical interrupt summary register 8	R	All zeros	25.3.4.5.9/25-66
0x4_3BC0 – 0x4_3BF0	Reserved	—	—	—
0x4_3C00	MCSR0—Machine check summary register 0	R	All zeros	25.3.4.5.10/25-67
0x4_3C10– 0x4_3C30	Reserved	—	—	—
0x4_3C40	MCSR1—Machine check summary register 1	R	All zeros	25.3.4.5.11/25-67
0x4_3C50	MCSR2—Machine check summary register 2	R	All zeros	25.3.4.5.12/25-68
0x4_3C60	MCSR3—Machine check summary register 3	R	All zeros	25.3.4.5.13/25-68
0x4_3C70	MCSR4—Machine check summary register 4	R	All zeros	25.3.4.5.14/25-68
0x4_3C80	MCSR5—Machine check summary register 5	R	All zeros	25.3.4.5.15/25-69
0x4_3C90	MCSR6—Machine check summary register 6	R	All zeros	25.3.4.5.16/25-69
0x4_3CA0	MCSR7—Machine check summary register 7	R	All zeros	25.3.4.5.17/25-70
0x4_3CB0	MCSR8—Machine check summary register 8	R	All zeros	25.3.4.5.18/25-70
0x4_3CC0 – 0x4_3CF0	Reserved	—	—	—
0x4_3D00	IRQSIESR0—$\overline{\text{IRQ\_OUT}}$/Soc Interrupt Event summary register 0	R	All zeros	25.3.4.5.19/25-71
0x4_3D10– 0x4_3D30	Reserved	—	—	—
0x4_3D40	IRQSIESR1—$\overline{\text{IRQ\_OUT}}$/Soc Interrupt Event summary register 1	R	All zeros	25.3.4.5.20/25-71
0x4_3D50	IRQSIESR2—$\overline{\text{IRQ\_OUT}}$/Soc Interrupt Event summary register 2	R	All zeros	25.3.4.5.21/25-72
0x4_3D60	IRQSIESR3—$\overline{\text{IRQ\_OUT}}$/Soc Interrupt Event summary register 3	R	All zeros	25.3.4.5.22/25-72
0x4_3D70	IRQSIESR4—$\overline{\text{IRQ\_OUT}}$/Soc Interrupt Event summary register 4	R	All zeros	25.3.4.5.23/25-72
0x4_3D80	IRQSIESR5—$\overline{\text{IRQ\_OUT}}$/Soc Interrupt Event summary register 5	R	All zeros	25.3.4.5.24/25-73
0x4_3D90	IRQSIESR6—$\overline{\text{IRQ\_OUT}}$/Soc Interrupt Event summary register 6	R	All zeros	25.3.4.5.25/25-73
0x4_3DA0	IRQSIESR7—$\overline{\text{IRQ\_OUT}}$/Soc Interrupt Event summary register 7	R	All zeros	25.3.4.5.26/25-74
0x4_3DB0	IRQSIESR8—$\overline{\text{IRQ\_OUT}}$/Soc Interrupt Event summary register 8	R	All zeros	25.3.4.5.27/25-74

Table 25-5. MPIC Register Address Map

Offset	Register	Access	Reset	Section/Page
0x4_3DC0 – 0x4_3FF0	Reserved	—	—	—
Global Shared Message Signaled Interrupt Index Registers (Alias)				
0x4_4000- 0x4_4130	Reserved	—	—	—
0x4_4140	MSIIRA—Shared message signaled interrupt index register (Bank A)	W	All zeros	25.3.6.3/25-78
0x4_4148	MSIIR1A—Shared message signaled interrupt index register 1(Bank A)	W	All zeros	25.3.6.3/25-78
0x4_4150– 0x4_4170	Reserved	—	—	—
0x4_4180	MSICRA0—Shared message signaled interrupt coalescing register 0 (Bank A)	W	All zeros	25.3.6.3/25-78
0x4_4184	MSICRA1—Shared message signaled interrupt coalescing register 1 (Bank A)	W	All zeros	25.3.6.3/25-78
0x4_4188	MSICRA2—Shared message signaled interrupt coalescing register 2 (Bank A)	W	All zeros	25.3.6.3/25-78
0x4_418C	MSICRA3—Shared message signaled interrupt coalescing register 3 (Bank A)	W	All zeros	25.3.6.3/25-78
0x4_4190	MSICRA4—Shared message signaled interrupt coalescing register 4 (Bank A)	W	All zeros	25.3.6.3/25-78
0x4_4194	MSICRA5—Shared message signaled interrupt coalescing register 5 (Bank A)	W	All zeros	25.3.6.3/25-78
0x4_4198	MSICRA6—Shared message signaled interrupt coalescing register 6 (Bank A)	W	All zeros	25.3.6.3/25-78
0x4_419C	MSICRA7—Shared message signaled interrupt coalescing register 7 (Bank A)	W	All zeros	25.3.6.3/25-78
0x4_41A0	MSICRA8—Shared message signaled interrupt coalescing register 8 (Bank A)	W	All zeros	25.3.6.3/25-78
0x4_41A4	MSICRA9—Shared message signaled interrupt coalescing register 9 (Bank A)	W	All zeros	25.3.6.3/25-78
0x4_41A8	MSICRA10—Shared message signaled interrupt coalescing register 10 (Bank A)	W	All zeros	25.3.6.3/25-78
0x4_41AC	MSICRA11—Shared message signaled interrupt coalescing register 11 (Bank A)	W	All zeros	25.3.6.3/25-78
0x4_41B0	MSICRA12—Shared message signaled interrupt coalescing register 12 (Bank A)	W	All zeros	25.3.6.3/25-78
0x4_41B4	MSICRA13—Shared message signaled interrupt coalescing register 13 (Bank A)	W	All zeros	25.3.6.3/25-78

Table 25-5. MPIC Register Address Map

Offset	Register	Access	Reset	Section/Page
0x4_41B8	MSICRA14—Shared message signaled interrupt coalescing register 14 (Bank A)	W	All zeros	25.3.6.3/25-78
0x4_41BC	MSICRA15—Shared message signaled interrupt coalescing register 15 (Bank A)	W	All zeros	25.3.6.3/25-78
0x4_41C0–0x4_5130	Reserved	—	—	—
0x4_5140	MSIIRB—Shared message signaled interrupt index register (Bank B)	W	All zeros	25.3.6.3/25-78
0x4_5148	MSIIR1B—Shared message signaled interrupt index register 1(Bank B)	W	All zeros	25.3.6.3/25-78
0x4_5150–0x4_5170	Reserved	—	—	—
0x4_5180	MSICRB0—Shared message signaled interrupt coalescing register 0 (Bank B)	W	All zeros	25.3.6.3/25-78
0x4_5184	MSICRB1—Shared message signaled interrupt coalescing register 1 (Bank B)	W	All zeros	25.3.6.3/25-78
0x4_5188	MSICRB2—Shared message signaled interrupt coalescing register 2 (Bank B)	W	All zeros	25.3.6.3/25-78
0x4_518C	MSICRB3—Shared message signaled interrupt coalescing register 3 (Bank B)	W	All zeros	25.3.6.3/25-78
0x4_5190	MSICRB4—Shared message signaled interrupt coalescing register 4 (Bank B)	W	All zeros	25.3.6.3/25-78
0x4_5194	MSICRB5—Shared message signaled interrupt coalescing register 5 (Bank B)	W	All zeros	25.3.6.3/25-78
0x4_5198	MSICRB6—Shared message signaled interrupt coalescing register 6 (Bank B)	W	All zeros	25.3.6.3/25-78
0x4_519C	MSICRB7—Shared message signaled interrupt coalescing register 7 (Bank B)	W	All zeros	25.3.6.3/25-78
0x4_51A0	MSICRB8—Shared message signaled interrupt coalescing register 8 (Bank B)	W	All zeros	25.3.6.3/25-78
0x4_51A4	MSICRB9—Shared message signaled interrupt coalescing register 9 (Bank B)	W	All zeros	25.3.6.3/25-78
0x4_51A8	MSICRB10—Shared message signaled interrupt coalescing register 10 (Bank B)	W	All zeros	25.3.6.3/25-78
0x4_51AC	MSICRB11—Shared message signaled interrupt coalescing register 11 (Bank B)	W	All zeros	25.3.6.3/25-78
0x4_51B0	MSICRB12—Shared message signaled interrupt coalescing register 12 (Bank B)	W	All zeros	25.3.6.3/25-78
0x4_51B4	MSICRB13—Shared message signaled interrupt coalescing register 13 (Bank B)	W	All zeros	25.3.6.3/25-78

Table 25-5. MPIC Register Address Map

Offset	Register	Access	Reset	Section/Page
0x4_51B8	MSICRB14—Shared message signaled interrupt coalescing register 14 (Bank B)	W	All zeros	25.3.6.3/25-78
0x4_51BC	MSICRB15—Shared message signaled interrupt coalescing register 15 (Bank B)	W	All zeros	25.3.6.3/25-78
0x4_51C0–0x4_6130	Reserved	—	—	—
0x4_6140	MSIIRC—Shared message signaled interrupt index register (Bank C)	W	All zeros	25.3.6.3/25-78
0x4_6148	MSIIR1C—Shared message signaled interrupt index register 1(Bank C)	W	All zeros	25.3.6.3/25-78
0x4_6150-0x4_6170	Reserved	—	—	—
0x4_6180	MSICRC0—Shared message signaled interrupt coalescing register 0 (Bank C)	W	All zeros	25.3.6.3/25-78
0x4_6184	MSICRC1—Shared message signaled interrupt coalescing register 1 (Bank C)	W	All zeros	25.3.6.3/25-78
0x4_6188	MSICRC2—Shared message signaled interrupt coalescing register 2 (Bank C)	W	All zeros	25.3.6.3/25-78
0x4_618C	MSICRC3—Shared message signaled interrupt coalescing register 3 (Bank C)	W	All zeros	25.3.6.3/25-78
0x4_6190	MSICRC4—Shared message signaled interrupt coalescing register 4 (Bank C)	W	All zeros	25.3.6.3/25-78
0x4_6194	MSICRC5—Shared message signaled interrupt coalescing register 5 (Bank C)	W	All zeros	25.3.6.3/25-78
0x4_6198	MSICRC6—Shared message signaled interrupt coalescing register 6 (Bank C)	W	All zeros	25.3.6.3/25-78
0x4_619C	MSICRC7—Shared message signaled interrupt coalescing register 7 (Bank C)	W	All zeros	25.3.6.3/25-78
0x4_61A0	MSICRC8—Shared message signaled interrupt coalescing register 8 (Bank C)	W	All zeros	25.3.6.3/25-78
0x4_61A4	MSICRC9—Shared message signaled interrupt coalescing register 9 (Bank C)	W	All zeros	25.3.6.3/25-78
0x4_61A8	MSICRC10—Shared message signaled interrupt coalescing register 10 (Bank C)	W	All zeros	25.3.6.3/25-78
0x4_61AC	MSICRC11—Shared message signaled interrupt coalescing register 11 (Bank C)	W	All zeros	25.3.6.3/25-78
0x4_61B0	MSICRC12—Shared message signaled interrupt coalescing register 12 (Bank C)	W	All zeros	25.3.6.3/25-78
0x4_61B4	MSICRC13—Shared message signaled interrupt coalescing register 13 (Bank C)	W	All zeros	25.3.6.3/25-78

Table 25-5. MPIC Register Address Map

Offset	Register	Access	Reset	Section/Page
0x4_61B8	MSICRC14—Shared message signaled interrupt coalescing register 14 (Bank C)	W	All zeros	25.3.6.3/25-78
0x4_61BC	MSICRC15—Shared message signaled interrupt coalescing register 15 (Bank C)	W	All zeros	25.3.6.3/25-78
0x4_61C0-0x4_7130	Reserved	—	—	—
0x4_7140	MSIIRD—Shared message signaled interrupt index register (Bank D)	W	All zeros	25.3.6.3/25-78
0x4_7148	MSIIR1D—Shared message signaled interrupt index register 1(Bank D)	W	All zeros	25.3.6.3/25-78
0x4_7150–0x4_7170	Reserved	—	—	—
0x4_7180	MSICRD0—Shared message signaled interrupt coalescing register 0 (Bank D)	W	All zeros	25.3.6.3/25-78
0x4_7184	MSICRD1—Shared message signaled interrupt coalescing register 1 (Bank D)	W	All zeros	25.3.6.3/25-78
0x4_7188	MSICRD2—Shared message signaled interrupt coalescing register 2 (Bank D)	W	All zeros	25.3.6.3/25-78
0x4_718C	MSICRD3—Shared message signaled interrupt coalescing register 3 (Bank D)	W	All zeros	25.3.6.3/25-78
0x4_7190	MSICRD4—Shared message signaled interrupt coalescing register 4 (Bank D)	W	All zeros	25.3.6.3/25-78
0x4_7194	MSICRD5—Shared message signaled interrupt coalescing register 5 (Bank D)	W	All zeros	25.3.6.3/25-78
0x4_7198	MSICRD6—Shared message signaled interrupt coalescing register 6 (Bank D)	W	All zeros	25.3.6.3/25-78
0x4_719C	MSICRD7—Shared message signaled interrupt coalescing register 7 (Bank D)	W	All zeros	25.3.6.3/25-78
0x4_71A0	MSICRD8—Shared message signaled interrupt coalescing register 8 (Bank D)	W	All zeros	25.3.6.3/25-78
0x4_71A4	MSICRD9—Shared message signaled interrupt coalescing register 9 (Bank D)	W	All zeros	25.3.6.3/25-78
0x4_71A8	MSICRD10—Shared message signaled interrupt coalescing register 10 (Bank D)	W	All zeros	25.3.6.3/25-78
0x4_71AC	MSICRD11—Shared message signaled interrupt coalescing register 11 (Bank D)	W	All zeros	25.3.6.3/25-78
0x4_71B0	MSICRD12—Shared message signaled interrupt coalescing register 12 (Bank D)	W	All zeros	25.3.6.3/25-78
0x4_71B4	MSICRD13—Shared message signaled interrupt coalescing register 13 (Bank D)	W	All zeros	25.3.6.3/25-78

Table 25-5. MPIC Register Address Map

Offset	Register	Access	Reset	Section/Page
0x4_71B8	MSICRD14—Shared message signaled interrupt coalescing register 14 (Bank D)	W	All zeros	25.3.6.3/25-78
0x4_71BC	MSICRD15—Shared message signaled interrupt coalescing register 15 (Bank D)	W	All zeros	25.3.6.3/25-78
0x4_71C0–0x4_FFF0	Reserved	—	—	—

Table 25-6 shows the MPIC registers used to configure the interrupt sources.

Table 25-6. MPIC Register Address Map — Interrupt Source Configuration Registers

Offset	Register	Access	Reset	Section/Page
External Interrupt Configuration Registers				
0x5_0000	EIVPR0—External interrupt 0 (IRQ0) vector/priority register	R/W	0x8000_0000	25.3.7.1/25-85
0x5_0010	EIDR0—External interrupt 0 (IRQ0) destination register	R/W	0x0000_0001	25.3.7.2/25-86
0x5_0018	EILR0—External interrupt 0 (IRQ0) level register	R/W	All zeros	25.3.7.3/25-88
0x5_0020	EIVPR1—External interrupt 1 (IRQ1) vector/priority register	R/W	0x8000_0000	25.3.7.1/25-85
0x5_0030	EIDR1—External interrupt 1 (IRQ1) destination register	R/W	0x0000_0001	25.3.7.2/25-86
0x5_0038	EILR1—External interrupt 1 (IRQ1) level register	R/W	All zeros	25.3.7.3/25-88
0x5_0040	EIVPR2—External interrupt 2 (IRQ2) vector/priority register	R/W	0x8000_0000	25.3.7.1/25-85
0x5_0050	EIDR2—External interrupt 2 (IRQ2) destination register	R/W	0x0000_0001	25.3.7.2/25-86
0x5_0058	EILR2—External interrupt 2 (IRQ2) level register	R/W	All zeros	25.3.7.3/25-88
0x5_0060	EIVPR3—External interrupt 3 (IRQ3) vector/priority register	R/W	0x8000_0000	25.3.7.1/25-85
0x5_0070	EIDR3—External interrupt 3 (IRQ3) destination register	R/W	0x0000_0001	25.3.7.2/25-86
0x5_0078	EILR3—External interrupt 3 (IRQ3) level register	R/W	All zeros	25.3.7.3/25-88
0x5_0080	EIVPR4—External interrupt 4 (IRQ4) vector/priority register	R/W	0x8000_0000	25.3.7.1/25-85
0x5_0090	EIDR4—External interrupt 4 (IRQ4) destination register	R/W	0x0000_0001	25.3.7.2/25-86
0x5_0098	EILR4—External interrupt 4 (IRQ4) level register	R/W	All zeros	25.3.7.3/25-88
0x5_00A0	EIVPR5—External interrupt 5 (IRQ5) vector/priority register	R/W	0x8000_0000	25.3.7.1/25-85
0x5_00B0	EIDR5—External interrupt 5 (IRQ58) destination register	R/W	0x0000_0001	25.3.7.2/25-86
0x5_00B8	EILR5—External interrupt 5 (IRQ0) level register	R/W	All zeros	25.3.7.3/25-88
0x5_00C0	EIVPR6—External interrupt 6 (IRQ6) vector/priority register	R/W	0x8000_0000	25.3.7.1/25-85
0x5_00D0	EIDR6—External interrupt 6 (IRQ6) destination register	R/W	0x0000_0001	25.3.7.2/25-86
0x5_00D8	EILR6—External interrupt 6 (IRQ6) level register	R/W	All zeros	25.3.7.3/25-88
0x5_00E0	EIVPR7—External interrupt 7 (IRQ7) vector/priority register	R/W	0x8000_0000	25.3.7.1/25-85
0x5_00F0	EIDR7—External interrupt 7 (IRQ7) destination register	R/W	0x0000_0001	25.3.7.2/25-86

Table 25-6. MPIC Register Address Map — Interrupt Source Configuration Registers

Offset	Register	Access	Reset	Section/Page
0x5_00F8	EILR7—External interrupt 7 (IRQ7) level register	R/W	All zeros	25.3.7.3/25-88
0x5_0100	EIVPR8—External interrupt 8 (IRQ8) vector/priority register	R/W	0x8000_0000	25.3.7.1/25-85
0x5_0110	EIDR8—External interrupt 8 (IRQ8) destination register	R/W	0x0000_0001	25.3.7.2/25-86
0x5_0118	EILR8—External interrupt 8 (IRQ8) level register	R/W	All zeros	25.3.7.3/25-88
0x5_0120	EIVPR9—External interrupt 9 (IRQ9) vector/priority register	R/W	0x8000_0000	25.3.7.1/25-85
0x5_0130	EIDR9—External interrupt 9 (IRQ9) destination register	R/W	0x0000_0001	25.3.7.2/25-86
0x5_0138	EILR9—External interrupt 9 (IRQ9) level register	R/W	All zeros	25.3.7.3/25-88
0x5_0140	EIVPR10—External interrupt 10 (IRQ10) vector/priority register	R/W	0x8000_0000	25.3.7.1/25-85
0x5_0150	EIDR10—External interrupt 10 (IRQ10) destination register	R/W	0x0000_0001	25.3.7.2/25-86
0x5_0158	EILR10—External interrupt 10 (IRQ10) level register	R/W	All zeros	25.3.7.3/25-88
0x5_0160	EIVPR11—External interrupt 11 (IRQ11) vector/priority register	R/W	0x8000_0000	25.3.7.1/25-85
0x5_0170	EIDR11—External interrupt 11 (IRQ11) destination register	R/W	0x0000_0001	25.3.7.2/25-86
0x5_0178	EILR11—External interrupt 11 (IRQ11) level register	R/W	All zeros	25.3.7.3/25-88
0x5_0180–0x5_01F0	Reserved	—	—	—
Internal Interrupt Configuration Registers				
0x5_0200	IIVPR0—Internal interrupt 0 vector/priority register	R/W	0x8080_0000	25.3.7.4/25-88
0x5_0210	IIDR0—Internal interrupt 0 destination register	R/W	0x0000_0001	25.3.7.5/25-89
0x5_0218	IILR0—Internal interrupt 0 level register	R/W	All zeros	25.3.7.6/25-90
0x5_0220	IIVPR1—Internal interrupt 1 vector/priority register	R/W	0x8080_0000	25.3.7.4/25-88
0x5_0230	IIDR1—Internal interrupt 1 destination register	R/W	0x0000_0001	25.3.7.5/25-89
0x5_0238	IILR1—Internal interrupt 1 level register	R/W	All zeros	25.3.7.6/25-90
0x5_0240	IIVPR2—Internal interrupt 2 vector/priority register	R/W	0x8080_0000	25.3.7.4/25-88
0x5_0250	IIDR2—Internal interrupt 2 destination register	R/W	0x0000_0001	25.3.7.5/25-89
0x5_0258	IILR2—Internal interrupt 2 level register	R/W	All zeros	25.3.7.6/25-90
0x5_0260	IIVPR3—Internal interrupt 3 vector/priority register	R/W	0x8080_0000	25.3.7.4/25-88
0x5_0270	IIDR3—Internal interrupt 3 destination register	R/W	0x0000_0001	25.3.7.5/25-89
0x5_0278	IILR3—Internal interrupt 3 level register	R/W	All zeros	25.3.7.6/25-90
0x5_0280	IIVPR4—Internal interrupt 4 vector/priority register	R/W	0x8080_0000	25.3.7.4/25-88
0x5_0290	IIDR4—Internal interrupt 4 destination register	R/W	0x0000_0001	25.3.7.5/25-89
0x5_0298	IILR4—Internal interrupt 4 level register	R/W	All zeros	25.3.7.6/25-90
0x5_02A0	IIVPR5—Internal interrupt 5 vector/priority register	R/W	0x8080_0000	25.3.7.4/25-88
0x5_02B0	IIDR5—Internal interrupt 5 destination register	R/W	0x0000_0001	25.3.7.5/25-89

Table 25-6. MPIC Register Address Map — Interrupt Source Configuration Registers

Offset	Register	Access	Reset	Section/Page
0x5_02B8	IILR5—Internal interrupt 5 level register	R/W	All zeros	25.3.7.6/25-90
0x5_02C0	IIVPR6—Internal interrupt 6 vector/priority register	R/W	0x8080_0000	25.3.7.4/25-88
0x5_02D0	IIDR6—Internal interrupt 6 destination register	R/W	0x0000_0001	25.3.7.5/25-89
0x5_02D8	IILR6—Internal interrupt 6 level register	R/W	All zeros	25.3.7.6/25-90
0x5_02E0	IIVPR7—Internal interrupt 7 vector/priority register	R/W	0x8080_0000	25.3.7.4/25-88
0x5_02F0	IIDR7—Internal interrupt 7 destination register	R/W	0x0000_0001	25.3.7.5/25-89
0x5_02F8	IILR7—Internal interrupt 7 level register	R/W	All zeros	25.3.7.6/25-90
0x5_0300	IIVPR8—Internal interrupt 8 vector/priority register	R/W	0x8080_0000	25.3.7.4/25-88
0x5_0310	IIDR8—Internal interrupt 8 destination register	R/W	0x0000_0001	25.3.7.5/25-89
0x5_0318	IILR8—Internal interrupt 8 level register	R/W	All zeros	25.3.7.6/25-90
0x5_0320	IIVPR9—Internal interrupt 9 vector/priority register	R/W	0x8080_0000	25.3.7.4/25-88
0x5_0330	IIDR9—Internal interrupt 9 destination register	R/W	0x0000_0001	25.3.7.5/25-89
0x5_0338	IILR9—Internal interrupt 9 level register	R/W	All zeros	25.3.7.6/25-90
0x5_0340	IIVPR10—Internal interrupt 10 vector/priority register	R/W	0x8080_0000	25.3.7.4/25-88
0x5_0350	IIDR10—Internal interrupt 10 destination register	R/W	0x0000_0001	25.3.7.5/25-89
0x5_0358	IILR10—Internal interrupt 10 level register	R/W	All zeros	25.3.7.6/25-90
0x5_0360	IIVPR11—Internal interrupt 11 vector/priority register	R/W	0x8080_0000	25.3.7.4/25-88
0x5_0370	IIDR11—Internal interrupt 11 destination register	R/W	0x0000_0001	25.3.7.5/25-89
0x5_0378	IILR11—Internal interrupt 11 level register	R/W	All zeros	25.3.7.6/25-90
0x5_0380	IIVPR12—Internal interrupt 12 vector/priority register	R/W	0x8080_0000	25.3.7.4/25-88
0x5_0390	IIDR12—Internal interrupt 12 destination register	R/W	0x0000_0001	25.3.7.5/25-89
0x5_0398	IILR12—Internal interrupt 12 level register	R/W	All zeros	25.3.7.6/25-90
0x5_03A0	IIVPR13—Internal interrupt 13 vector/priority register	R/W	0x8080_0000	25.3.7.4/25-88
0x5_03B0	IIDR13—Internal interrupt 13 destination register	R/W	0x0000_0001	25.3.7.5/25-89
0x5_03B8	IILR13—Internal interrupt 13 level register	R/W	All zeros	25.3.7.6/25-90
0x5_03C0	IIVPR14—Internal interrupt 14 vector/priority register	R/W	0x8080_0000	25.3.7.4/25-88
0x5_03D0	IIDR14—Internal interrupt 14 destination register	R/W	0x0000_0001	25.3.7.5/25-89
0x5_03D8	IILR14—Internal interrupt 14 level register	R/W	All zeros	25.3.7.6/25-90
0x5_03E0	IIVPR15—Internal interrupt 15 vector/priority register	R/W	0x8080_0000	25.3.7.4/25-88
0x5_03F0	IIDR15—Internal interrupt 15 destination register	R/W	0x0000_0001	25.3.7.5/25-89
0x5_03F8	IILR15—Internal interrupt 15 level register	R/W	All zeros	25.3.7.6/25-90
005_0400–0x5_0FE0	IIVPR16–111—Internal interrupt 16-111 vector/priority register	R/W	0x8080_0000	25.3.7.4/25-88

Table 25-6. MPIC Register Address Map — Interrupt Source Configuration Registers

Offset	Register	Access	Reset	Section/Page
0x5_0410–0x5_0FF0	IIDR16–111—Internal interrupt 16-111 destination register	R/W	0x0000_0001	25.3.7.5/25-89
0x5_0418–0x5_0FF8	IILR16–111—Internal interrupt 16-111 level register	R/W	All zeros	25.3.7.6/25-90
0x5_1000	IIVPR112—Internal interrupt 112 vector/priority register	R/W	0x8080_0000	25.3.7.4/25-88
0x5_1010	IIDR112—Internal interrupt 112 destination register	R/W	0x0000_0001	25.3.7.5/25-89
0x5_1018	IILR112—Internal interrupt 112 level register	R/W	All zeros	25.3.7.6/25-90
0x5_1020	IIVPR113—Internal interrupt 113 vector/priority register	R/W	0x8080_0000	25.3.7.4/25-88
0x5_1030	IIDR113—Internal interrupt 113 destination register	R/W	0x0000_0001	25.3.7.5/25-89
0x5_1038	IILR113—Internal interrupt 113 level register	R/W	All zeros	25.3.7.6/25-90
0x5_1040	IIVPR114—Internal interrupt 114 vector/priority register	R/W	0x8080_0000	25.3.7.4/25-88
0x5_1050	IIDR114—Internal interrupt 114 destination register	R/W	0x0000_0001	25.3.7.5/25-89
0x5_1058	IILR114—Internal interrupt 114 level register	R/W	All zeros	25.3.7.6/25-90
0x5_1060	IIVPR115—Internal interrupt 115 vector/priority register	R/W	0x8080_0000	25.3.7.4/25-88
0x5_1070	IIDR115—Internal interrupt 115 destination register	R/W	0x0000_0001	25.3.7.5/25-89
0x5_1078	IILR115—Internal interrupt 115 level register	R/W	All zeros	25.3.7.6/25-90
0x5_1080	IIVPR116—Internal interrupt 116 vector/priority register	R/W	0x8080_0000	25.3.7.4/25-88
0x5_1090	IIDR116—Internal interrupt 116 destination register	R/W	0x0000_0001	25.3.7.5/25-89
0x5_1098	IILR116—Internal interrupt 116 level register	R/W	All zeros	25.3.7.6/25-90
0x5_10A0	IIVPR117—Internal interrupt 117 vector/priority register	R/W	0x8080_0000	25.3.7.4/25-88
0x5_10B0	IIDR117—Internal interrupt 117 destination register	R/W	0x0000_0001	25.3.7.5/25-89
0x5_10B8	IILR117—Internal interrupt 117 level register	R/W	All zeros	25.3.7.6/25-90
0x5_10C0	IIVPR118—Internal interrupt 118 vector/priority register	R/W	0x8080_0000	25.3.7.4/25-88
0x5_10D0	IIDR118—Internal interrupt 118 destination register	R/W	0x0000_0001	25.3.7.5/25-89
0x5_10D8	IILR118—Internal interrupt 118 level register	R/W	All zeros	25.3.7.6/25-90
0x5_10E0	IIVPR119—Internal interrupt 119 vector/priority register	R/W	0x8080_0000	25.3.7.4/25-88
0x5_10F0	IIDR119—Internal interrupt 119 destination register	R/W	0x0000_0001	25.3.7.5/25-89
0x5_10F8	IILR119—Internal interrupt 119 level register	R/W	All zeros	25.3.7.6/25-90
0x5_1100	IIVPR120—Internal interrupt 120 vector/priority register	R/W	0x8080_0000	25.3.7.4/25-88
0x5_1110	IIDR120—Internal interrupt 120 destination register	R/W	0x0000_0001	25.3.7.5/25-89
0x5_1118	IILR120—Internal interrupt 120 level register	R/W	All zeros	25.3.7.6/25-90
0x5_1120	IIVPR121—Internal interrupt 121 vector/priority register	R/W	0x8080_0000	25.3.7.4/25-88
0x5_1130	IIDR121—Internal interrupt 121 destination register	R/W	0x0000_0001	25.3.7.5/25-89

Table 25-6. MPIC Register Address Map — Interrupt Source Configuration Registers

Offset	Register	Access	Reset	Section/Page
0x5_1138	IILR121—Internal interrupt 121 level register	R/W	All zeros	25.3.7.6/25-90
0x5_1140	IIVPR122—Internal interrupt 122 vector/priority register	R/W	0x8080_0000	25.3.7.4/25-88
0x5_1150	IIDR122—Internal interrupt 122 destination register	R/W	0x0000_0001	25.3.7.5/25-89
0x5_1158	IILR122—Internal interrupt 122 level register	R/W	All zeros	25.3.7.6/25-90
0x5_1160	IIVPR123—Internal interrupt 123 vector/priority register	R/W	0x8080_0000	25.3.7.4/25-88
0x5_1170	IIDR123—Internal interrupt 123 destination register	R/W	0x0000_0001	25.3.7.5/25-89
0x5_1178	IILR123—Internal interrupt 123 level register	R/W	All zeros	25.3.7.6/25-90
0x5_1180	IIVPR124—Internal interrupt 124 vector/priority register	R/W	0x8080_0000	25.3.7.4/25-88
0x5_1190	IIDR124—Internal interrupt 124 destination register	R/W	0x0000_0001	25.3.7.5/25-89
0x5_1198	IILR124—Internal interrupt 124 level register	R/W	All zeros	25.3.7.6/25-90
0x5_11A0	IIVPR125—Internal interrupt 125 vector/priority register	R/W	0x8080_0000	25.3.7.4/25-88
0x5_11B0	IIDR125—Internal interrupt 125 destination register	R/W	0x0000_0001	25.3.7.5/25-89
0x5_11B8	IILR125—Internal interrupt 125 level register	R/W	All zeros	25.3.7.6/25-90
0x5_11C0	IIVPR126—Internal interrupt 126 vector/priority register	R/W	0x8080_0000	25.3.7.4/25-88
0x5_11D0	IIDR126—Internal interrupt 126 destination register	R/W	0x0000_0001	25.3.7.5/25-89
0x5_11D8	IILR126—Internal interrupt 126 level register	R/W	All zeros	25.3.7.6/25-90
0x5_11E0	IIVPR127—Internal interrupt 127 vector/priority register	R/W	0x8080_0000	25.3.7.4/25-88
0x5_11F0	IIDR127—Internal interrupt 127 destination register	R/W	0x0000_0001	25.3.7.5/25-89
0x5_11F8	IILR127—Internal interrupt 127 level register	R/W	All zeros	25.3.7.6/25-90
0x5_1200-0x5_15E0	IIVPR128–159—Internal interrupt 128-159 vector/priority register	R/W	0x8080_0000	25.3.7.4/25-88
0x5_1210-0x5_15F0	IIDR128–159—Internal interrupt 128-159 destination register	R/W	0x0000_0001	25.3.7.5/25-89
0x5_1218-0x5_15F8	IILR128–159—Internal interrupt 128-159 level register	R/W	All zeros	25.3.7.6/25-90
Messaging Interrupt Configuration Registers				
0x5_1600	MIVPRA0—Messaging interrupt vector/priority register (Group A) 0	R/W	0x8080_0000	25.3.7.7/25-91
0x5_1610	MIDRA0—Messaging interrupt destination register (Group A) 0	R/W	0x0000_0001	25.3.7.8/25-92
0x5_1620	MIVPRA1—Messaging interrupt vector/priority register (Group A) 1	R/W	0x8080_0000	25.3.7.7/25-91
0x5_1630	MIDRA1—Messaging interrupt destination register (Group A) 1	R/W	0x0000_0001	25.3.7.8/25-92
0x5_1640	MIVPRA2—Messaging interrupt vector/priority register (Group A) 2	R/W	0x8080_0000	25.3.7.7/25-91
0x5_1650	MIDRA2—Messaging interrupt destination register (Group A) 2	R/W	0x0000_0001	25.3.7.8/25-92
0x5_1660	MIVPRA3—Messaging interrupt vector/priority register (Group A) 3	R/W	0x8080_0000	25.3.7.7/25-91
0x5_1670	MIDRA3—Messaging interrupt destination register (Group A) 3	R/W	0x0000_0001	25.3.7.8/25-92

Table 25-6. MPIC Register Address Map — Interrupt Source Configuration Registers

Offset	Register	Access	Reset	Section/Page
0x5_1680	MIVPRB0—Messaging interrupt vector/priority register (Group B) 0	R/W	0x8080_0000	25.3.7.7/25-91
0x5_1690	MIDRB0—Messaging interrupt destination register (Group B) 0	R/W	0x0000_0001	25.3.7.8/25-92
0x5_16A0	MIVPRB1—Messaging interrupt vector/priority register (Group B) 1	R/W	0x8080_0000	25.3.7.7/25-91
0x5_16B0	MIDRB1—Messaging interrupt destination register (Group B) 1	R/W	0x0000_0001	25.3.7.8/25-92
0x5_16C0	MIVPRB2—Messaging interrupt vector/priority register (Group B) 2	R/W	0x8080_0000	25.3.7.7/25-91
0x5_16D0	MIDRB2—Messaging interrupt destination register (Group B) 2	R/W	0x0000_0001	25.3.7.8/25-92
0x5_16E0	MIVPRB3—Messaging interrupt vector/priority register (Group B) 3	R/W	0x8080_0000	25.3.7.7/25-91
0x5_16F0	MIDRB3—Messaging interrupt destination register (Group B) 3	R/W	0x0000_0001	25.3.7.8/25-92
0x5_1700	MIVPRA4—Messaging interrupt vector/priority register (Group A) 4	R/W	0x8080_0000	25.3.7.7/25-91
0x5_1710	MIDRA4—Messaging interrupt destination register (Group A) 4	R/W	0x0000_0001	25.3.7.8/25-92
0x5_1720	MIVPRA5—Messaging interrupt vector/priority register (Group A) 5	R/W	0x8080_0000	25.3.7.7/25-91
0x5_1730	MIDRA5—Messaging interrupt destination register (Group A) 5	R/W	0x0000_0001	25.3.7.8/25-92
0x5_1740	MIVPRA6—Messaging interrupt vector/priority register (Group A) 6	R/W	0x8080_0000	25.3.7.7/25-91
0x5_1750	MIDRA6—Messaging interrupt destination register (Group A) 6	R/W	0x0000_0001	25.3.7.8/25-92
0x5_1760	MIVPRA7—Messaging interrupt vector/priority register (Group A) 7	R/W	0x8080_0000	25.3.7.7/25-91
0x5_1770	MIDRA7—Messaging interrupt destination register (Group A) 7	R/W	0x0000_0001	25.3.7.8/25-92
0x5_1780	MIVPRB4—Messaging interrupt vector/priority register (Group B) 4	R/W	0x8080_0000	25.3.7.7/25-91
0x5_1790	MIDRB4—Messaging interrupt destination register (Group B) 4	R/W	0x0000_0001	25.3.7.8/25-92
0x5_17A0	MIVPRB5—Messaging interrupt vector/priority register (Group B) 5	R/W	0x8080_0000	25.3.7.7/25-91
0x5_17B0	MIDRB5—Messaging interrupt destination register (Group B) 5	R/W	0x0000_0001	25.3.7.8/25-92
0x5_17C0	MIVPRB6—Messaging interrupt vector/priority register (Group B) 6	R/W	0x8080_0000	25.3.7.7/25-91
0x5_17D0	MIDRB6—Messaging interrupt destination register (Group B) 6	R/W	0x0000_0001	25.3.7.8/25-92
0x5_17E0	MIVPRB7—Messaging interrupt vector/priority register (Group B) 7	R/W	0x8080_0000	25.3.7.7/25-91
0x5_17F0	MIDRB7—Messaging interrupt destination register (Group B) 7	R/W	0x0000_0001	25.3.7.8/25-92
0x5_1800–0x5_1BF0	Reserved for additional interrupt sources	—	0x8080_0000	—
Shared Message Signaled Interrupt Configuration Registers				
0x5_1C00	MSIVPRA0—Shared message signaled interrupt vector/priority register (Bank A) 0	R/W	0x8000_0000	25.3.6.6/25-82
0x5_1C10	MSIDRA0—Shared message signaled interrupt destination register (Bank A) 0	R/W	0x0000_0001	25.3.6.7/25-83
0x5_1C20	MSIVPRA1—Shared message signaled interrupt vector/priority register (Bank A) 1	R/W	0x8000_0000	25.3.6.6/25-82

Table 25-6. MPIC Register Address Map — Interrupt Source Configuration Registers

Offset	Register	Access	Reset	Section/Page
0x5_1C30	MSIDRA1—Shared message signaled interrupt destination register (Bank A) 1	R/W	0x0000_0001	25.3.6.7/25-83
0x5_1C40	MSIVPRA2—Shared message signaled interrupt vector/priority register (Bank A) 2	R/W	0x8000_0000	25.3.6.6/25-82
0x5_1C50	MSIDRA2—Shared message signaled interrupt destination register (Bank A) 2	R/W	0x0000_0001	25.3.6.7/25-83
0x5_1C60	MSIVPRA3—Shared message signaled interrupt vector/priority register (Bank A) 3	R/W	0x8000_0000	25.3.6.6/25-82
0x5_1C70	MSIDRA3—Shared message signaled interrupt destination register (Bank A) 3	R/W	0x0000_0001	25.3.6.7/25-83
0x5_1C80	MSIVPRA4—Shared message signaled interrupt vector/priority register (Bank A) 4	R/W	0x8000_0000	25.3.6.6/25-82
0x5_1C90	MSIDRA4—Shared message signaled interrupt destination register (Bank A) 4	R/W	0x0000_0001	25.3.6.7/25-83
0x5_1CA0	MSIVPRA5—Shared message signaled interrupt vector/priority register (Bank A) 5	R/W	0x8000_0000	25.3.6.6/25-82
0x5_1CB0	MSIDRA5—Shared message signaled interrupt destination register (Bank A) 5	R/W	0x0000_0001	25.3.6.7/25-83
0x5_1CC0	MSIVPRA6—Shared message signaled interrupt vector/priority register (Bank A) 6	R/W	0x8000_0000	25.3.6.6/25-82
0x5_1CD0	MSIDRA6—Shared message signaled interrupt destination register (Bank A) 6	R/W	0x0000_0001	25.3.6.7/25-83
0x5_1CE0	MSIVPRA7—Shared message signaled interrupt vector/priority register (Bank A) 7	R/W	0x8000_0000	25.3.6.6/25-82
0x5_1CF0	MSIDRA7—Shared message signaled interrupt destination register (Bank A) 7	R/W	0x0000_0001	25.3.6.7/25-83
0x5_1D00	MSIVPRB0—Shared message signaled interrupt vector/priority register (Bank B) 0	R/W	0x8000_0000	25.3.6.6/25-82
0x5_1D10	MSIDRB0—Shared message signaled interrupt destination register (Bank B) 0	R/W	0x0000_0001	25.3.6.7/25-83
0x5_1D20	MSIVPRB1—Shared message signaled interrupt vector/priority register (Bank B) 1	R/W	0x8000_0000	25.3.6.6/25-82
0x5_1D30	MSIDRB1—Shared message signaled interrupt destination register (Bank B) 1	R/W	0x0000_0001	25.3.6.7/25-83
0x5_1D40	MSIVPRB2—Shared message signaled interrupt vector/priority register (Bank B) 2	R/W	0x8000_0000	25.3.6.6/25-82
0x5_1D50	MSIDRB2—Shared message signaled interrupt destination register (Bank B) 2	R/W	0x0000_0001	25.3.6.7/25-83
0x5_1D60	MSIVPRB3—Shared message signaled interrupt vector/priority register (Bank B) 3	R/W	0x8000_0000	25.3.6.6/25-82

Table 25-6. MPIC Register Address Map — Interrupt Source Configuration Registers

Offset	Register	Access	Reset	Section/Page
0x5_1D70	MSIDRB3—Shared message signaled interrupt destination register (Bank B) 3	R/W	0x0000_0001	25.3.6.7/25-83
0x5_1D80	MSIVPRB4—Shared message signaled interrupt vector/priority register (Bank B) 4	R/W	0x8000_0000	25.3.6.6/25-82
0x5_1D90	MSIDRB4—Shared message signaled interrupt destination register (Bank B) 4	R/W	0x0000_0001	25.3.6.7/25-83
0x5_1DA0	MSIVPRB5—Shared message signaled interrupt vector/priority register (Bank B) 5	R/W	0x8000_0000	25.3.6.6/25-82
0x5_1DB0	MSIDRB5—Shared message signaled interrupt destination register (Bank B) 5	R/W	0x0000_0001	25.3.6.7/25-83
0x5_1DC0	MSIVPRB6—Shared message signaled interrupt vector/priority register (Bank B) 6	R/W	0x8000_0000	25.3.6.6/25-82
0x5_1DD0	MSIDRB6—Shared message signaled interrupt destination register (Bank B) 6	R/W	0x0000_0001	25.3.6.7/25-83
0x5_1DE0	MSIVPRB7—Shared message signaled interrupt vector/priority register (Bank B) 7	R/W	0x8000_0000	25.3.6.6/25-82
0x5_1DF0	MSIDRB7—Shared message signaled interrupt destination register (Bank B) 7	R/W	0x0000_0001	25.3.6.7/25-83
0x5_1E00	MSIVPRC0—Shared message signaled interrupt vector/priority register (Bank C) 0	R/W	0x8000_0000	25.3.6.6/25-82
0x5_1E10	MSIDRC0—Shared message signaled interrupt destination register (Bank C) 0	R/W	0x0000_0001	25.3.6.7/25-83
0x5_1E20	MSIVPRC1—Shared message signaled interrupt vector/priority register (Bank C) 1	R/W	0x8000_0000	25.3.6.6/25-82
0x5_1E30	MSIDRC1—Shared message signaled interrupt destination register (Bank C) 1	R/W	0x0000_0001	25.3.6.7/25-83
0x5_1E40	MSIVPRC2—Shared message signaled interrupt vector/priority register (Bank C) 2	R/W	0x8000_0000	25.3.6.6/25-82
0x5_1E50	MSIDRC2—Shared message signaled interrupt destination register (Bank C) 2	R/W	0x0000_0001	25.3.6.7/25-83
0x5_1E60	MSIVPRC3—Shared message signaled interrupt vector/priority register (Bank C) 3	R/W	0x8000_0000	25.3.6.6/25-82
0x5_1E70	MSIDRC3—Shared message signaled interrupt destination register (Bank C) 3	R/W	0x0000_0001	25.3.6.7/25-83
0x5_1E80	MSIVPRC4—Shared message signaled interrupt vector/priority register (Bank C) 4	R/W	0x8000_0000	25.3.6.6/25-82
0x5_1E90	MSIDRC4—Shared message signaled interrupt destination register (Bank C) 4	R/W	0x0000_0001	25.3.6.7/25-83
0x5_1EA0	MSIVPRC5—Shared message signaled interrupt vector/priority register (Bank C) 5	R/W	0x8000_0000	25.3.6.6/25-82

Table 25-6. MPIC Register Address Map — Interrupt Source Configuration Registers

Offset	Register	Access	Reset	Section/Page
0x5_1EB0	MSIDRC5—Shared message signaled interrupt destination register (Bank C) 5	R/W	0x0000_0001	25.3.6.7/25-83
0x5_1EC0	MSIVPRC6—Shared message signaled interrupt vector/priority register (Bank C) 6	R/W	0x8000_0000	25.3.6.6/25-82
0x5_1ED0	MSIDRC6—Shared message signaled interrupt destination register (Bank C) 6	R/W	0x0000_0001	25.3.6.7/25-83
0x5_1EE0	MSIVPRC7—Shared message signaled interrupt vector/priority register (Bank C) 7	R/W	0x8000_0000	25.3.6.6/25-82
0x5_1EF0	MSIDRC7—Shared message signaled interrupt destination register (Bank C) 7	R/W	0x0000_0001	25.3.6.7/25-83
0x5_1F00	MSIVPRD0—Shared message signaled interrupt vector/priority register (Bank D) 0	R/W	0x8000_0000	25.3.6.6/25-82
0x5_1F10	MSIDRD0—Shared message signaled interrupt destination register (Bank D) 0	R/W	0x0000_0001	25.3.6.7/25-83
0x5_1F20	MSIVPRD1—Shared message signaled interrupt vector/priority register (Bank D) 1	R/W	0x8000_0000	25.3.6.6/25-82
0x5_1F30	MSIDRD1—Shared message signaled interrupt destination register (Bank D) 1	R/W	0x0000_0001	25.3.6.7/25-83
0x5_1F40	MSIVPRD2—Shared message signaled interrupt vector/priority register (Bank D) 2	R/W	0x8000_0000	25.3.6.6/25-82
0x5_1F50	MSIDRD2—Shared message signaled interrupt destination register (Bank D) 2	R/W	0x0000_0001	25.3.6.7/25-83
0x5_1F60	MSIVPRD3—Shared message signaled interrupt vector/priority register (Bank D) 3	R/W	0x8000_0000	25.3.6.6/25-82
0x5_1F70	MSIDRD3—Shared message signaled interrupt destination register (Bank D) 3	R/W	0x0000_0001	25.3.6.7/25-83
0x5_1F80	MSIVPRD4—Shared message signaled interrupt vector/priority register (Bank D) 4	R/W	0x8000_0000	25.3.6.6/25-82
0x5_1F90	MSIDRD4—Shared message signaled interrupt destination register (Bank D) 4	R/W	0x0000_0001	25.3.6.7/25-83
0x5_1FA0	MSIVPRD5—Shared message signaled interrupt vector/priority register (Bank D) 5	R/W	0x8000_0000	25.3.6.6/25-82
0x5_1FB0	MSIDRD5—Shared message signaled interrupt destination register (Bank D) 5	R/W	0x0000_0001	25.3.6.7/25-83
0x5_1FC0	MSIVPRD6—Shared message signaled interrupt vector/priority register (Bank D) 6	R/W	0x8000_0000	25.3.6.6/25-82
0x5_1FD0	MSIDRD6—Shared message signaled interrupt destination register (Bank D) 6	R/W	0x0000_0001	25.3.6.7/25-83
0x5_1FE0	MSIVPRD7—Shared message signaled interrupt vector/priority register (Bank D) 7	R/W	0x8000_0000	25.3.6.6/25-82

Table 25-6. MPIC Register Address Map — Interrupt Source Configuration Registers

Offset	Register	Access	Reset	Section/Page
0x5_1FF0	MSIDRD7—Shared message signaled interrupt destination register (Bank D) 7	R/W	0x0000_0001	25.3.6.7/25-83
0x5_2000	MSIVPRA8—Shared message signaled interrupt vector/priority register (Bank A) 8	R/W	0x8000_0000	25.3.6.6/25-82
0x5_2010	MSIDRA8—Shared message signaled interrupt destination register (Bank A) 8	R/W	0x0000_0001	25.3.6.7/25-83
0x5_2020	MSIVPRA9—Shared message signaled interrupt vector/priority register (Bank A) 9	R/W	0x8000_0000	25.3.6.6/25-82
0x5_2030	MSIDRA9—Shared message signaled interrupt destination register (Bank A) 9	R/W	0x0000_0001	25.3.6.7/25-83
0x5_2040	MSIVPRA10—Shared message signaled interrupt vector/priority register (Bank A) 10	R/W	0x8000_0000	25.3.6.6/25-82
0x5_2050	MSIDRA10—Shared message signaled interrupt destination register (Bank A) 10	R/W	0x0000_0001	25.3.6.7/25-83
0x5_2060	MSIVPRA11—Shared message signaled interrupt vector/priority register (Bank A) 11	R/W	0x8000_0000	25.3.6.6/25-82
0x5_2070	MSIDRA11—Shared message signaled interrupt destination register (Bank A) 11	R/W	0x0000_0001	25.3.6.7/25-83
0x5_2080	MSIVPRA12—Shared message signaled interrupt vector/priority register (Bank A) 12	R/W	0x8000_0000	25.3.6.6/25-82
0x5_2090	MSIDRA12—Shared message signaled interrupt destination register (Bank A) 12	R/W	0x0000_0001	25.3.6.7/25-83
0x5_20A0	MSIVPRA13—Shared message signaled interrupt vector/priority register (Bank A) 13	R/W	0x8000_0000	25.3.6.6/25-82
0x5_20B0	MSIDRA13—Shared message signaled interrupt destination register (Bank A) 13	R/W	0x0000_0001	25.3.6.7/25-83
0x5_20C0	MSIVPRA14—Shared message signaled interrupt vector/priority register (Bank A) 14	R/W	0x8000_0000	25.3.6.6/25-82
0x5_20D0	MSIDRA14—Shared message signaled interrupt destination register (Bank A) 14	R/W	0x0000_0001	25.3.6.7/25-83
0x5_20E0	MSIVPRA15—Shared message signaled interrupt vector/priority register (Bank A) 15	R/W	0x8000_0000	25.3.6.6/25-82
0x5_20F0	MSIDRA15—Shared message signaled interrupt destination register (Bank A) 15	R/W	0x0000_0001	25.3.6.7/25-83
0x5_2100	MSIVPRB8—Shared message signaled interrupt vector/priority register (Bank B) 8	R/W	0x8000_0000	25.3.6.6/25-82
0x5_2110	MSIDRB8—Shared message signaled interrupt destination register (Bank B) 8	R/W	0x0000_0001	25.3.6.7/25-83
0x5_2120	MSIVPRB9—Shared message signaled interrupt vector/priority register (Bank B) 9	R/W	0x8000_0000	25.3.6.6/25-82

Table 25-6. MPIC Register Address Map — Interrupt Source Configuration Registers

Offset	Register	Access	Reset	Section/Page
0x5_2130	MSIDRB9—Shared message signaled interrupt destination register (Bank B) 9	R/W	0x0000_0001	25.3.6.7/25-83
0x5_2140	MSIVPRB10—Shared message signaled interrupt vector/priority register (Bank B) 10	R/W	0x8000_0000	25.3.6.6/25-82
0x5_2150	MSIDRB10—Shared message signaled interrupt destination register (Bank B) 10	R/W	0x0000_0001	25.3.6.7/25-83
0x5_2160	MSIVPRB11—Shared message signaled interrupt vector/priority register (Bank B) 11	R/W	0x8000_0000	25.3.6.6/25-82
0x5_2170	MSIDRB11—Shared message signaled interrupt destination register (Bank B) 11	R/W	0x0000_0001	25.3.6.7/25-83
0x5_2180	MSIVPRB12—Shared message signaled interrupt vector/priority register (Bank B) 12	R/W	0x8000_0000	25.3.6.6/25-82
0x5_2190	MSIDRB12—Shared message signaled interrupt destination register (Bank B) 12	R/W	0x0000_0001	25.3.6.7/25-83
0x5_21A0	MSIVPRB13—Shared message signaled interrupt vector/priority register (Bank B) 13	R/W	0x8000_0000	25.3.6.6/25-82
0x5_21B0	MSIDRB13—Shared message signaled interrupt destination register (Bank B) 13	R/W	0x0000_0001	25.3.6.7/25-83
0x5_21C0	MSIVPRB14—Shared message signaled interrupt vector/priority register (Bank B) 14	R/W	0x8000_0000	25.3.6.6/25-82
0x5_21D0	MSIDRB14—Shared message signaled interrupt destination register (Bank B) 14	R/W	0x0000_0001	25.3.6.7/25-83
0x5_21E0	MSIVPRB15—Shared message signaled interrupt vector/priority register (Bank B) 15	R/W	0x8000_0000	25.3.6.6/25-82
0x5_21F0	MSIDRB15—Shared message signaled interrupt destination register (Bank B) 15	R/W	0x0000_0001	25.3.6.7/25-83
0x5_2200	MSIVPRC8—Shared message signaled interrupt vector/priority register (Bank C) 8	R/W	0x8000_0000	25.3.6.6/25-82
0x5_2210	MSIDRC8—Shared message signaled interrupt destination register (Bank C) 8	R/W	0x0000_0001	25.3.6.7/25-83
0x5_2220	MSIVPRC9—Shared message signaled interrupt vector/priority register (Bank C) 9	R/W	0x8000_0000	25.3.6.6/25-82
0x5_2230	MSIDRC9—Shared message signaled interrupt destination register (Bank C) 9	R/W	0x0000_0001	25.3.6.7/25-83
0x5_2240	MSIVPRC10—Shared message signaled interrupt vector/priority register (Bank C) 10	R/W	0x8000_0000	25.3.6.6/25-82
0x5_2250	MSIDRC10—Shared message signaled interrupt destination register (Bank C) 10	R/W	0x0000_0001	25.3.6.7/25-83
0x5_2260	MSIVPRC11—Shared message signaled interrupt vector/priority register (Bank C) 11	R/W	0x8000_0000	25.3.6.6/25-82

Table 25-6. MPIC Register Address Map — Interrupt Source Configuration Registers

Offset	Register	Access	Reset	Section/Page
0x5_2270	MSIDRC11—Shared message signaled interrupt destination register (Bank C) 11	R/W	0x0000_0001	25.3.6.7/25-83
0x5_2280	MSIVPRC12—Shared message signaled interrupt vector/priority register (Bank C) 12	R/W	0x8000_0000	25.3.6.6/25-82
0x5_2290	MSIDRC12—Shared message signaled interrupt destination register (Bank C) 12	R/W	0x0000_0001	25.3.6.7/25-83
0x5_22A0	MSIVPRC13—Shared message signaled interrupt vector/priority register (Bank C) 13	R/W	0x8000_0000	25.3.6.6/25-82
0x5_22B0	MSIDRC13—Shared message signaled interrupt destination register (Bank C) 13	R/W	0x0000_0001	25.3.6.7/25-83
0x5_22C0	MSIVPRC14—Shared message signaled interrupt vector/priority register (Bank C) 14	R/W	0x8000_0000	25.3.6.6/25-82
0x5_22D0	MSIDRC14—Shared message signaled interrupt destination register (Bank C) 14	R/W	0x0000_0001	25.3.6.7/25-83
0x5_22E0	MSIVPRC15—Shared message signaled interrupt vector/priority register (Bank C) 15	R/W	0x8000_0000	25.3.6.6/25-82
0x5_22F0	MSIDRC15—Shared message signaled interrupt destination register (Bank C) 15	R/W	0x0000_0001	25.3.6.7/25-83
0x5_2300	MSIVPRD8—Shared message signaled interrupt vector/priority register (Bank D) 8	R/W	0x8000_0000	25.3.6.6/25-82
0x5_2310	MSIDRD8—Shared message signaled interrupt destination register (Bank D) 8	R/W	0x0000_0001	25.3.6.7/25-83
0x5_2320	MSIVPRD9—Shared message signaled interrupt vector/priority register (Bank D) 9	R/W	0x8000_0000	25.3.6.6/25-82
0x5_2330	MSIDRD9—Shared message signaled interrupt destination register (Bank D) 9	R/W	0x0000_0001	25.3.6.7/25-83
0x5_2340	MSIVPRD10—Shared message signaled interrupt vector/priority register (Bank D) 10	R/W	0x8000_0000	25.3.6.6/25-82
0x5_2350	MSIDRD10—Shared message signaled interrupt destination register (Bank D) 10	R/W	0x0000_0001	25.3.6.7/25-83
0x5_2360	MSIVPRD11—Shared message signaled interrupt vector/priority register (Bank D) 11	R/W	0x8000_0000	25.3.6.6/25-82
0x5_2370	MSIDRD11—Shared message signaled interrupt destination register (Bank D) 11	R/W	0x0000_0001	25.3.6.7/25-83
0x5_2380	MSIVPRD12—Shared message signaled interrupt vector/priority register (Bank D) 12	R/W	0x8000_0000	25.3.6.6/25-82
0x5_2390	MSIDRD12—Shared message signaled interrupt destination register (Bank D) 12	R/W	0x0000_0001	25.3.6.7/25-83
0x5_23A0	MSIVPRD13—Shared message signaled interrupt vector/priority register (Bank D) 13	R/W	0x8000_0000	25.3.6.6/25-82

Table 25-6. MPIC Register Address Map — Interrupt Source Configuration Registers

Offset	Register	Access	Reset	Section/Page
0x5_23B0	MSIDRD13—Shared message signaled interrupt destination register (Bank D) 13	R/W	0x0000_0001	25.3.6.7/25-83
0x5_23C0	MSIVPRD14—Shared message signaled interrupt vector/priority register (Bank D) 14	R/W	0x8000_0000	25.3.6.6/25-82
0x5_23D0	MSIDRD14—Shared message signaled interrupt destination register (Bank D) 14	R/W	0x0000_0001	25.3.6.7/25-83
0x5_23E0	MSIVPRD15—Shared message signaled interrupt vector/priority register (Bank D) 15	R/W	0x8000_0000	25.3.6.6/25-82
0x5_23F0	MSIDRD15—Shared message signaled interrupt destination register (Bank D) 15	R/W	0x0000_0001	25.3.6.7/25-83
0x5_2400–0x5_2FF0	Reserved for additional interrupt sources	—	—	—
Internal Interrupt Configuration Registers				
0x5_3000	IIVPR160—Internal interrupt 160 vector/priority register	R/W	0x8080_0000	25.3.7.4/25-88
0x5_3010	IIDR160—Internal interrupt 160 destination register	R/W	0x0000_0001	25.3.7.5/25-89
0x5_3018	IILR160—Internal interrupt 160 level register	R/W	All zeros	25.3.7.6/25-90
0x5_3020	IIVPR161—Internal interrupt 161 vector/priority register	R/W	0x8080_0000	25.3.7.4/25-88
0x5_3030	IIDR161—Internal interrupt 161 destination register	R/W	0x0000_0001	25.3.7.5/25-89
0x5_3038	IILR161—Internal interrupt 161 level register	R/W	All zeros	25.3.7.6/25-90
0x5_3040	IIVPR162—Internal interrupt 162 vector/priority register	R/W	0x8080_0000	25.3.7.4/25-88
0x5_3050	IIDR162—Internal interrupt 162 destination register	R/W	0x0000_0001	25.3.7.5/25-89
0x5_3058	IILR162—Internal interrupt 162 level register	R/W	All zeros	25.3.7.6/25-90
0x5_3060	IIVPR163—Internal interrupt 163 vector/priority register	R/W	0x8080_0000	25.3.7.4/25-88
0x5_3070	IIDR163—Internal interrupt 163 destination register	R/W	0x0000_0001	25.3.7.5/25-89
0x5_3078	IILR163—Internal interrupt 163 level register	R/W	All zeros	25.3.7.6/25-90
0x5_3080	IIVPR164—Internal interrupt 164 vector/priority register	R/W	0x8080_0000	25.3.7.4/25-88
0x5_3090	IIDR164—Internal interrupt 164 destination register	R/W	0x0000_0001	25.3.7.5/25-89
0x5_3098	IILR164—Internal interrupt 164 level register	R/W	All zeros	25.3.7.6/25-90
0x5_30A0	IIVPR165—Internal interrupt 165 vector/priority register	R/W	0x8080_0000	25.3.7.4/25-88
0x5_30B0	IIDR165—Internal interrupt 165 destination register	R/W	0x0000_0001	25.3.7.5/25-89
0x5_30B8	IILR165—Internal interrupt 165 level register	R/W	All zeros	25.3.7.6/25-90
0x5_30C0	IIVPR166—Internal interrupt 166 vector/priority register	R/W	0x8080_0000	25.3.7.4/25-88
0x5_30D0	IIDR166—Internal interrupt 166 destination register	R/W	0x0000_0001	25.3.7.5/25-89
0x5_30D8	IILR166—Internal interrupt 166 level register	R/W	All zeros	25.3.7.6/25-90
0x5_30E0	IIVPR167—Internal interrupt 167 vector/priority register	R/W	0x8080_0000	25.3.7.4/25-88

Table 25-6. MPIC Register Address Map — Interrupt Source Configuration Registers

Offset	Register	Access	Reset	Section/Page
0x5_30F0	IIDR167—Internal interrupt 167 destination register	R/W	0x0000_0001	25.3.7.5/25-89
0x5_30F8	IILR167—Internal interrupt 167 level register	R/W	All zeros	25.3.7.6/25-90
0x5_3100	IIVPR168—Internal interrupt 168 vector/priority register	R/W	0x8080_0000	25.3.7.4/25-88
0x5_3110	IIDR168—Internal interrupt 168 destination register	R/W	0x0000_0001	25.3.7.5/25-89
0x5_3118	IILR168—Internal interrupt 168 level register	R/W	All zeros	25.3.7.6/25-90
0x5_3120	IIVPR169—Internal interrupt 169 vector/priority register	R/W	0x8080_0000	25.3.7.4/25-88
0x5_3130	IIDR169—Internal interrupt 169 destination register	R/W	0x0000_0001	25.3.7.5/25-89
0x5_3138	IILR169—Internal interrupt 169 level register	R/W	All zeros	25.3.7.6/25-90
0x5_3140	IIVPR170—Internal interrupt 170 vector/priority register	R/W	0x8080_0000	25.3.7.4/25-88
0x5_3150	IIDR170—Internal interrupt 170 destination register	R/W	0x0000_0001	25.3.7.5/25-89
0x5_3158	IILR170—Internal interrupt 170 level register	R/W	All zeros	25.3.7.6/25-90
0x5_3160	IIVPR171—Internal interrupt 171 vector/priority register	R/W	0x8080_0000	25.3.7.4/25-88
0x5_3170	IIDR171—Internal interrupt 171 destination register	R/W	0x0000_0001	25.3.7.5/25-89
0x5_3178	IILR171—Internal interrupt 171 level register	R/W	All zeros	25.3.7.6/25-90
0x5_3180	IIVPR172—Internal interrupt 172 vector/priority register	R/W	0x8080_0000	25.3.7.4/25-88
0x5_3190	IIDR172—Internal interrupt 172 destination register	R/W	0x0000_0001	25.3.7.5/25-89
0x5_3198	IILR172—Internal interrupt 172 level register	R/W	All zeros	25.3.7.6/25-90
0x5_31A0	IIVPR173—Internal interrupt 173 vector/priority register	R/W	0x8080_0000	25.3.7.4/25-88
0x5_31B0	IIDR173—Internal interrupt 173 destination register	R/W	0x0000_0001	25.3.7.5/25-89
0x5_31B8	IILR173—Internal interrupt 173 level register	R/W	All zeros	25.3.7.6/25-90
0x5_31C0	IIVPR174—Internal interrupt 174 vector/priority register	R/W	0x8080_0000	25.3.7.4/25-88
0x5_31D0	IIDR174—Internal interrupt 174 destination register	R/W	0x0000_0001	25.3.7.5/25-89
0x5_31D8	IILR174—Internal interrupt 174 level register	R/W	All zeros	25.3.7.6/25-90
0x5_31E0	IIVPR175—Internal interrupt 175 vector/priority register	R/W	0x8080_0000	25.3.7.4/25-88
0x5_31F0	IIDR175—Internal interrupt 175 destination register	R/W	0x0000_0001	25.3.7.5/25-89
0x5_31F8	IILR175—Internal interrupt 175 level register	R/W	All zeros	25.3.7.6/25-90
005_3200–0x5_3BE0	IIVPR176–255—Internal interrupt 176-255 vector/priority register	R/W	0x8080_0000	25.3.7.4/25-88
0x5_3210–0x5_3BF0	IIDR176–255—Internal interrupt 176-255 destination register	R/W	0x0000_0001	25.3.7.5/25-89
0x5_3218–0x5_3BF8	IILR176–255—Internal interrupt 176-255 level register	R/W	All zeros	25.3.7.6/25-90
0x5_3C00 – 0x5_FFF0	Reserved for additional interrupt sources	—	—	—

Table 25-7 shows the Per-CPU global access MPIC registers.

Table 25-7. MPIC Per-CPU Global Access Register Map

Offset	Register	Access	Reset	Section/Page
Core 0 Per-CPU Registers (global access)				
0x6_0000–0x6_0030	Reserved (for core 0 test)	—	—	—
0x6_0040	IPIDR0—Processor core 0 interprocessor 0 dispatch register	W	All zeros	25.3.8.1/25-93
0x6_0050	IPIDR1—Processor core 0 interprocessor 1 dispatch register	W	All zeros	25.3.8.1/25-93
0x6_0060	IPIDR2—Processor core 0 interprocessor 2 dispatch register	W	All zeros	25.3.8.1/25-93
0x6_0070	IPIDR3—Processor core 0 interprocessor 3 dispatch register	W	All zeros	25.3.8.1/25-93
0x6_0080	CTPR0—Processor core 0 current task priority register	R/W	0x0000_000F	25.3.8.2/25-95
0x6_0090	WHOAMI0—Processor core 0 who am I register	R	n/a	25.3.8.3/25-96
0x6_00A0	IACK0—Processor core 0 interrupt acknowledge register	R	All zeros	25.3.8.4/25-96
0x6_00B0	EOI0—Processor core 0 end of interrupt register	W	All zeros	25.3.8.5/25-97
0x6_00C0–0x6_0FF0	Reserved for Processor core 0	—	—	—
Core 1 Per-CPU Registers (global access)				
0x6_1000–0x6_1030	Reserved (for core 1 test)	—	—	—
0x6_1040	IPIDR0—Processor core 1 interprocessor 0 dispatch register [1]	W	All zeros	25.3.8.1/25-93
0x6_1050	IPIDR1—Processor core 1 interprocessor 1 dispatch register [1]	W	All zeros	25.3.8.1/25-93
0x6_1060	IPIDR2—Processor core 1 interprocessor 2 dispatch register [1]	W	All zeros	25.3.8.1/25-93
0x6_1070	IPIDR3—Processor core 1 interprocessor 3 dispatch register [1]	W	All zeros	25.3.8.1/25-93
0x6_1080	CTPR1—Processor core 1 current task priority register [1]	R/W	0x0000_000F	25.3.8.2/25-95
0x6_1090	WHOAMI1—Processor core 1 who am I register [1]	R	n/a	25.3.8.3/25-96
0x6_10A0	IACK1—Processor core 1 interrupt acknowledge register [1]	R	All zeros	25.3.8.4/25-96
0x6_10B0	EOI1—Processor core 1 end of interrupt register [1]	W	All zeros	25.3.8.5/25-97
0x6_10C0–0x6_1FF0	Reserved for Processor core 1	—	—	—
Core 2 Per-CPU Registers (global access)				
0x6_2000–0x6_2030	Reserved (for core 2 test)	—	—	—
0x6_2040	IPIDR0—Processor core 2 interprocessor 0 dispatch register [1]	W	All zeros	25.3.8.1/25-93
0x6_2050	IPIDR1—Processor core 2 interprocessor 1 dispatch register [1]	W	All zeros	25.3.8.1/25-93
0x6_2060	IPIDR2—Processor core 2 interprocessor 2 dispatch register [1]	W	All zeros	25.3.8.1/25-93
0x6_2070	IPIDR3—Processor core 2 interprocessor 3 dispatch register [1]	W	All zeros	25.3.8.1/25-93

Table 25-7. MPIC Per-CPU Global Access Register Map

Offset	Register	Access	Reset	Section/Page
0x6_2080	CTPR2—Processor core 2 current task priority register [1]	R/W	0x0000_000F	25.3.8.2/25-95
0x6_2090	WHOAMI2—Processor core 2 who am I register [1]	R	n/a	25.3.8.3/25-96
0x6_20A0	IACK2—Processor core 2 interrupt acknowledge register [1]	R	All zeros	25.3.8.4/25-96
0x6_20B0	EOI2—Processor core 2 end of interrupt register [1]	W	All zeros	25.3.8.5/25-97
0x6_20C0–0x6_2FF0	Reserved for Processor core 2	—	—	—
Core 3 Per-CPU Registers (global access)				
0x6_3000–0x6_3030	Reserved (for core 3 test)	—	—	—
0x6_3040	IPIDR0—Processor core 3 interprocessor 0 dispatch register [1]	W	All zeros	25.3.8.1/25-93
0x6_3050	IPIDR1—Processor core 3 interprocessor 1 dispatch register [1]	W	All zeros	25.3.8.1/25-93
0x6_3060	IPIDR2—Processor core 3 interprocessor 2 dispatch register [1]	W	All zeros	25.3.8.1/25-93
0x6_3070	IPIDR3—Processor core 3 interprocessor 3 dispatch register [1]	W	All zeros	25.3.8.1/25-93
0x6_3080	CTPR3—Processor core 3 current task priority register [1]	R/W	0x0000_000F	25.3.8.2/25-95
0x6_3090	WHOAMI3—Processor core 3 who am I register [1]	R	n/a	25.3.8.3/25-96
0x6_30A0	IACK3—Processor core 3 interrupt acknowledge register [1]	R	All zeros	25.3.8.4/25-96
0x6_30B0	EOI3—Processor core 3 end of interrupt register [1]	W	All zeros	25.3.8.5/25-97
0x6_30C0–0x6_3FF0	Reserved for Processor core 3	—	—	—
Core 4 Per-CPU Registers (global access)				
0x6_4000–0x6_4030	Reserved (for core 4 test)	—	—	—
0x6_4040	IPIDR0—Processor core 4 interprocessor 0 dispatch register [1]	W	All zeros	25.3.8.1/25-93
0x6_4050	IPIDR1—Processor core 4 interprocessor 1 dispatch register [1]	W	All zeros	25.3.8.1/25-93
0x6_4060	IPIDR2—Processor core 4 interprocessor 2 dispatch register [1]	W	All zeros	25.3.8.1/25-93
0x6_4070	IPIDR3—Processor core 4 interprocessor 3 dispatch register [1]	W	All zeros	25.3.8.1/25-93
0x6_4080	CTPR4—Processor core 4 current task priority register [1]	R/W	0x0000_000F	25.3.8.2/25-95
0x6_4090	WHOAMI4—Processor core 4 who am I register [1]	R	n/a	25.3.8.3/25-96
0x6_40A0	IACK4—Processor core 4 interrupt acknowledge register [1]	R	all zeros	25.3.8.4/25-96
0x6_40B0	EOI4—Processor core 4 end of interrupt register [1]	W	all zeros	25.3.8.5/25-97
0x6_40C0–0x6_4FF0	Reserved for Processor core 4	—	—	—
Core 5 Per-CPU Registers (global access)				

Table 25-7. MPIC Per-CPU Global Access Register Map

Offset	Register	Access	Reset	Section/Page
0x6_5000–0x6_5030	Reserved (for core 5 test)	—	—	—
0x6_5040	IPIDR0—Processor core 5 interprocessor 0 dispatch register [2]	W	All zeros	25.3.8.1/25-93
0x6_5050	IPIDR1—Processor core 5 interprocessor 1 dispatch register [2]	W	All zeros	25.3.8.1/25-93
0x6_5060	IPIDR2—Processor core 5 interprocessor 2 dispatch register [2]	W	All zeros	25.3.8.1/25-93
0x6_5070	IPIDR3—Processor core 5 interprocessor 3 dispatch register [2]	W	All zeros	25.3.8.1/25-93
0x6_5080	CTPR5—Processor core 5 current task priority register [2]	R/W	0x0000_000F	25.3.8.2/25-95
0x6_5090	WHOAMI5—Processor core 5 who am I register [2]	R	n/a	25.3.8.3/25-96
0x6_50A0	IACK5—Processor core 5 interrupt acknowledge register [2]	R	All zeros	25.3.8.4/25-96
0x6_50B0	EOI5—Processor core 5 end of interrupt register [2]	W	All zeros	25.3.8.5/25-97
0x6_50C0-0x6_5FF0	Reserved for Processor core 5	—	—	—
Core 6 Per-CPU Registers (global access)				
0x6_6000-0x6_6030	Reserved (for core 6 test)	—	—	—
0x6_6040	IPIDR0—Processor core 6 interprocessor 0 dispatch register [3]	W	All zeros	25.3.8.1/25-93
0x6_6050	IPIDR1—Processor core 6 interprocessor 1 dispatch register [3]	W	All zeros	25.3.8.1/25-93
0x6_6060	IPIDR2—Processor core 6 interprocessor 2 dispatch register [3]	W	All zeros	25.3.8.1/25-93
0x6_6070	IPIDR3—Processor core 6 interprocessor 3 dispatch register [3]	W	All zeros	25.3.8.1/25-93
0x6_6080	CTPR6—Processor core 6 current task priority register [3]	R/W	0x0000_000F	25.3.8.2/25-95
0x6_6090	WHOAMI6—Processor core 6 who am I register [3]	R	n/a	25.3.8.3/25-96
0x6_60A0	IACK6—Processor core 6 interrupt acknowledge register [3]	R	All zeros	25.3.8.4/25-96
0x6_60B0	EOI6—Processor core 6 end of interrupt register [3]	W	All zeros	25.3.8.5/25-97
0x6_60C0–0x6_6FF0	Reserved for Processor core 6	—	—	—
Core 7 Per-CPU Registers (global access)				
0x6_7000–0x6_7030	Reserved (for core 7 test)	—	—	—
0x6_7040	IPIDR0—Processor core 7 interprocessor 0 dispatch register [4]	W	All zeros	25.3.8.1/25-93
0x6_7050	IPIDR1—Processor core 7 interprocessor 1 dispatch register [4]	W	All zeros	25.3.8.1/25-93
0x6_7060	IPIDR2—Processor core 7 interprocessor 2 dispatch register [4]	W	All zeros	25.3.8.1/25-93
0x6_7070	IPIDR3—Processor core 7 interprocessor 3 dispatch register [4]	W	All zeros	25.3.8.1/25-93
0x6_7080	CTPR7—Processor core 7 current task priority register [4]	R/W	0x0000_000F	25.3.8.2/25-95
0x6_7090	WHOAMI7—Processor core 7 who am I register [4]	R	n/a	25.3.8.3/25-96

Table 25-7. MPIC Per-CPU Global Access Register Map

Offset	Register	Access	Reset	Section/Page
0x6_70A0	IACK7—Processor core 7 interrupt acknowledge register [4]	R	All zeros	25.3.8.4/25-96
0x6_70B0	EOI7—Processor core 7 end of interrupt register [4]	W	All zeros	25.3.8.5/25-97
0x6_70C0–0x6_7FF0	Reserved for Processor core 7	—	—	—

[1] These registers are not supported in single-processor implementations.
[2] These registers are not supported in single-processor implementations.
[3] These registers are not supported in single-processor implementations.
[4] These registers are not supported in single-processor implementations.

25.3.1 Global Registers

Although most MPIC registers have one address, some are replicated for each processor core in a multiprocessor device. For such registers, each core accesses its separate registers using the same address, the address decoding being sensitive to the processor core. A copy of the per-CPU registers is available to each processor core at the same physical address, that is, in a private access address space that acts like an alias to a processor's own copy of the per-CPU registers. The ID of the core initiating the read/write transaction determines which processor's per-CPU registers to access. For more information, see Section 25.3.8, "Per-CPU (Private Access) Registers."

NOTE

Register fields designated as write-1-to-clear are cleared only by writing ones to them. Writing zeros to them has no effect.

25.3.1.1 Block Revision Register 1 (BRR1)

BRR1, shown in Figure 25-3, provides information about the MPIC IP block.

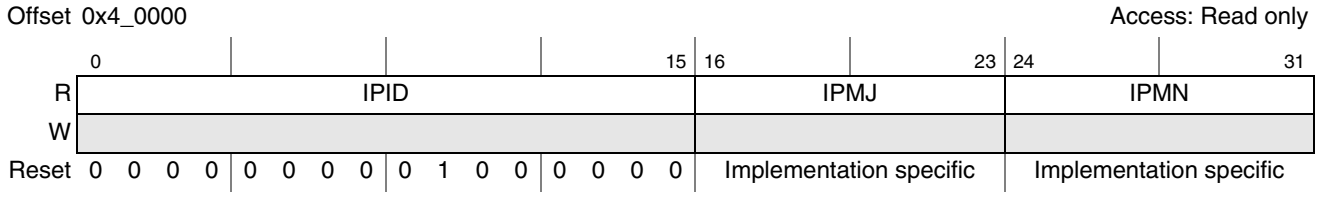


Figure 25-3. Block Revision Register 1 (BRR1)

Table 25-10 describes the BRR1 fields.

Table 25-8. BRR1 Field Descriptions

Bits	Name	Description
0–15	IPID	IP block ID.
16–23	IPMJ	The major revision of the IP block.
24–31	IPMN	The minor revision of the IP block.

25.3.1.2 Block Revision Register 2 (BRR2)

BRR2, shown in Figure 25-4, provides information about the IP block integration option and IP block configuration options.

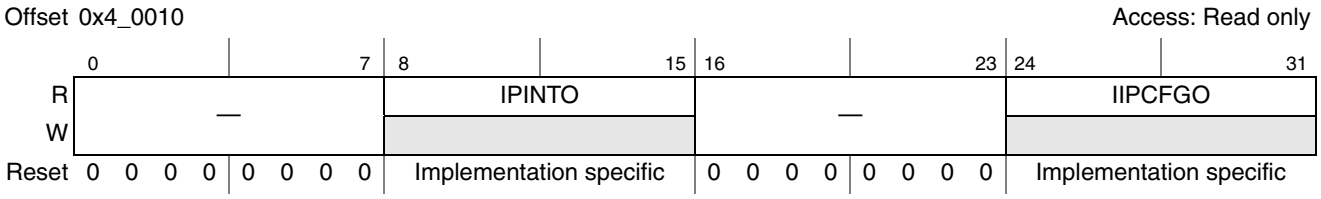


Figure 25-4. Block Revision Register 2 (BRR2)

Table 25-9 describes the BRR2 fields.

Table 25-9. BRR2 Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8–15	IPINTO	IP block integration options
16–23	—	Reserved
24–31	IPCFGO	IP block configuration options

25.3.1.3 Feature Reporting Register (FRR)

FRR, shown in Figure 25-5, provides information about interrupt and processor core configurations. It also informs the programming environment of the controller version.

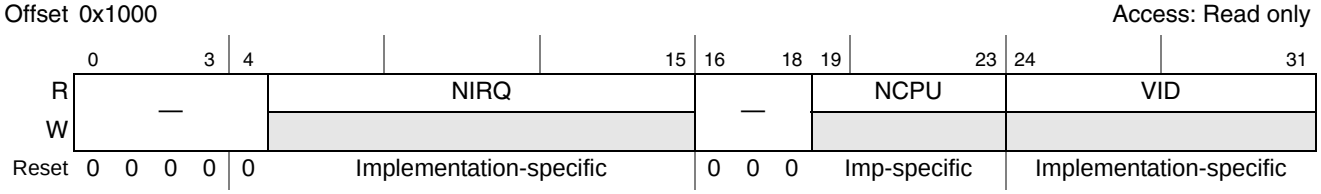


Figure 25-5. Feature Reporting Register (FRR)

Table 25-10 describes the FRR fields.

Table 25-10. FRR Field Descriptions

Bits	Name	Description
0-4	—	Reserved

5–15	NIRQ	Number of interrupts. Holds the binary value of the number of the highest interrupt source supported minus one.
		Equation: #internal + #external + MSGA + MSGB + MSIA + MSIB + MSIC + MSID + #(IPI * cores) + #timers
		256(internal) + 12(external) + 16(MSGA&B) + 64(MSIA&B&C&D) + 32(IPI4*8cores) + 8(timersA&B) = 0x183 (total-1)

16–18	—	Reserved

Table 25-10. FRR Field Descriptions (continued)

Bits	Name	Description
19–23	NCPU	Number of CPUs. The number of virtual CPUs (or threads) supported minus one. **Note:** The OpenPIC standard specifies "physical" cores, but this chip uses "virtual CPUs" for NCPU. 00000 Single core—core0 00001 Two cores—core0 and core1 (same continues all the way to 00111)
24–31	VID	Version ID. Reports the OpenPIC specification revision level supported by this interrupt controller implementation. The VID field's value of two (0x02) corresponds to revision 1.2 which is the revision level currently supported.

25.3.1.4 Global Configuration Register (GCR)

GCR, shown in Figure 25-6, controls the MPIC's operating mode, and allows software to reset the MPIC.

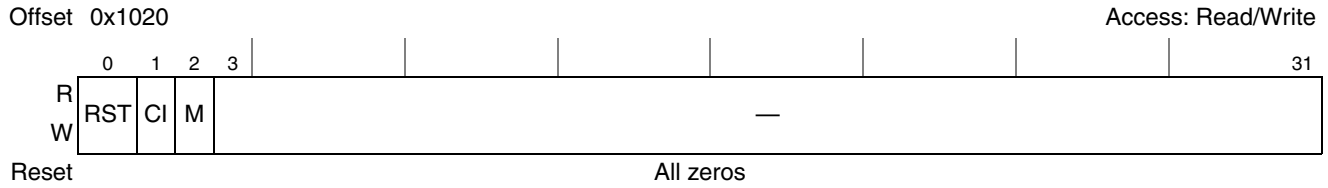


Figure 25-6. Global Configuration Register (GCR)

Table 25-11 describes the GCR fields.

Table 25-11. GCR Field Descriptions

Bits	Name	Description
0	RST	Reset. Setting RST forces the MPIC to be reset. Cleared automatically when the reset sequence is complete. See Section 25.4.7, "Resetting the MPIC," for more information.
1	CI	External proxy facility. Controls the method of core interrupt delivery 0 Legacy mode. Interrupts are delivered to the cores using the legacy OpenPIC model. 1 External proxy facilitymode. Interrupts are delivered to the cores using the external proxy facility model. Note: When Global configuration, GCR[M] = 0 (when the MPIC is in Pass-through mode), this field is a don't care.
2	M	Mode. MPIC operating mode. Section 25.1.4, "Modes of Operation," provides details about these modes. 0 Pass-through mode. On-chip MPIC is disabled and interrupts detected on IRQ0 are passed directly to core 0. 1 Mixed mode. Interrupts are handled by the normal priority and delivery mechanisms of the MPIC.
3–31	—	Reserved

25.3.1.5 Global Configuration Register 1 (GCR1)

GCR1, shown in Figure 25-7, controls the MPIC's operating mode, and allows software to reset the MPIC.

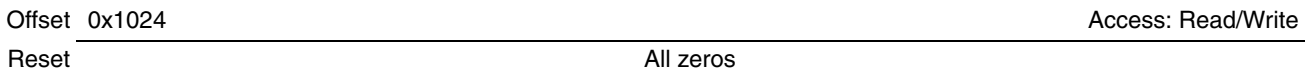


Figure 25-7. Global Configuration Register 1 (GCR1)

Table 25-12 describes the GCR1 fields.

25.3.1.6 Vendor Identification Register (VIR)

VIR, shown in Figure 25-8, is defined by the OpenPIC specifications and is provided for compliance. The zero value for VIR[VENDORID] indicates a generic OpenPIC-compliant device, which makes the other VIR fields meaningless.

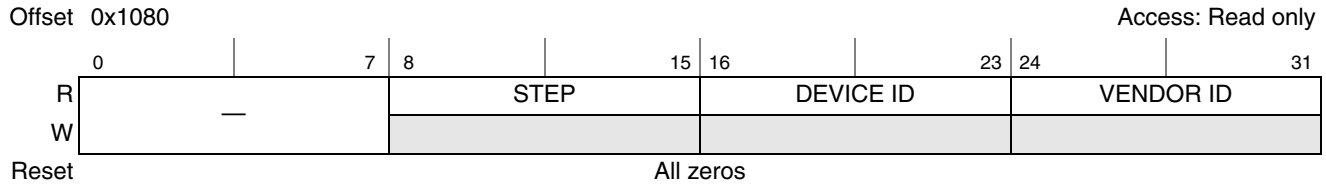


Figure 25-8. Vendor Identification Register (VIR)

Table 25-13 describes the VIR fields.

Table 25-13. VIR Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8–15	STEP	Stepping. Indicates the silicon revision for this device. Has no meaning if VENDOR ID value is zero.
16–23	DEVICE ID	Device identification. Vendor-specified identifier for this device. Has no meaning if VENDOR ID is zero.
24–31	VENDOR ID	Vendor identification. Specifies the manufacturer of this part. A value of zero implies a generic OpenPIC-compliant device.

25.3.1.7 Processor Core Initialization Register (PIR)

PIR, shown in Figure 25-9, provides a way for software to initiate a reset to the processor cores. Setting P*n* causes the respective signal to assert. These signals are routed to the appropriate *coren_hreset* signals and cause the corresponding core to redirect program flow to the reset interrupt vector.

For proper system operation, a core should be reset in this way only if the core is already in nap or sleep state. Because a core in either state cannot perform the necessary write to cause a hard reset, a core cannot put itself into hard reset.

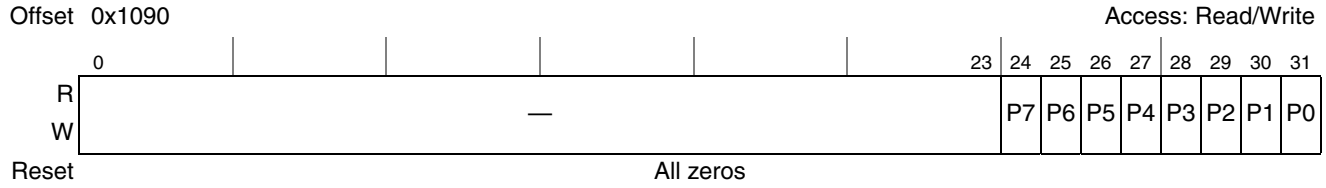


Figure 25-9. Processor Core Initialization Register (PIR)

Table 25-14 describes the PIR fields.

Table 25-14. PIR Field Descriptions

Bits	Name	Description
0–17	—	Reserved
24	P7	Processor core 7 reset. Setting this bit causes the MPIC to assert the *pic_core_hard_reset* signal to core 7.
25	P6	Processor core 6 reset. Setting this bit causes the MPIC to assert the *pic_core_hard_reset* signal to core 6.
26	P5	Processor core 5 reset. Setting this bit causes the MPIC to assert the *pic_core_hard_reset* signal to core 5.
27	P4	Processor core 4 reset. Setting this bit causes the MPIC to assert the *pic_core_hard_reset* signal to core 4.
28	P3	Processor core 3 reset. Setting this bit causes the MPIC to assert the *pic_core_hard_reset* signal to core 3.
29	P2	Processor core 2 reset. Setting this bit causes the MPIC to assert the *pic_core_hard_reset* signal to core 2.
30	P1	Processor core 1 reset. Setting this bit causes the MPIC to assert the *pic_core_hard_reset* signal to core 1.
31	P0	Processor core 0 reset. Setting this bit causes the MPIC to assert the *pic_core_hard_reset* signal to core 0.

25.3.1.8 Processor NMI Register (PNMIR)

The PNMIR provides a way for software on another device to signal a non-maskable interrupt event to the processor cores by writing to a MPIC register. A *coren_nmi* signal to the cores is asserted when a one is written to the corresponding PNMIR field; these signals are held active until a zero is written to those same bits. This deasserts the *core_nmi* input of the corresponding core.

The fields corresponding to processors on a device are shown in Figure 25-10.

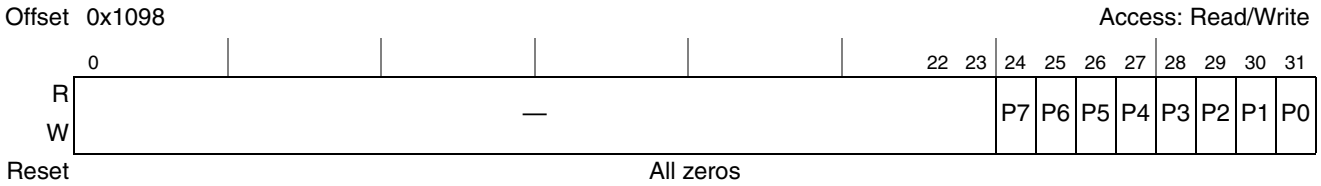


Figure 25-10. Processor NMI Register (PNMIR)

Table 25-15 describes the PNMIR fields.

Table 25-15. PNMIR Field Descriptions

Bits	Name	Description
0–23	—	Reserved
24	P7	Processor core 7 NMI. Setting P7 causes the MPIC to assert the *pic_core_nmi* signal to core 7.
25	P6	Processor core 6 NMI. Setting P6 causes the MPIC to assert the *pic_core_nmi* signal to core 6.
26	P5	Processor core 5 NMI. Setting P5 causes the MPIC to assert the *pic_core_nmi* signal to core 5.
27	P4	Processor core 4 NMI. Setting P4 causes the MPIC to assert the *pic_core_nmi* signal to core 4.
28	P3	Processor core 3 NMI. Setting P3 causes the MPIC to assert the *pic_core_nmi* signal to core 3.
29	P2	Processor core 2 NMI. Setting P2 causes the MPIC to assert the *pic_core_nmi* signal to core 2.

Table 25-15. PNMIR Field Descriptions (continued)

Bits	Name	Description
30	P1	Processor core 1 NMI. Setting P1 causes the MPIC to assert the *pic_core_nmi* signal to core 1.
31	P0	Processor core 0 NMI. Setting P0 causes the MPIC to assert the *pic_core_nmi* signal to core 0.

25.3.1.9 Interprocessor Interrupt Vector/Priority Registers (IPIVPR0–IPIVPR3)

IPIVPRs, shown in Figure 25-11, contain the interrupt vector and priority fields for the four interprocessor interrupt channels. There is one vector/priority register per channel. The VECTOR and PRIORITY values should not be changed while IPIVPR*n*[A] is set.

Offset IPIVPR0: 0x10A0; IPIVPR1: 0x10B0; IPIVPR2: 0x10C0; IPIVPR3: 0x10D0 Access: Read/Write

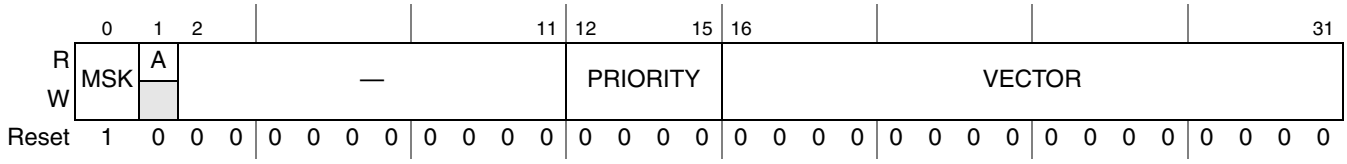


Figure 25-11. Interprocessor Interrupt Vector/Priority Register (IPIVPR*n*)

Table 25-16 describes the IPIVPR*n* fields.

Table 25-16. IPIVPR*n* Field Descriptions

Bits	Name	Description
0	MSK	Mask. Mask interrupts from this source. MSK affects interrupts routed to *int*, *cint*, *mcp*, *sie*[0:2], and IRQ_OUT. 0 An interrupt request is generated if the corresponding IPR bit is set. 1 Further interrupts from this source are disabled.
1	A	Activity. Indicates an interrupt has been requested or is in service. The VECTOR and PRIORITY values should not be changed while this bit is set. Affects only interrupts routed to *int*. 0 No current interrupt activity associated with this source. 1 The interrupt field for this source is set in the IPR or ISR.
2–11	—	Reserved
12–15	PRIORITY	Priority. Specifies the interrupt priority. The lowest priority is 0 and the highest priority is 15. A priority level of 0 inhibits signalling of this interrupt to the core. Affects only interrupts routed to *int*.
16–31	VECTOR	Vector (Affects only interrupts routed to *int*). Contains the value returned when IACK is read and this interrupt resides in the corresponding interrupt request register (IRR) for that core, as shown in Figure 25-86.

25.3.1.10 Spurious Vector Register (SVR)

SVR, shown in Figure 25-12, contains the 16-bit vector returned to the processor core when the corresponding IACK register is read for a spurious interrupt.

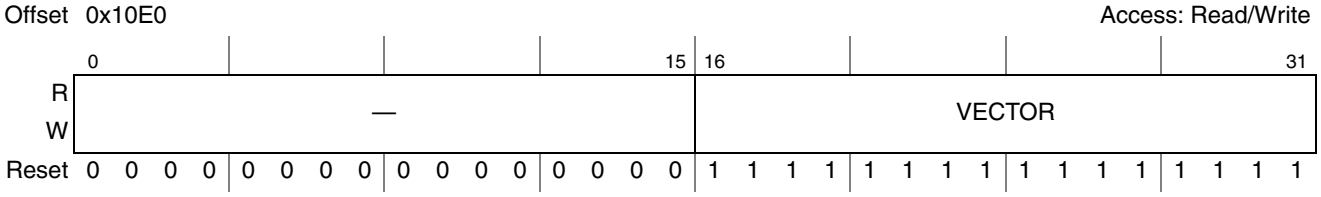


Figure 25-12. Spurious Vector Register (SVR)

Table 25-17 describes the SVR fields.

Table 25-17. SVR Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	VECTOR	Spurious interrupt vector. Value returned when IACK is read during a spurious vector fetch. Section 25.4.1.2.4, "Spurious Vector Generation," gives information about the conditions that may cause a spurious vector fetch.

25.3.2 Global Timer Registers

The two independent groups of global timer registers, group A and group B, are identical in their functionality, except that they appear at different locations within the MPIC register map. Note that each of the four timers within an *x* group have four individual configuration registers (GTCCR*xn*, GTBCR*xn*, GTVPR*xn*, GTDR*xn*), but they are only shown once in this section. These two groups of timers cannot be cascaded together.

25.3.2.1 Timer Frequency Reporting Register (TFRRA–TFRRB)

The TFRRs, shown in Figure 25-13, are written by software to report the clocking frequency of the MPIC timers. Note that although TFRRs are read/write, the MPIC ignores the register values.

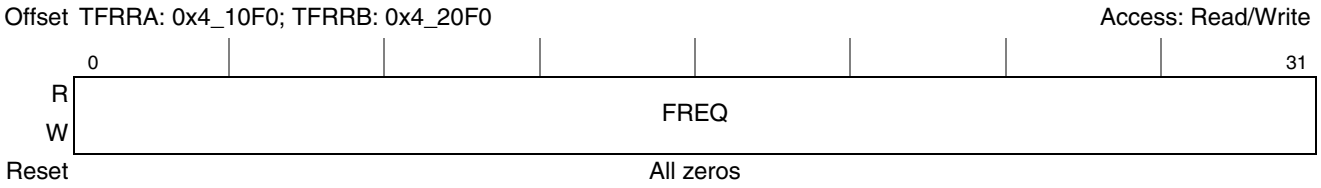


Figure 25-13. Timer Frequency Reporting Registers (TFRR*x*)

Table 25-18 describes the TFRR*x* registers.

Table 25-18. TFRR*x* Field Descriptions

Bits	Name	Description
0–31	FREQ	Timer frequency (in ticks/second (Hz)). Used to communicate the frequency of the global timers' clock source,—either the MPIC input clock or the frequency of the RTC signal—to user software. TFRR*x* is set only by software for later use by other applications and its value in no way affects the operating frequency of the global timers. The timers operate at a ratio of this clock frequency, as set by TCR*x*[CLKR]. See Section 25.3.2.6, "Timer Control Registers (TCRA–TCRB)."

25.3.2.2 Global Timer Current Count Registers (GTCCRA0–GTCCRA3, GTCCRB0–GTCCRB3)

The GTCCRs, shown in Figure 25-14, contain the current count for each of the four MPIC timers in each of the two groups.

Offset Group A: GTCCRA0: 0x4_1100; GTCCRA1: 0x4_1140; GTCCRA2: 0x4_1180; GTCCRA3: 0x4_11C0 Access:
Group B: GTCCRB0: 0x4_2100; GTCCRB1: 0x4_2140; GTCCRB2: 0x4_2180; GTCCRB3: 0x4_21C0 Read only

	0	1 … 31
R	TOG	COUNT
W		
Reset	All zeros	

Figure 25-14. Global Timer Current Count Registers (GTCCR*xn*)

Table 25-19 describes the GTCCR*xn* fields.

Table 25-19. GTCCR*xn* Field Descriptions

Bits	Name	Description
0	TOG	Toggle. Toggles when the current count decrements to zero. Cleared when GTBCR*xn*[CI] goes from 1 to 0.
1–31	COUNT	Current count. Decremented while GTBCR*xn*[CI] is zero. When the timer count reaches zero, an interrupt is generated (provided it is not masked), the toggle bit is inverted, and the count is reloaded. For non-cascaded timers, the reload value is the contents of the corresponding GTBCR*xn*. Cascaded timers are reloaded with either all ones, or the GTBCR*xn* contents, depending on the value of TCR*n*[ROVR]. See Section 25.3.2.6, "Timer Control Registers (TCRA–TCRB)," for more details.

25.3.2.3 Global Timer Base Count Registers (GTBCRA0–GTBCRA3, GTBCRB0–GTBCRB3)

The GTBCRs contain the base counts for each of the four MPIC timers in each of the two groups, as shown in Figure 25-15. This value is reloaded into the corresponding GTCCR*xn* when the current count reaches

zero. Note that when zero is written to the base count field, (and GTCCR*xn*[CI] = 0), the timer generates an interrupt on every timer cycle.

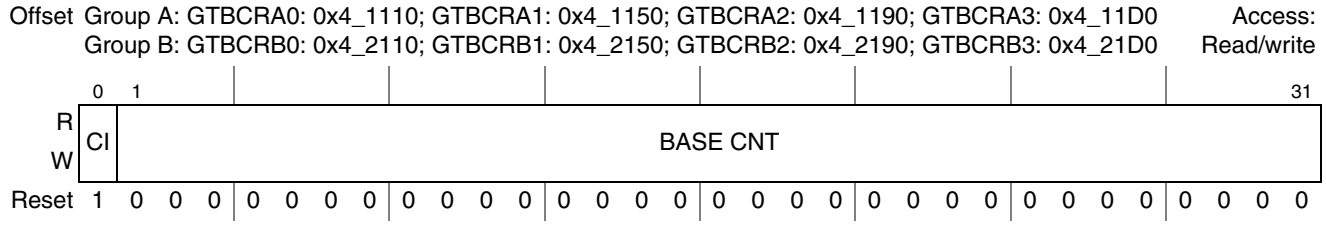


Figure 25-15. Global Timer Base Count Register (GTBCR*xn*)

Table 25-20 describes the GTBCR*xn* fields.

Table 25-20. GTBCR*xn* Field Descriptions

Bits	Name	Description
0	CI	Count inhibit. Always set following reset 0 Counting enabled 1 Counting inhibited
1–31	BASE CNT	Base count. When CI transitions from 1 to 0, this value is copied into the corresponding GTCCR*xn* and the toggle bit is cleared. If CI is already cleared (counting is in progress), the base count is copied to the GTCCR*xn* at the next zero crossing of the current count.

25.3.2.4 Global Timer Vector/Priority Registers (GTVPRA0–GTVPRA3, GTVPRB0–GTVPRB3)

The GTVPRs contain the interrupt vector and the interrupt priority values for the timers as shown in Figure 25-16. They also contain the mask and activity fields for all the timers.

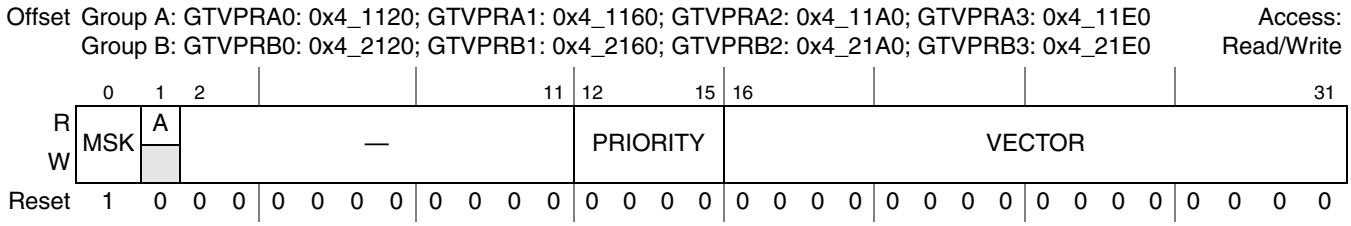


Figure 25-16. Global Timer Vector/Priority Register (GTVPR*xn*)

Table 25-21 describes the GTVPR*xn* fields.

Table 25-21. GTVPR*xn* Field Descriptions

Bits	Name	Description
0	MSK	Mask. Mask interrupts from this source. MSK affects interrupts routed to *int*, *cint*, *mcp*, *sie*[0:2], and IRQ_OUT. 0 An interrupt request is generated if the corresponding IPR bit is set. 1 Further interrupts from this source are disabled.
1	A	Activity. Indicates an interrupt has been requested or is in service. The VECTOR and PRIORITY values should not be changed while this bit is set. Affects only interrupts routed to *int*. 0 No current interrupt activity associated with this source. 1 The interrupt field for this source is set in the IPR or ISR.
2–11	—	Reserved
12–15	PRIORITY	Priority. Specifies the interrupt priority. The lowest priority is 0 and the highest priority is 15. A priority level of 0 inhibits signalling of this interrupt to the core. Affects only interrupts routed to *int*.
16–31	VECTOR	Vector (Affects only interrupts routed to *int*). Contains the value returned when IACK is read and this interrupt resides in the corresponding interrupt request register (IRR) for that core, as shown in Figure 25-86.

25.3.2.5 Global Timer Destination Registers (GTDRA0–GTDRA3, GTDRB0–GTDRB3)

The GTDR*xn* registers, shown in Figure 25-17, control the destination core to which each timer's interrupt is directed. Note that GTDR*xn* bits can be set independently of each other and any or all can be set for this type of interrupt.

Offset Group A: GTDRA0: 0x4_1130; GTDRA1: 0x4_1170; GTDRA2: 0x4_11B0; GTDRA3: 0x4_11F0 Access:
Group B: GTDRB0: 0x4_2130; GTDRB1: 0x4_2170; GTDRB2: 0x4_21B0; GTDRB3: 0x4_21F0 Read/Write

	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	—																								P7	P6	P5	P4	P3	P2	P1	P0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 25-17. Global Timer Destination Registers (GTDR*xn*)

Table 25-22 describes the GTDR*xn* fields.

Table 25-22. GTDR*xn* Field Descriptions

Bits	Name	Description
0–23	—	Reserved
24	P7	Processor core 7. This interrupt is multicasting, so multiple bits can be set. 0 Processor core 7 does not receive this interrupt 1 Directs the timer interrupt to processor core 7.
25	P6	Processor core 6. This interrupt is multicasting, so multiple bits can be set. 0 Processor core 6 does not receive this interrupt 1 Directs the timer interrupt to Processor core 6.

Table 25-22. GTDR*xn* Field Descriptions (continued)

Bits	Name	Description
26	P5	Processor core 5. This interrupt is multicasting, so multiple bits can be set. 0 Processor core 5 does not receive this interrupt 1 Directs the timer interrupt to processor core 5.
27	P4	Processor core 4. This interrupt is multicasting, so multiple bits can be set. 0 Processor core 4 does not receive this interrupt 1 Directs the timer interrupt to processor core 4.
28	P3	Processor core 3. This interrupt is multicasting, so multiple bits can be set. 0 Processor core 3 does not receive this interrupt 1 Directs the timer interrupt to processors core 3.
29	P2	Processor core 2. This interrupt is multicasting, so multiple bits can be set. 0 Processor core 2 does not receive this interrupt 1 Directs the timer interrupt to processor core 2.
30	P1	Processor core 1. This interrupt is multicasting, so multiple bits can be set. 0 Processor core 1 does not receive this interrupt 1 Directs the timer interrupt to processor core 1.
31	P0	Processor core 0. Default destination after MPIC is reset. This interrupt is multicasting, so multiple bits can be set. 0 Processor core 0 does not receive this interrupt. 1 Directs the timer interrupt to processor core 0.

25.3.2.6 Timer Control Registers (TCRA–TCRB)

The TCR registers, shown in Figure 25-18, provide various configuration options such as count frequency and roll-over behavior for the timers.

There are two choices for the clock source for the timers: a selectable frequency ratio from the MPIC input clock, or the RTC signal. TCRs can be cascaded to create timers larger than the default 31-bit global timers. Timer cascade fields allow configuration of up to two 63-bit timers, one 95-bit timer, or one 127-bit timer (within each group).

With one exception mentioned below, the value reloaded into a timer is determined by its roll-over control field, TCR*x*[ROVR]. Setting TCR*x*[ROVR] causes its GTCCR*xn* to roll over to all ones when the count reaches zero. This is equivalent to reloading the count register with 0xFFFF_FFFF instead of its base count value. Clearing a timer's associated ROVR bit ensures the timer always reloads with its base count value.

When timers are cascaded, the last (most significant) counter in the cascade also affects their roll-over behavior. Cascaded timers always reload their base count when the most significant counter has decremented to zero, regardless of the TCR*x*[ROVR] settings.

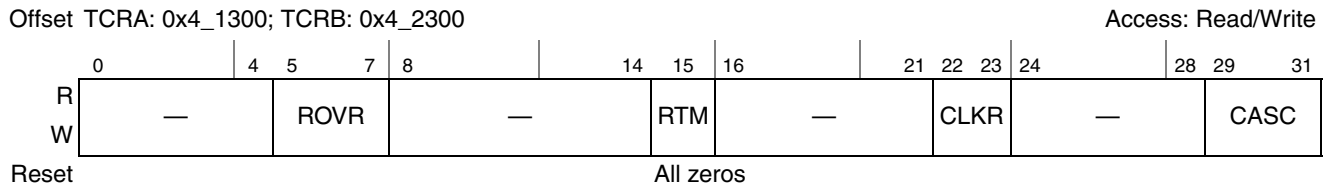


Figure 25-18. Timer Control Registers (TCR*x*)

Table 25-23 describes the TCR*x* fields.

Table 25-23. TCR*x* Field Descriptions

Bits	Name	Description
0–4	—	Reserved
5–7	ROVR	Roll-over control for cascaded timers only. Specifies behavior when count reaches zero by identifying the source of the reload value. Cascaded timers are always reloaded with their base count value when the more significant timer in the cascade (the upstream timer) is zero. Bits 5–7 correspond to timers 2–0. Note that global timer 3 always reloads with its GTBCR*xn*. 0 The timer does not roll over. When the count reaches zero, GTCCR*xn* is reloaded with the GTBCR*xn* value. 1 Timer rolls over at zero to all ones. (When the count reaches zero, GTCCR*xn* is reloaded with 0xFFFF_FFFF.) 000 All timers reload with base count. 001 Timers 1 and 2 reload with base count, timer 0 rolls over (reloads with 0xFFFF_FFFF). 010 Timers 0 and 2 reload with base count, timer 1 rolls over (reloads with 0xFFFF_FFFF). 011 Timer 2 reloads with base count, timers 0 and 1 roll over (reload with 0xFFFF_FFFF). 100 Timers 0 and 1 reload with base count, timer 2 rolls over (reloads with 0xFFFF_FFFF). 101 Timer 1 reloads with base count, timers 0 and 2 roll over (reload with 0xFFFF_FFFF). 110 Timer 0 reloads with base count, timers 1 and 2 roll over (reload with 0xFFFF_FFFF). 111 Timers 0, 1, and 2 roll over (reload with 0xFFFF_FFFF).
8–14	—	Reserved
15	RTM	Real time mode. Specifies the clock source for the MPIC timers. 0 Timer clock frequency is a ratio of the frequency of the MPIC input clock as determined by the CLKR field. This is the default value. 1 The RTC signal is used to clock the MPIC timers. If this bit is set, the CLKR field has no meaning.
16–21	—	Reserved
22–23	CLKR	Clock ratio. Specifies the ratio of the timer frequency to the MPIC input clock. The following clock ratios are supported: 00 Default. Divide by 8 01 Divide by 16 10 Divide by 32 11 Divide by 64
24–28	—	Reserved
29–31	CASC	Cascade timers. Specifies the output of particular global timers as input to others. 000 Default. Timers not cascaded 001 Cascade timers 0 and 1 010 Cascade timers 1 and 2 011 Cascade timers 0, 1, and 2 100 Cascade timers 2 and 3 101 Cascade timers 0 and 1; timers 2 and 3 110 Cascade timers 1, 2, and 3 111 Cascade timers 0, 1, 2, and 3

25.3.3 Global Performance Monitor Mask Registers (PMMRs)

The performance monitor mask registers consist of four sets of twelve 32-bit registers, PM*n*MR0, PM*n*MR1, through PM*n*MR11, one set per performance monitor signal. Each of the four sets can be configured to select one interrupt source (interprocessor, timer, message, shared message signaled, external, or internal) to generate a performance monitor event. The performance monitor can be configured to track this event in the performance monitor local control registers.

25.3.3.1 Performance Monitor Mask Registers 0 (PM0MR0–PM3MR0)

Each PM*n*MR0 register, shown in Figure 25-19, is matched with a PM*n*MR1 through PM*n*MR11 register. Because each unreserved bit in the mask register vector (PM*n*MR0/1/.../11) specifies a different interrupt, only one bit in the vector can be unmasked at a time. Unmasking more than one bit per set is considered a programming error and results in unpredictable behavior.

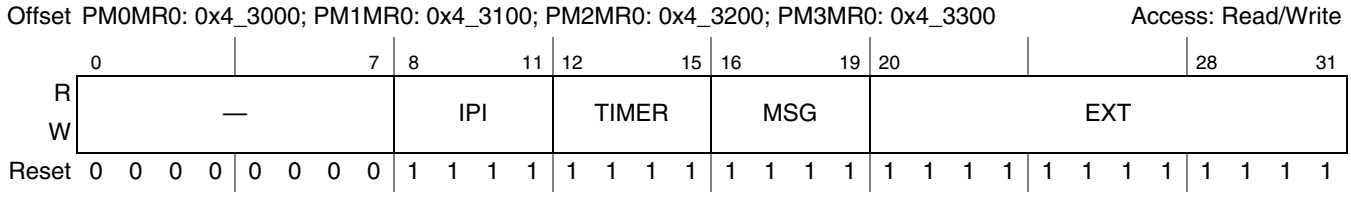


Figure 25-19. Performance Monitor Mask Registers 0 (PM*n*MR0)

Table 25-24 describes the PM*n*MR0 fields.

Table 25-24. PM*n*MR0 Field Descriptions

Bits	Name	Description
0–7	—	Reserved
8–11	IPI	Interprocessor interrupts 0–3 0 The corresponding interrupt source generates a performance monitor event when the interrupt occurs. 1 The corresponding interrupt does not generate a performance monitor event.
12–15	TIMER	Timer interrupts 0–3 (Group A and Group B: Each bit represents an OR of the event for the correspondingly numbered timer in Group A and that in Group B). 0 The corresponding interrupt source generates a performance monitor event when the interrupt occurs. 1 The corresponding interrupt does not generate a performance monitor event.
16–19	MSG	Message interrupts 0–3 (Group A and Group B: Each bit represents an OR of the event for the correspondingly numbered timer in Group A and that in Group B) 0 The corresponding interrupt source generates a performance monitor event when the interrupt occurs. 1 The corresponding interrupt does not generate a performance monitor event.
20–31	EXT	External interrupts IRQ[0:11] 0 The corresponding interrupt source generates a performance monitor event when the interrupt occurs. 1 The corresponding interrupt does not generate a performance monitor event.

25.3.3.2 Performance Monitor Mask Registers 1 (PM0MR1–PM3MR1)

Each PM*n*MR1 register, shown in Figure 25-20, is matched with a PM*n*MR1 through PM*n*MR8 register. Because each unreserved bit in the mask register vector (PM*n*MR0/1/.../8) specifies a different interrupt, only one bit in the vector can be unmasked at a time. Unmasking more than one bit per set is considered a programming error and results in unpredictable behavior.

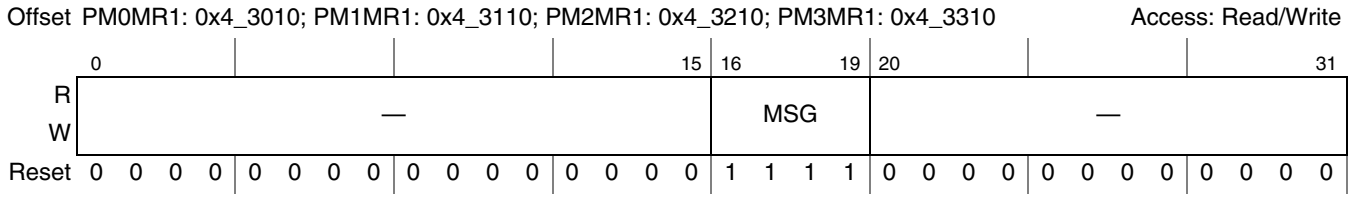


Figure 25-20. Performance Monitor Mask Registers 1 (PM*n*MR1)

Table 25-25 describes the PM*n*MR1 fields.

Table 25-25. PM*n*MR1 Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–19	MSG	Message interrupts 4–7 (Group A and Group B: Each bit represents an OR of the event for the correspondingly numbered timer in Group A and that in Group B) 0 The corresponding interrupt source generates a performance monitor event when the interrupt occurs. 1 The corresponding interrupt does not generate a performance monitor event.
20–31	—	Reserved

25.3.3.3 Performance Monitor Mask Registers 2 (PM0MR2–PM3MR2)

Table 25-21 describes the PM*n*MR2 fields.

Offset PM0MR2: 0x4_3020; PM1MR2: 0x4_3120; PM2MR2: 0x4_3220; PM3MR2: 0x4_3320 Access: Read/Write

	0				7	8			15	16			23	24			31
R W	MSID					MSIC				MSIB				MSIA			
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 25-21. Performance Monitor Mask Registers 2 (PM*n*MR2)

Table 25-26 describes the PM*n*MR2 fields.

Table 25-26. PM*n*MR2 Field Descriptions

Bits	Name	Description
0–7	MSID	Shared message signaled interrupts D 0-7 0 The corresponding interrupt source generates a performance monitor event when the interrupt occurs. 1 The corresponding interrupt does not generate a performance monitor event.
8–15	MSIC	Shared message signaled interrupts C 0-7 0 The corresponding interrupt source generates a performance monitor event when the interrupt occurs. 1 The corresponding interrupt does not generate a performance monitor event.
16–23	MSIB	Shared message signaled interrupts B 0-7 0 The corresponding interrupt source generates a performance monitor event when the interrupt occurs. 1 The corresponding interrupt does not generate a performance monitor event.
24–31	MSIA	Shared message signaled interrupts A 0-7 0 The corresponding interrupt source generates a performance monitor event when the interrupt occurs. 1 The corresponding interrupt does not generate a performance monitor event.

25.3.3.4 Performance Monitor Mask Registers 3 (PM0MR3–PM3MR3)

Table 25-22 describes the PM*n*MR3 fields.

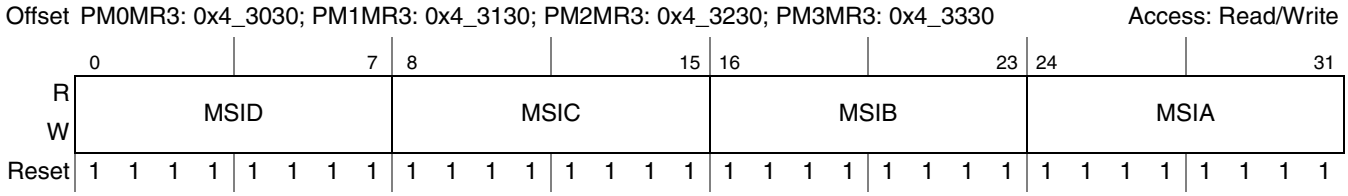


Figure 25-22. Performance Monitor Mask Registers 3 (PM*n*MR3)

Table 25-27 describes the PM*n*MR3 fields.

Table 25-27. PM*n*MR3 Field Descriptions

Bits	Name	Description
0–7	MSID	Shared message signaled interrupts D 8-15 0 The corresponding interrupt source generates a performance monitor event when the interrupt occurs. 1 The corresponding interrupt does not generate a performance monitor event.
8–15	MSIC	Shared message signaled interrupts C 8-15 0 The corresponding interrupt source generates a performance monitor event when the interrupt occurs. 1 The corresponding interrupt does not generate a performance monitor event.
16–23	MSIB	Shared message signaled interrupts B 8-15 0 The corresponding interrupt source generates a performance monitor event when the interrupt occurs. 1 The corresponding interrupt does not generate a performance monitor event.
24–31	MSIA	Shared message signaled interrupts A 8-15 0 The corresponding interrupt source generates a performance monitor event when the interrupt occurs. 1 The corresponding interrupt does not generate a performance monitor event.

25.3.3.5 Performance Monitor Mask Registers 4(PM0MR4–PM3MR4)

Figure 25-23 shows the PM*n*MR4 registers.

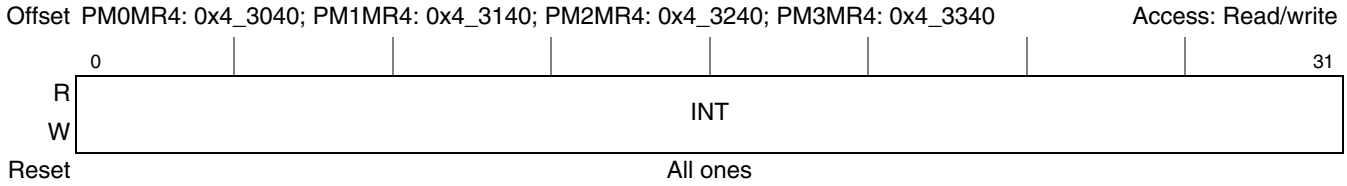


Figure 25-23. Performance Monitor Mask Registers 4 (PM*n*MR4)

Table 25-28 describes the PM*n*MR4 registers.

Table 25-28. PM*n*MR4 Field Descriptions

Bits	Name	Description
0–31	INT	Internal interrupts 0–31 0 The corresponding interrupt source generates a performance monitor event when the interrupt occurs. 1 The corresponding interrupt does not generate a performance monitor event.

25.3.3.6 Performance Monitor Mask Registers 5 (PM0MR5–PM3MR5)

Figure 25-24 shows the PM*n*MR5 registers.

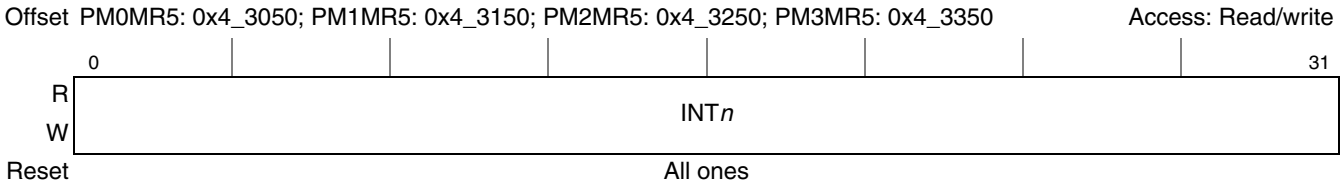


Figure 25-24. Performance Monitor Mask Registers 5 (PM*n*MR5)

Table 25-29 describes the PM*n*MR5 registers.

Table 25-29. PM*n*MR5 Field Descriptions

Bits	Name	Description
0–31	INT	Internal interrupts 32-63 0 The corresponding interrupt source generates a performance monitor event when the interrupt occurs. 1 The corresponding interrupt does not generate a performance monitor event.

25.3.3.7 Performance Monitor Mask Registers 6 (PM0MR6–PM3MR6)

Figure 25-25 shows the PM*n*MR6 registers.

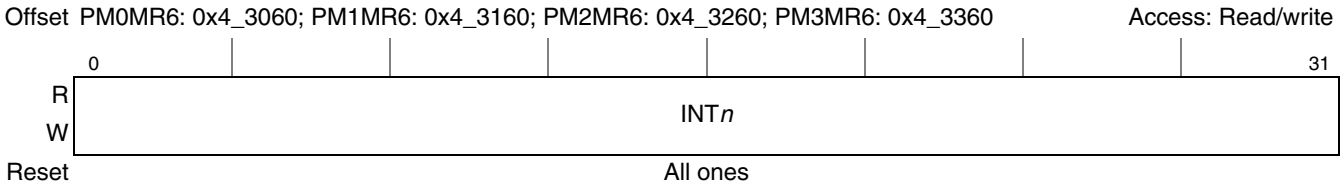


Figure 25-25. Performance Monitor Mask Registers 6 (PM*n*MR6)

Table 25-30 describes the PM*n*MR6 registers.

Table 25-30. PM*n*MR6 Field Descriptions

Bits	Name	Description
0–31	INT	Internal interrupts 64-95 0 The corresponding interrupt source generates a performance monitor event when the interrupt occurs. 1 The corresponding interrupt does not generate a performance monitor event.

25.3.3.8 Performance Monitor Mask Registers 7 (PM0MR7–PM3MR7)

Figure 25-26 shows the PM*n*MR7 registers.

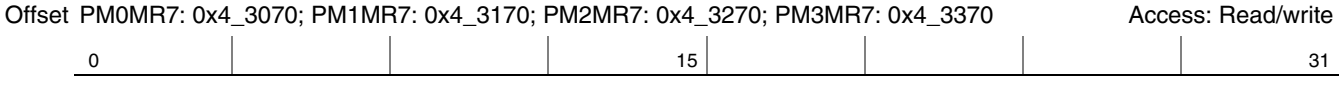


Figure 25-26. Performance Monitor Mask Registers 7 (PM*n*MR7)

Table 25-31 describes the PM*n*MR7 registers.

25.3.3.9 Performance Monitor Mask Registers 8 (PM0MR8–PM3MR8)

Figure 25-27 shows the PM*n*MR8 registers.

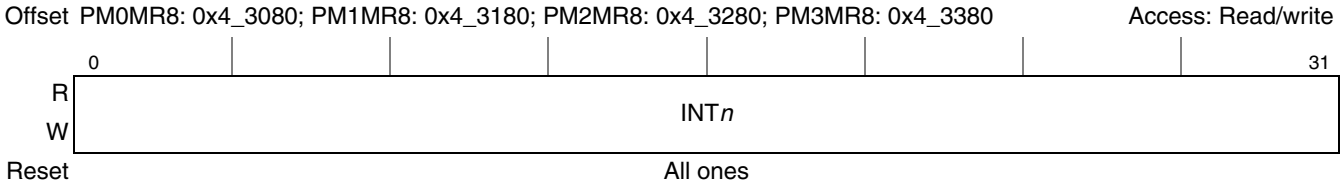


Figure 25-27. Performance Monitor Mask Registers 8 (PM*n*MR8)

Table 25-32 describes the PM*n*MR8 registers.

Table 25-32. PM*n*MR8 Field Descriptions

Bits	Name	Description
0–31	INT	Internal interrupts 128-159 0 The corresponding interrupt source generates a performance monitor event when the interrupt occurs. 1 The corresponding interrupt does not generate a performance monitor event.

25.3.3.10 Performance Monitor Mask Registers 9 (PM0MR9–PM3MR9)

Figure 25-27 shows the PM*n*MR9 registers.

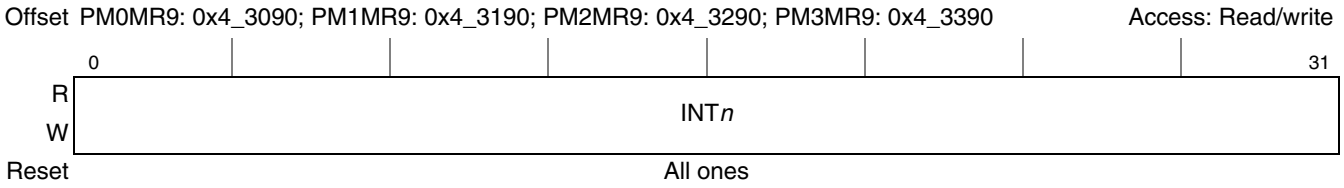


Figure 25-28. Performance Monitor Mask Registers 9 (PM*n*MR9)

Table 25-32 describes the PM*n*MR9 registers.

Table 25-33. PM*n*MR9 Field Descriptions

Bits	Name	Description
0–31	INT	Internal interrupts 160-191 0 The corresponding interrupt source generates a performance monitor event when the interrupt occurs. 1 The corresponding interrupt does not generate a performance monitor event.

25.3.3.11 Performance Monitor Mask Registers 10 (PM0MR10–PM3MR10)

Figure 25-27 shows the PM*n*MR10 registers.

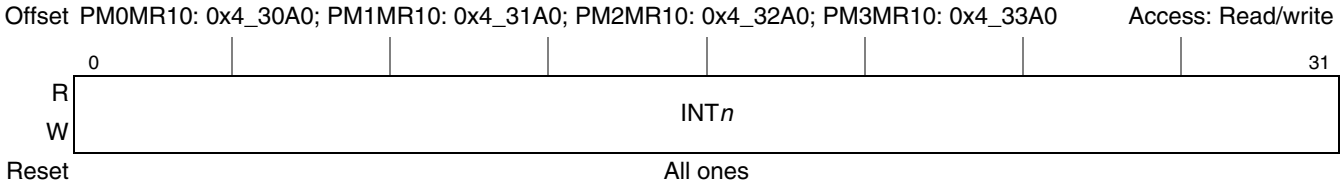


Figure 25-29. Performance Monitor Mask Registers 10 (PM*n*MR10)

Table 25-32 describes the PM*n*MR10 registers.

Table 25-34. PM*n*MR10 Field Descriptions

Bits	Name	Description
0–31	INT	Internal interrupts 192-223 0 The corresponding interrupt source generates a performance monitor event when the interrupt occurs. 1 The corresponding interrupt does not generate a performance monitor event.

25.3.3.12 Performance Monitor Mask Registers 11 (PM0MR11–PM3MR11)

Figure 25-27 shows the PM*n*MR11 registers.

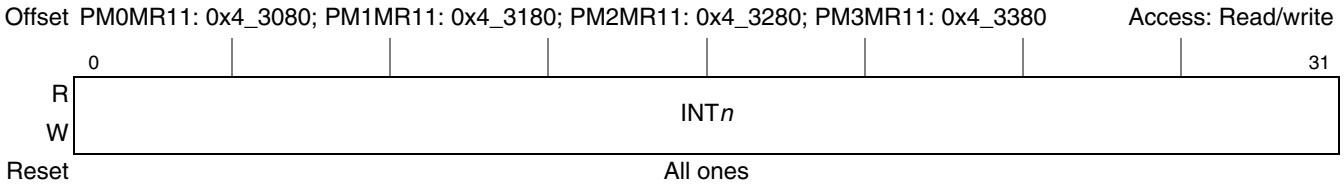


Figure 25-30. Performance Monitor Mask Registers 11 (PM*n*MR11)

Table 25-32 describes the PM*n*MR11 registers.

Table 25-35. PM*n*MR11 Field Descriptions

Bits	Name	Description
0–31	INT	Internal interrupts 224-255 0 The corresponding interrupt source generates a performance monitor event when the interrupt occurs. 1 The corresponding interrupt does not generate a performance monitor event.

25.3.4 Global Interrupt Summary Registers

This section describes the various interrupt summary registers.

25.3.4.1 External Interrupt Summary Register (ERQSR)

NOTE

The fields in ERQSR report only the current state of IRQ0–11. These fields were designed to work with level-sensitive interrupts; the values returned for edge-sensitive interrupts may be unreliable.

Figure 25-31 shows the ERQSR fields.

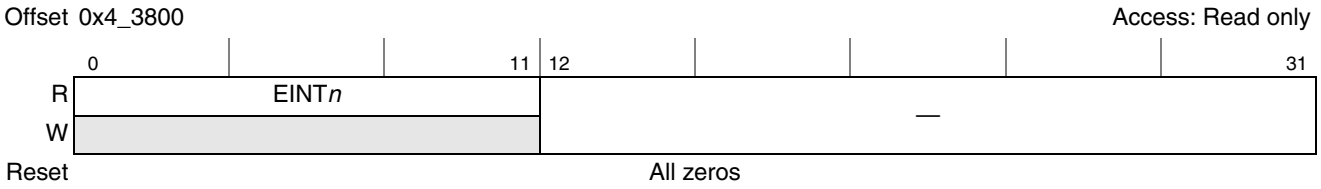


Figure 25-31. External Interrupt Summary Register (ERQSR)

Table 25-36 describes the ERQSR fields.

Table 25-36. ERQSR Field Descriptions

Bits	Name	Description
0–11	EINT*n*	External interrupts signal 0–11 status. Bit 0 represents external interrupt 0. 0 The corresponding external interrupt signal is not active. 1 The corresponding external interrupt signal is active.
12–31	—	Reserved

25.3.4.2 Error Interrupt Summary Register 0 (EISR0)

Figure 25-32 shows the EISR0 fields. The error interrupt is mapped to internal interrupt number 0.

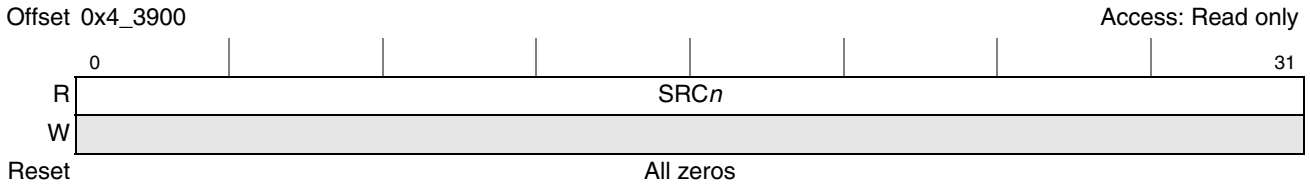


Figure 25-32. Error Interrupt Summary Register 0 (EISR0)

Table 25-37 describes the EISR0 fields.

Table 25-37. EISR0 Field Descriptions

Bits	Name	Description
0–31	SRC*n*	Error interrupt source 0–31. Bit 0 represents SRC0. 0 The corresponding error interrupt source is not active. 1 The corresponding error interrupt source is active. See “ORed Error Interrupt Sources,” in the Interrupt Assignments chapter for the error interrupt sources reported in this register.

25.3.4.3 Error Interrupt Mask Register 0 (EIMR0)

Figure 25-33 shows the EIMR0 fields. The error interrupt is mapped to internal interrupt number 0.

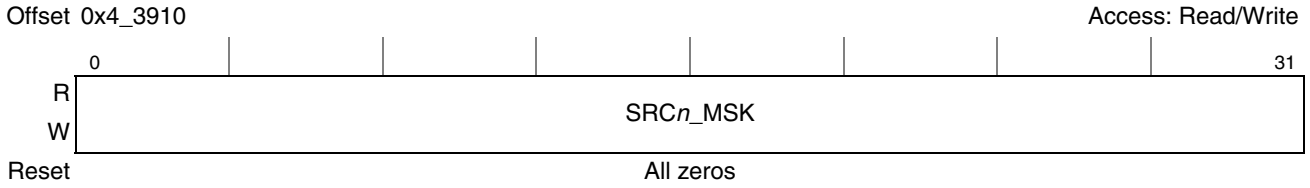


Figure 25-33. Error Interrupt Mask Register 0 (EIMR0)

Table 25-38 describes the EIMR0 fields.

Table 25-38. EIMR0 Field Descriptions

Bits	Name	Description
0–31	SRC*n*_MSK	Error interrupt source mask 0–31. Bit 0 represents SRC0_MSK (for SRC0). 0 The corresponding error interrupt source is not masked and will report an error in EISR0.. 1 The corresponding error interrupt source is masked and will not report an error in EISR0. See “ORed Error Interrupt Sources,” in the Interrupt Assignments chapter for the error interrupt sources associated with this register.

25.3.4.4 Watchdog Status Register Summary Register (WSRSR)

Figure 25-34 shows WSRSR. If any of the WRS*n* bits in this register are set, an internal interrupt is generated. The watchdog interrupt is mapped to internal interrupt number 1.

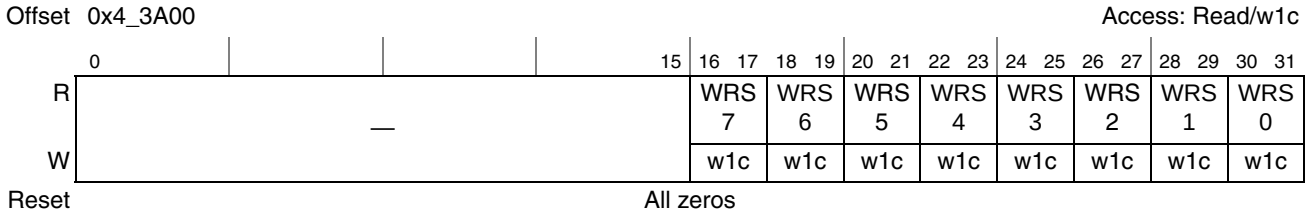


Figure 25-34. Watchdog Status Register Summary Register (WSRSR)

Table 25-39 describes WSRSR fields.

Table 25-39. WSRSR Field Descriptions

Bits	Name	Description
0–15	—	Reserved, should be cleared
16–17	WRS7[0:1]	Processor 7 WRS signals. A non-zero value means that a core watchdog timer event has occurred. Write 1s to clear.
18–19	WRS6[0:1]	Processor 6 WRS signals. A non-zero value means that a core watchdog timer event has occurred. Write 1s to clear.
20–21	WRS5[0:1]	Processor 5 WRS signals. A non-zero value means that a core watchdog timer event has occurred. Write 1s to clear.
22–23	WRS4[0:1]	Processor 4 WRS signals. A non-zero value means that a core watchdog timer event has occurred. Write 1s to clear.

Table 25-39. WSRSR Field Descriptions (continued)

Bits	Name	Description
24–25	WRS3[0:1]	Processor 3 WRS signals. A non-zero value means that a core watchdog timer event has occurred. Write 1s to clear.
26–27	WRS2[0:1]	Processor 2 WRS signals. A non-zero value means that a core watchdog timer event has occurred. Write 1s to clear.
28–29	WRS1[0:1]	Processor 1 WRS signals. A non-zero value means that a core watchdog timer event has occurred. Write 1s to clear.
30–31	WRS0[0:1]	Processor 0 WRS signals. A non-zero value means that a core watchdog timer event has occurred. Write 1s to clear.

25.3.4.5 Critical Interrupt, Machine Check, and IRQ_OUT/SoC Interrupt Event Summary Registers

The summary registers indicate the specific interrupt sources routed to the *cintn, mcpn, sien or* $\overline{\text{IRQ\_OUT}}$ MPIC outputs. Summary register bits are cleared when the corresponding interrupt that caused a bit to be set is negated. Note that only level-sensitive interrupts can be routed to *cintn*, *mcpn sien* or $\overline{\text{IRQ\_OUT}}$. Also note that if an interrupt source is masked through *x*VPR*n* mask bit, and the destination is *cint*, *mcp*, $\overline{\text{IRQ\_OUT}}$ or *sie*, the corresponding summary register bit will not be set even if the source is active.

The critical interrupt summary registers, shown in Figure 25-35 through Figure 25-39, contain one bit for each interrupt source that can be designated as a critical interrupt. The corresponding bit is set if the interrupt is active and is routed to the appropriate *cintn* outputs of the MPIC.

The machine check summary registers, shown in Figure 25-44 through Figure 25-48, contain one bit for each interrupt source that can be designated as a machine check interrupt. The corresponding bit is set if the interrupt is active and is routed to the appropriate *mcpn* outputs of the MPIC.

The $\overline{\text{IRQ\_OUT}}$/SoC interrupt event summary registers, shown in Figure 25-53 through Figure 25-57 contain one bit for each interrupt source that can be routed to *sien* or $\overline{\text{IRQ\_OUT}}$. The corresponding bit is set if the interrupt is active and is routed to *sien* or $\overline{\text{IRQ\_OUT}}$.

25.3.4.5.1 Critical Interrupt Summary Register 0 (CISR0)

Figure 25-35 shows CISR0.

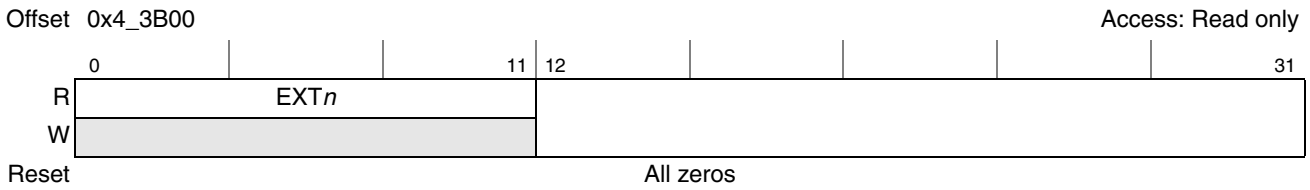


Figure 25-35. Critical Interrupt Summary Register 0 (CISR0)

Table 25-40 describes CISR0 fields.

Table 25-40. CISR0 Field Descriptions

Bits	Name	Description
0–11	EXT*n*	External interrupts 0–11. Bit 0 represents IRQ0. 0 The corresponding interrupt is not active or not routed to *cintn*. 1 The corresponding interrupt is active and is routed to *cintn* (if the corresponding interrupt level register field is set to *cint*).
12–31	—	Reserved

25.3.4.5.2 Critical Interrupt Summary Register 1 (CISR1)

Figure 25-36 shows the CISR1.

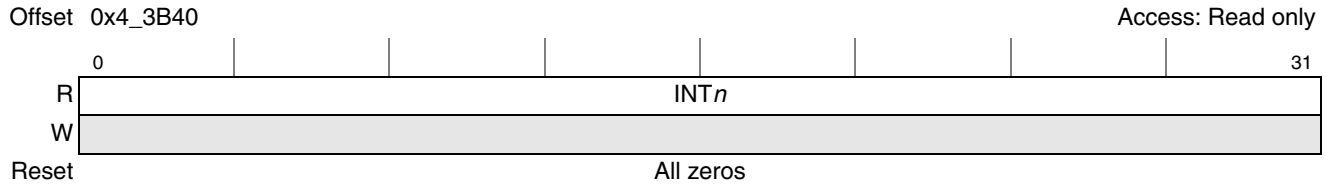


Figure 25-36. Critical Interrupt Summary Register 1 (CISR1)

Table 25-41 describes CISR1.

Table 25-41. CISR1 Field Descriptions

Bits	Name	Description
0–31	INT*n*	Internal interrupts 0–31. Bit 0 represents INT0. 0 The corresponding interrupt is not active or not routed to *cintn*. 1 The corresponding interrupt is active and is routed to *cintn* (if the corresponding interrupt level register field is set to *cint*).

25.3.4.5.3 Critical Interrupt Summary Register 2 (CISR2)

Figure 25-37 shows the CISR2.

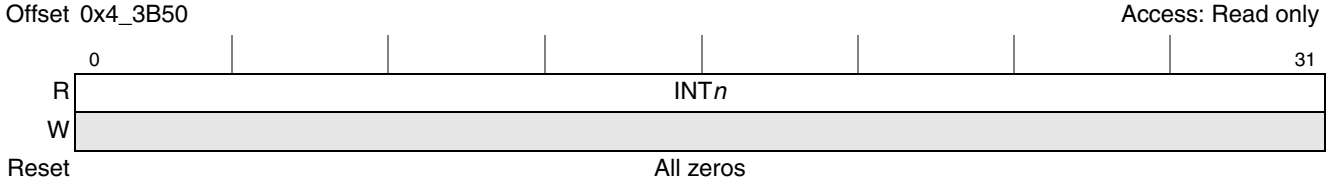


Figure 25-37. Critical Interrupt Summary Register 2 (CISR2)

Table 25-42 describes CISR2.

Table 25-42. CISR2 Field Descriptions

Bits	Name	Description
0–31	INT*n*	Internal interrupts 32–63. Bit 0 represents INT32. 0 The corresponding interrupt is not active or not routed to *cintn*. 1 The corresponding interrupt is active and is routed to *cintn* (if the corresponding interrupt level register field is set to *cint*).

25.3.4.5.4 Critical Interrupt Summary Register 3 (CISR3)

Figure 25-38 shows the CISR3.

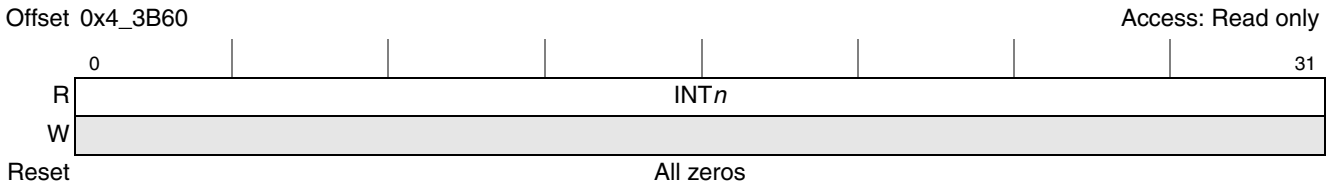


Figure 25-38. Critical Interrupt Summary Register 3 (CISR3)

Table 25-43 describes CISR3.

Table 25-43. CISR3 Field Descriptions

Bits	Name	Description
0–31	INT*n*	Internal interrupts 64-95. Bit 0 represents INT64. 0 The corresponding interrupt is not active or not routed to *cintn*. 1 The corresponding interrupt is active and is routed to *cintn* (if the corresponding interrupt level register field is set to *cint*).

25.3.4.5.5 Critical Interrupt Summary Register 4 (CISR4)

Figure 25-39 shows the CISR4.

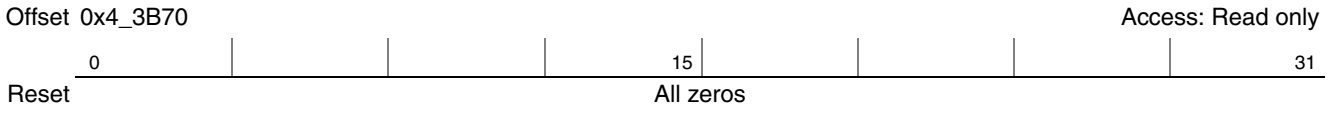


Figure 25-39. Critical Interrupt Summary Register 4 (CISR4)

Table 25-44 describes CISR4.

25.3.4.5.6 Critical Interrupt Summary Register 5 (CISR5)

Figure 25-40 shows the CISR5.

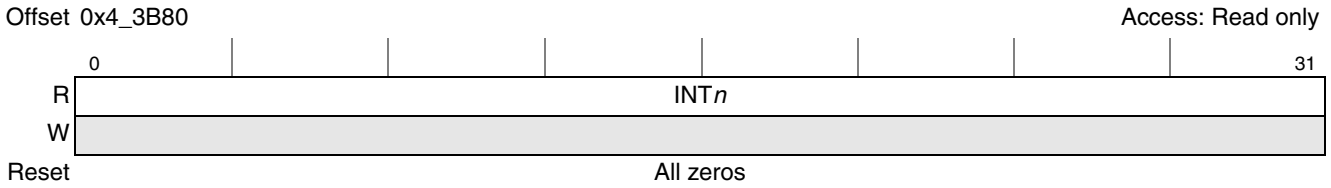


Figure 25-40. Critical Interrupt Summary Register 5 (CISR5)

Table 25-45 describes CISR5.

Table 25-45. CISR5 Field Descriptions

Bits	Name	Description
0–31	INT*n*	Internal interrupts 128-159. Bit 0 represents INT128. 0 The corresponding interrupt is not active or not routed to *cintn*. 1 The corresponding interrupt is active and is routed to *cintn* (if the corresponding interrupt level register field is set to *cint*).

25.3.4.5.7 Critical Interrupt Summary Register 6 (CISR6)

Figure 25-40 shows the CISR6.

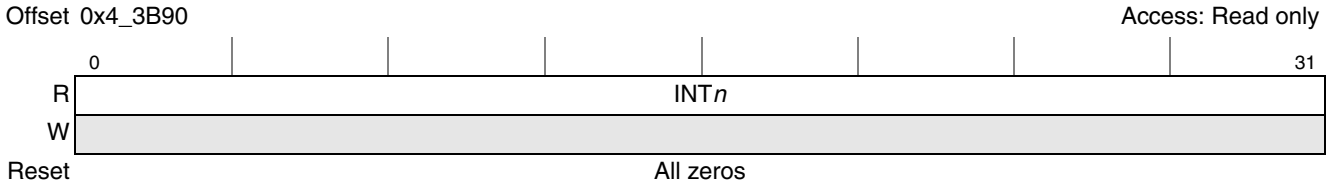


Figure 25-41. Critical Interrupt Summary Register 6 (CISR6)

Table 25-45 describes CISR6.

Table 25-46. CISR6 Field Descriptions

Bits	Name	Description
0–31	INT*n*	Internal interrupts 160-191. Bit 0 represents INT160. 0 The corresponding interrupt is not active or not routed to *cintn*. 1 The corresponding interrupt is active and is routed to *cintn* (if the corresponding interrupt level register field is set to *cint*).

25.3.4.5.8 Critical Interrupt Summary Register 7 (CISR7)

Figure 25-40 shows the CISR7.

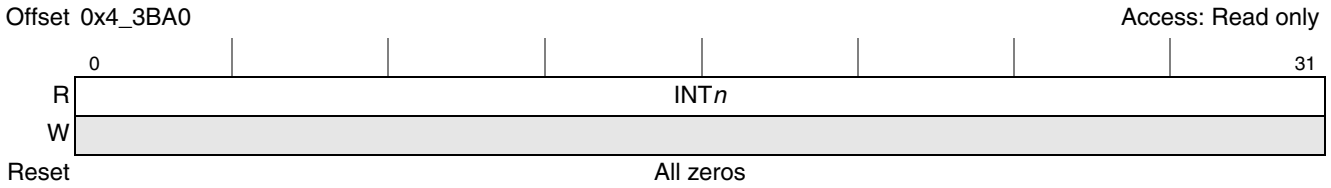


Figure 25-42. Critical Interrupt Summary Register 7 (CISR7)

Table 25-45 describes CISR7.

Table 25-47. CISR7 Field Descriptions

Bits	Name	Description
0–31	INT*n*	Internal interrupts 192-223. Bit 0 represents INT192. 0 The corresponding interrupt is not active or not routed to *cintn*. 1 The corresponding interrupt is active and is routed to *cintn* (if the corresponding interrupt level register field is set to *cint*).

25.3.4.5.9 Critical Interrupt Summary Register 8 (CISR8)

Figure 25-40 shows the CISR8.

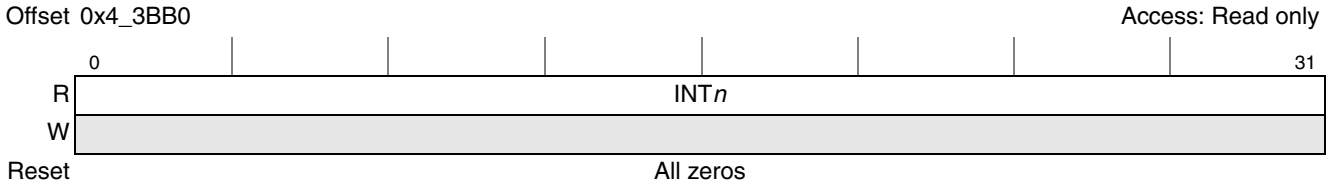


Figure 25-43. Critical Interrupt Summary Register 8 (CISR8)

Table 25-45 describes CISR8.

Table 25-48. CISR8 Field Descriptions

Bits	Name	Description
0–31	INT*n*	Internal interrupts 224-255. Bit 0 represents INT224. 0 The corresponding interrupt is not active or not routed to *cintn*. 1 The corresponding interrupt is active and is routed to *cintn* (if the corresponding interrupt level register field is set to *cint*).

25.3.4.5.10 Machine Check Summary Register 0 (MCSR0)

Figure 25-44 shows MCSR0.

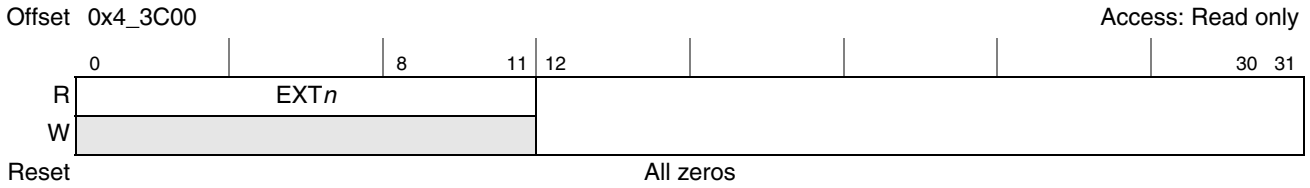


Figure 25-44. Machine Check Summary Register 0 (MCSR0)

Table 25-49 describes MCSR0 fields.

Table 25-49. MCSR0 Field Descriptions

Bits	Name	Description
0–11	EXT*n*	External interrupts 0–11. Bit 0 represents IRQ0. 0 The corresponding interrupt is not active or not routed to *mcpn*. 1 The corresponding interrupt is active and is routed to *mcpn* (if the corresponding interrupt level register field is set to *mcp*).
12–31	—	Reserved

25.3.4.5.11 Machine Check Summary Register 1 (MCSR1)

Figure 25-45 shows the MCSR1.

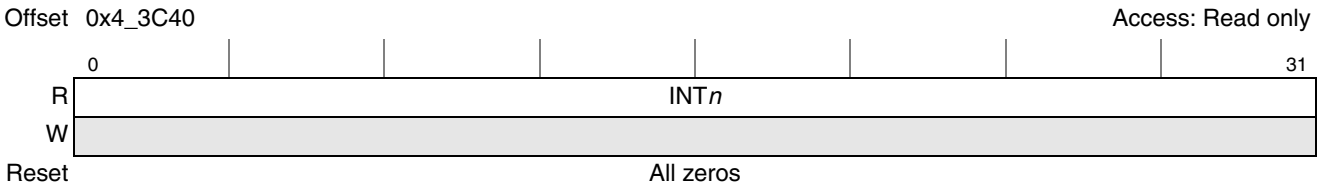


Figure 25-45. Machine Check Summary Register 1 (MCSR1)

Table 25-50 describes MCSR1.

Table 25-50. MCSR1 Field Descriptions

Bits	Name	Description
0–31	INT*n*	Internal interrupts 0–31. Bit 0 represents INT0. 0 Corresponding interrupt is not active or not routed to *mcpn*. 1 The corresponding interrupt is active and is routed to the *mcpn* (if the corresponding interrupt level register field is set to *mcp*).

25.3.4.5.12 Machine Check Summary Register 2 (MCSR2)

Figure 25-46 shows the MCSR2.

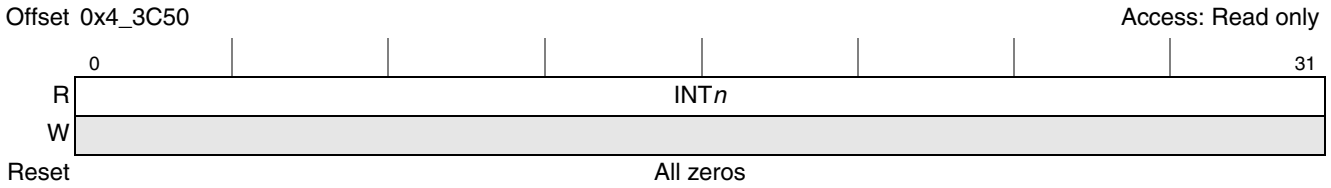


Figure 25-46. Machine Check Summary Register 2 (MCSR2)

Table 25-51 describes MCSR2.

Table 25-51. MCSR2 Field Descriptions

Bits	Name	Description
0–31	INT*n*	Internal interrupts 32–63. Bit 0 represents INT32. 0 The corresponding interrupt is not active or not routed to *mcpn*. 1 The corresponding interrupt is active and is routed to *mcpn* (if the corresponding interrupt level register field is set to *mcp*).

25.3.4.5.13 Machine Check Summary Register 3 (MCSR3)

Figure 25-47 shows the MCSR3.

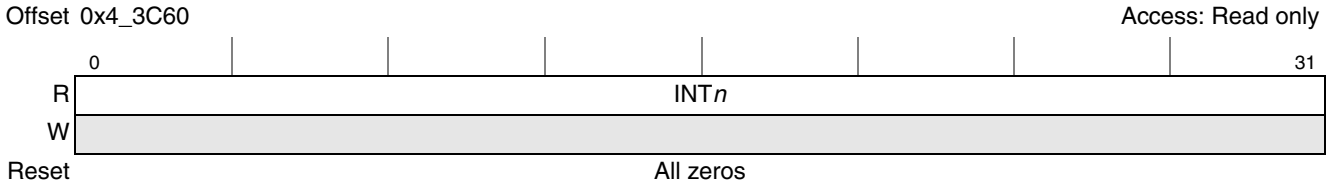


Figure 25-47. Machine Check Summary Register 3 (MCSR3)

Table 25-52 describes MCSR3.

Table 25-52. MCSR3 Field Descriptions

Bits	Name	Description
0–31	INT*n*	Internal interrupts 64–95. Bit 0 represents INT64. 0 The corresponding interrupt is not active or not routed to *mcpn*. 1 The corresponding interrupt is active and is routed to *mcpn* (if the corresponding interrupt level register field is set to *mcp*).

25.3.4.5.14 Machine Check Summary Register 4 (MCSR4)

Figure 25-48 shows the MCSR4.

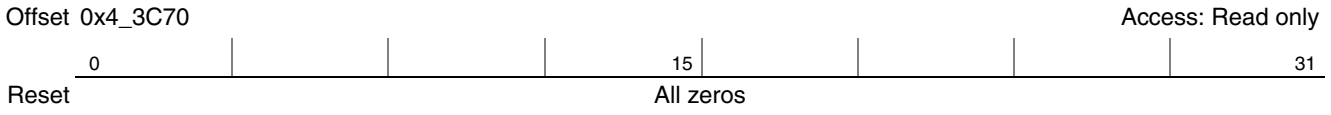


Figure 25-48. Machine Check Summary Register 4 (MCSR4)

Table 25-53 describes MCSR4.

25.3.4.5.15 Machine Check Summary Register 5 (MCSR5)

Figure 25-49 shows the MCSR5.

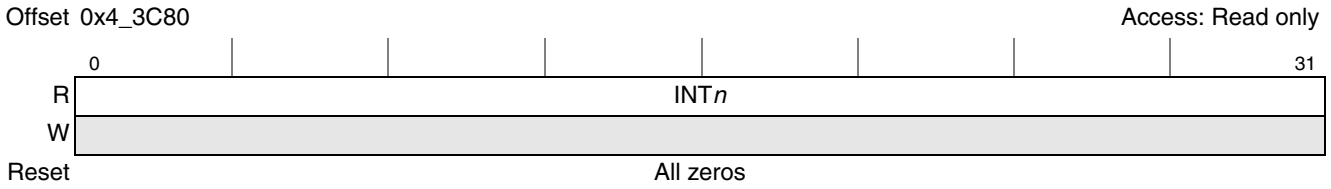


Figure 25-49. Machine Check Summary Register 5 (MCSR5)

Table 25-54 describes MCSR5.

Table 25-54. MCSR5 Field Descriptions

Bits	Name	Description
0–31	INT*n*	Internal interrupts 128–159. Bit 0 represents INT128. 0 The corresponding interrupt is not active or not routed to *mcpn*. 1 The corresponding interrupt is active and is routed to *mcpn* (if the corresponding interrupt level register field is set to *mcp*).

25.3.4.5.16 Machine Check Summary Register 6 (MCSR6)

Figure 25-49 shows the MCSR6.

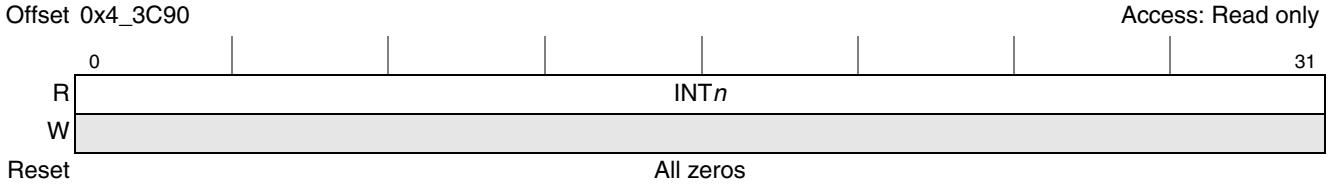


Figure 25-50. Machine Check Summary Register 6 (MCSR6)

Table 25-54 describes MCSR6.

Table 25-55. MCSR6 Field Descriptions

Bits	Name	Description
0–31	INT*n*	Internal interrupts 160–191. Bit 0 represents INT160. 0 The corresponding interrupt is not active or not routed to *mcpn*. 1 The corresponding interrupt is active and is routed to *mcpn* (if the corresponding interrupt level register field is set to *mcp*).

25.3.4.5.17 Machine Check Summary Register 7 (MCSR7)

Figure 25-49 shows the MCSR7.

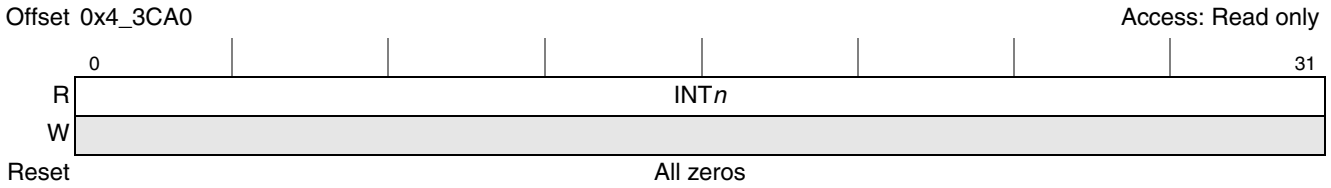


Figure 25-51. Machine Check Summary Register 7 (MCSR7)

Table 25-54 describes MCSR7.

Table 25-56. MCSR7 Field Descriptions

Bits	Name	Description
0–31	INT*n*	Internal interrupts 192–223. Bit 0 represents INT192. 0 The corresponding interrupt is not active or not routed to *mcpn*. 1 The corresponding interrupt is active and is routed to *mcpn* (if the corresponding interrupt level register field is set to *mcp*).

25.3.4.5.18 Machine Check Summary Register 8 (MCSR8)

Figure 25-49 shows the MCSR8.

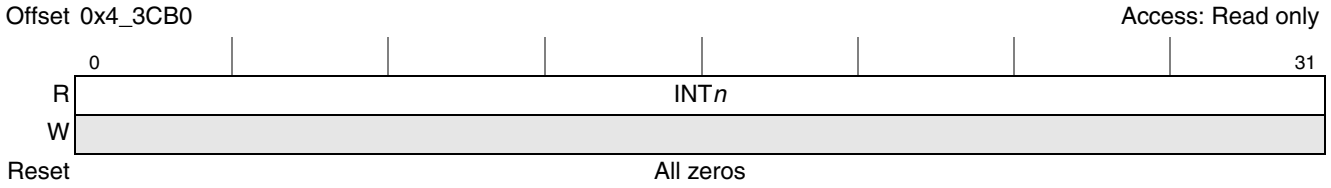


Figure 25-52. Machine Check Summary Register 8 (MCSR8)

Table 25-54 describes MCSR8.

Table 25-57. MCSR8 Field Descriptions

Bits	Name	Description
0–31	INT*n*	Internal interrupts 224–255. Bit 0 represents INT224. 0 The corresponding interrupt is not active or not routed to *mcpn*. 1 The corresponding interrupt is active and is routed to *mcpn* (if the corresponding interrupt level register field is set to *mcp*).

25.3.4.5.19 $\overline{\text{IRQ\_OUT}}$/SoC Interrupt Event Summary Register 0 (IRQSIESR0)

Figure 25-53 shows the IRQSIESR0 fields.

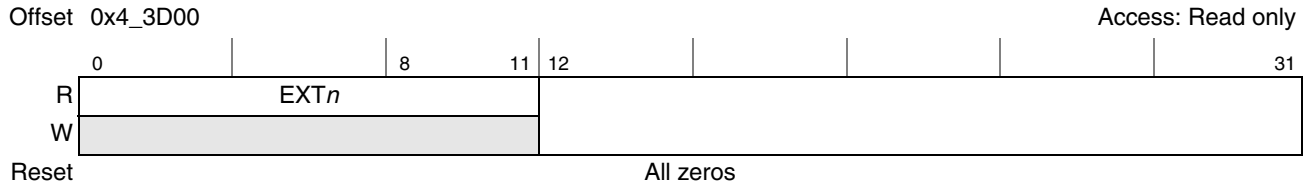


Figure 25-53. $\overline{\text{IRQ\_OUT}}$/SoC Interrupt Event Summary Register 0 (IRQSIESR0)

Table 25-58 describes the IRQSIESR0 fields.

Table 25-58. IRQSIESR0 Field Descriptions

Bits	Name	Description
0–11	EXT*n*	External interrupts 0–11. Bit 0 represents IRQ0. 0 The corresponding interrupt is not active or not routed to *sien* or $\overline{\text{IRQ\_OUT}}$. 1 The corresponding interrupt is active and is routed to *sien* or $\overline{\text{IRQ\_OUT}}$ (if the corresponding interrupt level register field is set to *sie*[0:6] or $\overline{\text{IRQ\_OUT}}$).
12–31	—	Reserved

25.3.4.5.20 $\overline{\text{IRQ\_OUT}}$/SoC Interrupt Event Summary Register 1 (IRQSIESR1)

Figure 25-54 shows the IRQSIESR1 fields.

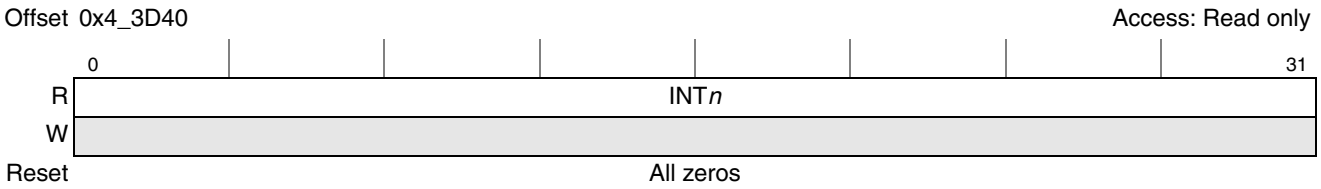


Figure 25-54. $\overline{\text{IRQ\_OUT}}$/SoC Interrupt Event Summary Register 1 (IRQSIESR1)

Table 25-59 describes the IRQSIESR1 fields.

Table 25-59. IRQSIESR1 Field Descriptions

Bits	Name	Description
0–31	INT*n*	Internal interrupts 0–31 status. Bit 0 represents INT0. Bit 31 represents INT31. 0 The corresponding interrupt is not active or not routed to *sien* or $\overline{\text{IRQ\_OUT}}$. 1 The corresponding interrupt is active and is routed to *sien* or $\overline{\text{IRQ\_OUT}}$ (if the corresponding interrupt level register field is set to sie[0:2] or $\overline{\text{IRQ\_OUT}}$).

25.3.4.5.21 $\overline{\text{IRQ\_OUT}}$/SoC Interrupt Event Summary Register 2 (IRQSIESR2)

Figure 25-55 shows the IRQSIESR2 fields.

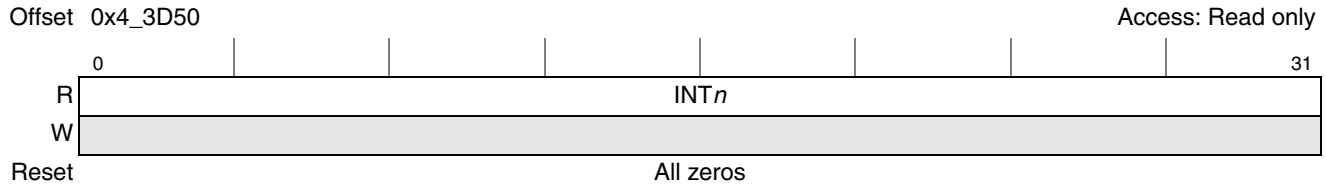


Figure 25-55. $\overline{\text{IRQ\_OUT}}$/SoC Interrupt Event Summary Register 2 (IRQSIESR2)

Table 25-60 describes the IRQSIESR2 fields.

Table 25-60. IRQSIESR2 Field Descriptions

Bits	Name	Description
0–31	INT*n*	Internal interrupts status 32–63. Bit 0 represents INT32. 0 The corresponding interrupt is not active or not routed to *sien* or $\overline{\text{IRQ\_OUT}}$ 1 The corresponding interrupt is active and is routed to *sien* or $\overline{\text{IRQ\_OUT}}$ (if the corresponding interrupt level register field is set to *sie[0:2]* or $\overline{\text{IRQ\_OUT}}$).

25.3.4.5.22 $\overline{\text{IRQ\_OUT}}$/SoC Interrupt Event Summary Register 3 (IRQSIESR3)

Figure 25-56 shows the IRQSIESR3 fields.

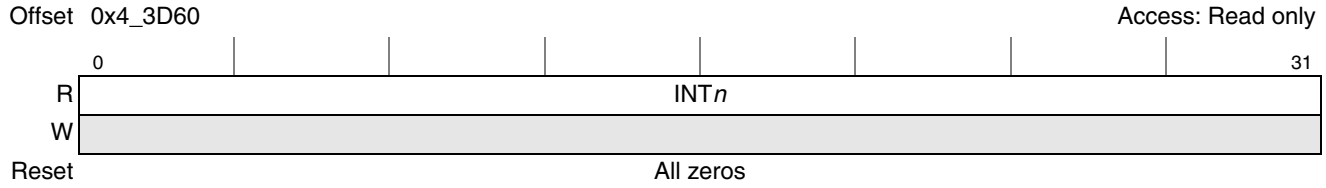


Figure 25-56. $\overline{\text{IRQ\_OUT}}$/SoC Interrupt Event Summary Register 3 (IRQSIESR3)

Table 25-61 describes the IRQSIESR3 fields.

Table 25-61. IRQSIESR3 Field Descriptions

Bits	Name	Description
0–31	INT*n*	Internal interrupts status 64-95. Bit 0 represents INT64. 0 The corresponding interrupt is not active or not routed to *sien* or $\overline{\text{IRQ\_OUT}}$ 1 The corresponding interrupt is active and is routed to *sien* or $\overline{\text{IRQ\_OUT}}$ (if the corresponding interrupt level register field is set to *sie*[0:2] or $\overline{\text{IRQ\_OUT}}$).

25.3.4.5.23 $\overline{\text{IRQ\_OUT}}$/Soc Interrupt Event Summary Register 4 (IRQSIESR4)

Figure 25-57 shows the IRQSIESR4 fields.

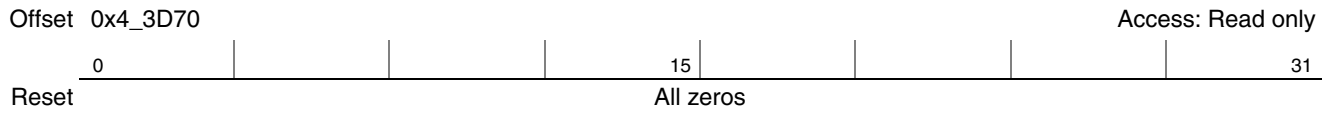


Figure 25-57. $\overline{\text{IRQ\_OUT}}$/SoC Interrupt Event Summary Register 4 (IRQSIESR4)

Table 25-62 describes the IRQSIESR4 fields.

25.3.4.5.24 $\overline{\text{IRQ\_OUT}}$/SoC Interrupt Event Summary Register 5 (IRQSIESR5)

Figure 25-56 shows the IRQSIESR5 fields.

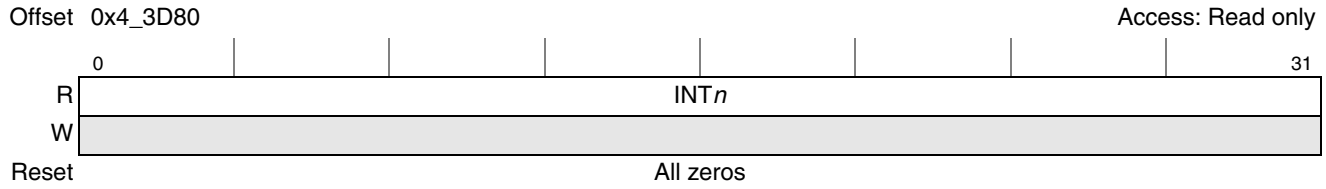


Figure 25-58. $\overline{\text{IRQ\_OUT}}$/SoC Interrupt Event Summary Register 5 (IRQSIESR5)

Table 25-61 describes the IRQSIESR5 fields.

Table 25-63. IRQSIESR5 Field Descriptions

Bits	Name	Description
0–31	INT*n*	Internal interrupts status 128-159. Bit 0 represents INT128. 0 The corresponding interrupt is not active or not routed to *sien* or $\overline{\text{IRQ\_OUT}}$ 1 The corresponding interrupt is active and is routed to *sien* or $\overline{\text{IRQ\_OUT}}$ (if the corresponding interrupt level register field is set to *sie*[0:2] or $\overline{\text{IRQ\_OUT}}$).

25.3.4.5.25 $\overline{\text{IRQ\_OUT}}$/SoC Interrupt Event Summary Register 6 (IRQSIESR6)

Figure 25-56 shows the IRQSIESR6 fields.

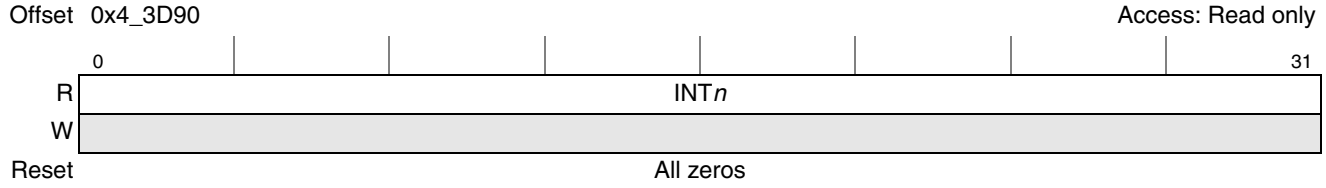


Figure 25-59. $\overline{\text{IRQ\_OUT}}$/SoC Interrupt Event Summary Register 6 (IRQSIESR6)

Table 25-61 describes the IRQSIESR6 fields.

Table 25-64. IRQSIESR6 Field Descriptions

Bits	Name	Description
0–31	INT*n*	Internal interrupts status 160-191. Bit 0 represents INT160. 0 The corresponding interrupt is not active or not routed to *sien* or $\overline{\text{IRQ\_OUT}}$ 1 The corresponding interrupt is active and is routed to *sien* or $\overline{\text{IRQ\_OUT}}$ (if the corresponding interrupt level register field is set to *sie*[0:2] or $\overline{\text{IRQ\_OUT}}$).

25.3.4.5.26 $\overline{\text{IRQ\_OUT}}$/SoC Interrupt Event Summary Register 7 (IRQSIESR7)

Figure 25-56 shows the IRQSIESR7 fields.

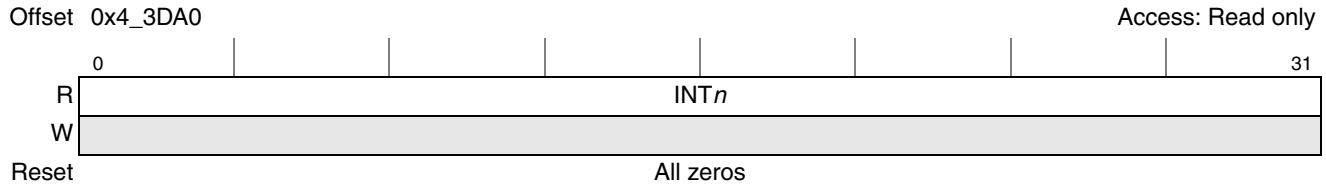


Figure 25-60. $\overline{\text{IRQ\_OUT}}$/SoC Interrupt Event Summary Register 7 (IRQSIESR7)

Table 25-61 describes the IRQSIESR7 fields.

Table 25-65. IRQSIESR7 Field Descriptions

Bits	Name	Description
0–31	INT*n*	Internal interrupts status 192-223. Bit 0 represents INT192. 0 The corresponding interrupt is not active or not routed to *sien* or $\overline{\text{IRQ\_OUT}}$ 1 The corresponding interrupt is active and is routed to *sien* or $\overline{\text{IRQ\_OUT}}$ (if the corresponding interrupt level register field is set to *sie*[0:2] or $\overline{\text{IRQ\_OUT}}$).

25.3.4.5.27 $\overline{\text{IRQ\_OUT}}$/SoC Interrupt Event Summary Register 8 (IRQSIESR8)

Figure 25-56 shows the IRQSIESR8 fields.

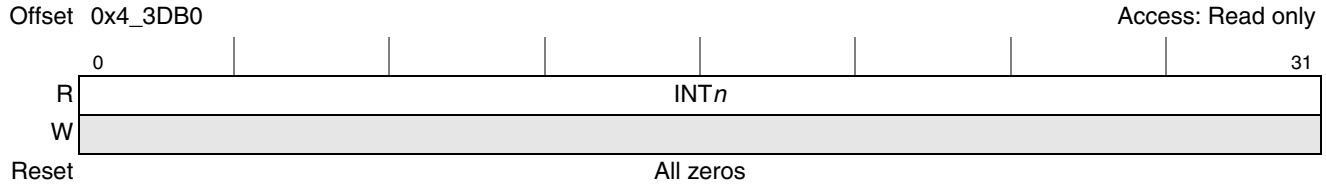


Figure 25-61. $\overline{\text{IRQ\_OUT}}$/SoC Interrupt Event Summary Register 8 (IRQSIESR8)

Table 25-61 describes the IRQSIESR8 fields.

Table 25-66. IRQSIESR8 Field Descriptions

Bits	Name	Description
0–31	INT*n*	Internal interrupts status 224-255. Bit 0 represents INT224. 0 The corresponding interrupt is not active or not routed to *sien* or $\overline{\text{IRQ\_OUT}}$ 1 The corresponding interrupt is active and is routed to *sien* or $\overline{\text{IRQ\_OUT}}$ (if the corresponding interrupt level register field is set to *sie*[0:2] or $\overline{\text{IRQ\_OUT}}$).

25.3.5 Global Message Registers

The MPIC supports two banks (A and B) of eight 32-bit message registers. The following registers support the message register interrupts:

- Section 25.3.5.1, “Message Registers (MSGRA0–MSGRA7, MSGRB0–MSGRB7)”
- Section 25.3.5.2, “Message Enable Registers A and B (MERA, MERB)”
- Section 25.3.5.3, “Message Status Register (MSRA, MSRB)”

- Section 25.3.7.7, "Messaging Interrupt Vector/Priority Registers (MIVPRA0–MIVPRA7, MIVPRB0–MIVPRB7)"
- Section 25.3.7.8, "Messaging Interrupt Destination Registers (MIDRA0–MIDRA7, MIDRB0–MIDRB7)"

Writing a message register (MSGR*x*0–MSGR*x*7) causes a messaging interrupt as directed by the other message registers listed above. Reading a message register clears the messaging interrupt. Note that a messaging interrupt can also be cleared by writing a one to the corresponding status field of the MPIC message status register (MSR), shown in Figure 25-64.

25.3.5.1 Message Registers (MSGRA0–MSGRA7, MSGRB0–MSGRB7)

The message registers (MSGR*x*0–MSGR*x*7), shown in Figure 25-62, can contain a 32-bit message.

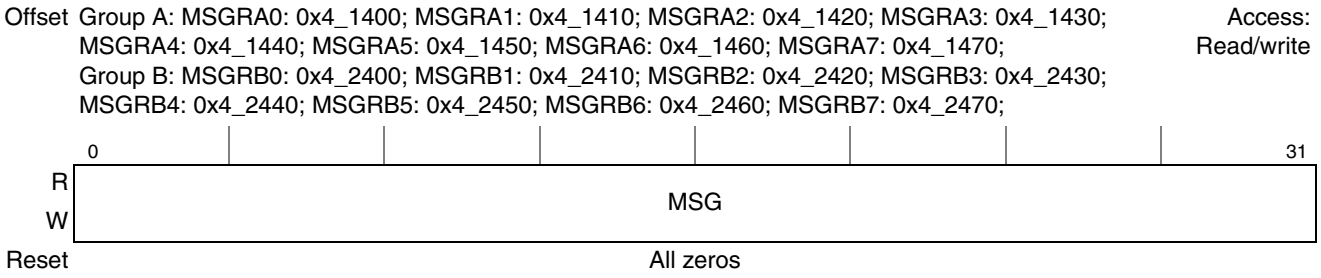


Figure 25-62. Message Registers (MSGRs)

Table 25-67 describes the MSGR registers.

Table 25-67. MSGR*n* Field Descriptions

Bits	Name	Description
0–31	MSG	Message. Contains the 32-bit message data.

25.3.5.2 Message Enable Registers A and B (MERA, MERB)

The MERs, shown in Figure 25-63, contain the enable bits for each message register. The enable bit must be set to enable interrupt generation when the corresponding message register is written.

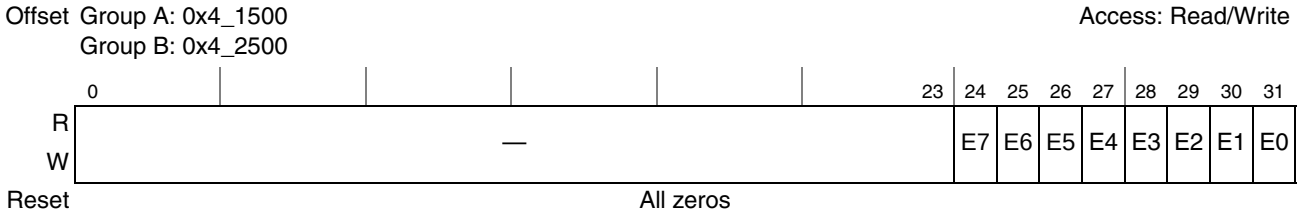


Figure 25-63. Message Enable Registers (MERA and MERB)

Table 25-68 describes the MER fields.

Table 25-68. MER Field Descriptions

Bits	Name	Description
0–23	—	Reserved
24–32	E*n*	Enable 7–enable 0. Used to enable interrupt generation for MSGR*n* (where *n* = 0–7). 0 Interrupt generation for MSGR*n* disabled. 1 Interrupt generation for MSGR*n* enabled.

25.3.5.3 Message Status Register (MSRA, MSRB)

The MSRs shown in Figure 25-64 contain status bits for each message register. A status bit is set when the corresponding messaging interrupt is active. Writing a 1 to a status bit clears the corresponding message interrupt and the status bit.

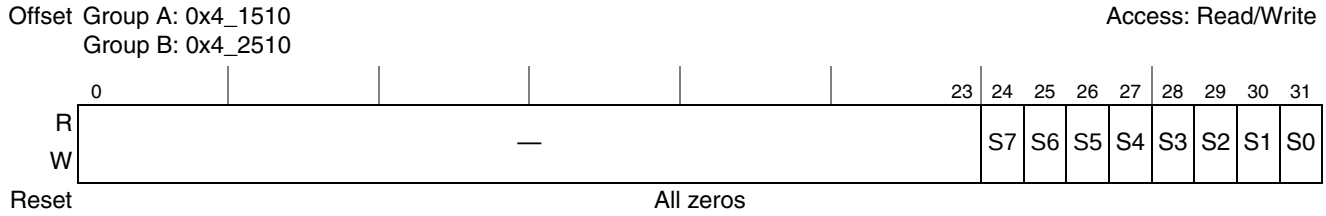


Figure 25-64. Message Status Register (MSRA and MSRB)

Table 25-69 describes the MSR fields.

Table 25-69. MSR Field Descriptions

Bits	Name	Description
0–23	—	Reserved
24-32	S*n*	Status 7–status 0. Reports status of messaging interrupt *n*. Writing a 1 clears this field. 0 Messaging interrupt *n* is not active. 1 Messaging interrupt *n* is active.

25.3.6 Global Shared Message Signaled Interrupt Registers

This section contains the description of all of the shared message signaled interrupt registers (MSIRs). The shared message signaled interrupt structures allows programs to interrupt each other by simply writing to these shared memory-mapped registers in the MPIC. There are four banks for MSIRs—A, B, C, and D. These are intended for Message Signalled Interrupts on PCIe root complex. Each of the sixteen MSIRs for each bank can be thought of as collecting interrupts from 32 different memory-mapped writes that can cause shared message signaled interrupts.

There are two sources on each bank to set the message signaled interrupt registers. One (MSIIRa) can select up to eight registers of thirty-two sources each and the other (MSIIR1a) can select up to sixteen registers of thirty-two sources each. The first eight registers are shared by both sources and these are ORed with those of MSISRb and MSIRbn.

25.3.6.1 Shared Message Signaled Interrupt Registers (MSIR*a*0–MSIR*a*15)

Each bank of sixteen MSIRs indicates which of the up to 32 interrupt sources sharing a message register have pending interrupts. These registers are cleared when read. A write to these registers has no effect.

Offset MSIRA0: 0x4_1600; MSIRA1: 0x4_1610; MSIRA2: 0x4_1620; MSIRA3: 0x4_1630; Access: Read only
MSIRA4: 0x4_1640; MSIRA5: 0x4_1650; MSIRA6: 0x4_1660; MSIRA7: 0x4_1670,
MSIRA8: 0x4_1680; MSIRA9: 0x4_1690; MSIRA10: 0x4_16A0; MSIRA11: 0x4_16B0;
MSIRA12: 0x4_16C0; MSIRA13: 0x4_16D0; MSIRA14: 0x4_16E0; MSIRA15: 0x4_16F0,
MSIRB0: 0x4_1800; MSIRB1: 0x4_1810; MSIRB2: 0x4_1820; MSIRB3: 0x4_1830;
MSIRB4: 0x4_1840; MSIRB5: 0x4_1850; MSIRB6: 0x4_1860; MSIRB7: 0x4_1870
MSIRB8: 0x4_1880; MSIRB9: 0x4_1890; MSIRB10: 0x4_18A0; MSIRB11: 0x4_18B0;
MSIRB12: 0x4_18C0; MSIRB13: 0x4_18D0; MSIRB14: 0x4_18E0; MSIRB15: 0x4_18F0,
MSIRC0: 0x4_1A00; MSIRC1: 0x4_1A10; MSIRC2: 0x4_1A20; MSIRC3: 0x4_1A30;
MSIRC4: 0x4_1A40; MSIRC5: 0x4_1A50; MSIRC6: 0x4_1A60; MSIRC7: 0x4_1A70
MSIRC8: 0x4_1A80; MSIRC9: 0x4_1A90; MSIRC10: 0x4_1AA0; MSIRC11: 0x4_1AB0;
MSIRC12: 0x4_1AC0; MSIRC13: 0x4_1AD0; MSIRC14: 0x4_1AE0; MSIRC15: 0x4_1AF0,
MSIRD0: 0x4_1C00; MSIRD1: 0x4_1C10; MSIRD2: 0x4_1C20; MSIRD3: 0x4_1C30;
MSIRD4: 0x4_1C40; MSIRD5: 0x4_1C50; MSIRD6: 0x4_1C60; MSIRD7: 0x4_1C70,
MSIRD8: 0x4_1C80; MSIRD9: 0x4_1C90; MSIRD10: 0x4_1CA0; MSIRD11: 0x4_1CB0;
MSIRD12: 0x4_1CC0; MSIRD13: 0x4_1CD0; MSIRD14: 0x4_1CE0; MSIRD15: 0x4_1CF0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SH31	SH30	SH29	SH28	SH27	SH26	SH25	SH24	SH23	SH22	SH21	SH20	SH19	SH18	SH17	SH16
W																
Reset	All zeros															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	SH15	SH14	SH13	SH12	SH11	SH10	SH9	SH8	SH7	SH6	SH5	SH4	SH3	SH2	SH1	SH0
W																
Reset	All zeros															

Figure 25-65. Shared Message Signaled Interrupt Registers (MSIR*an*)

Table 25-70 describes the bits of the MSIRs.

Table 25-70. MSIR*n* Field Descriptions

Bits	Name	Description
n	SH*n*	Message sharer *n* has a pending interrupt.

25.3.6.2 Shared Message Signaled Interrupt Status Register *a* (MSISR*a*)

MSISR*a*, shown in Figure 25-66, contains the status bits for the shared message signaled interrupts. A status bit is set when the corresponding MSIR*a* has an active interrupt. The status bit is 0 if all the corresponding shared interrupt sources are cleared for that MSIR*a*.

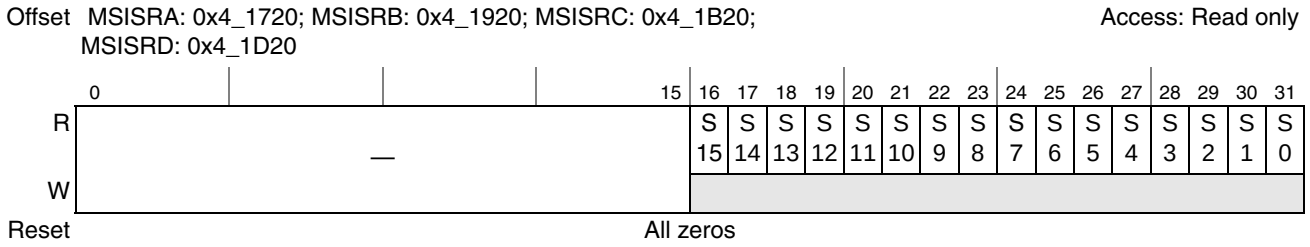


Figure 25-66. Shared Message Signaled Interrupt Status Register *a* (MSISR*a*)

Table 25-71 describes the bits of the MSISR*a*.

Table 25-71. MSISR Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	S*n*	Status *n*. 0 MSIR*n* is not active. 1 MSIR*n* has an active interrupt.

25.3.6.3 Shared Message Signaled Interrupt Index Register (MSIIR)

MSIIR*a*, shown in Figure 25-66, provides the mechanism for setting an interrupt in the MSIR*a*s. When MSIIR*a* is written, MSIIR*a*[SRS] selects the register in which an interrupt bit is to be set MSIIR*a*[IBS] selects the shared interrupt field in the selected MSIR*a* register. MSIIR*a* is primarily intended to support PCI Express MSIs.

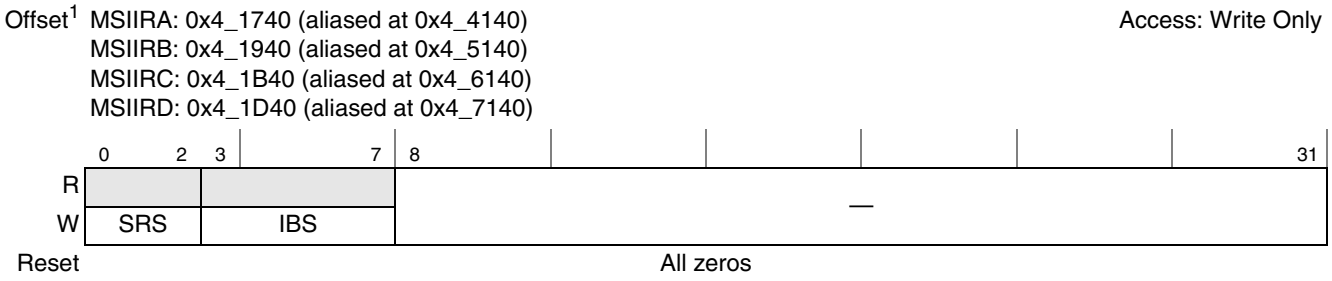


Figure 25-67. Shared Message Signaled Interrupt Index Register (MSIIR)

[1] MSIIRA, MSIIRB, MSIIRC, and MSIIRD are aliased to offsets 0x4_4140, 0x4_5140, 0x4_6140, and 0x4_7140 respectively. See rows beginning at “Shared Message Signaled Interrupt Index Registers (Alias)” in Table 25-5, “MPIC Register Address Map.”

Table 25-72 describes the bits of the MSIIR*a*.

Table 25-72. MSIIR*a* Field Descriptions

Bits	Name	Description
0–2	SRS	Shared interrupt register select. Selects the MSIR to be written. 000 MSIR 0 001 MSIR 1 010 MSIR 2 ... 111 MSIR 7
3–7	IBS	Interrupt bit select. Selects the bit to set in the MSIR. 00000 Set field SH0 (bit 31) 00001 Set field SH1 (bit 30) 00010 Set field SH2 (bit 29) ... 11111 Set field SH31 (bit 0)
8–31	—	Reserved

25.3.6.4 Shared Message Signaled Interrupt Index Register (MSIIR1)

MSIIR1*a*, shown in Figure 25-66, provides the mechanism for setting an interrupt in the MSIR*a*s. When MSIIR1*a* is written, MSIIR1*a*[SRS] selects the register in which an interrupt bit is to be set MSIIR1*a*[IBS] selects the shared interrupt field in the selected MSIR*a* register. MSIIR1*a* is primarily intended to support PCI Express MSIs.

Offset[1] MSIIR1A: 0x4_4148; MSIIR1B: 0x4_5148; MSIIR1C: 0x4_6148; MSIIR1D: 0x4_7148 Access: Write Only

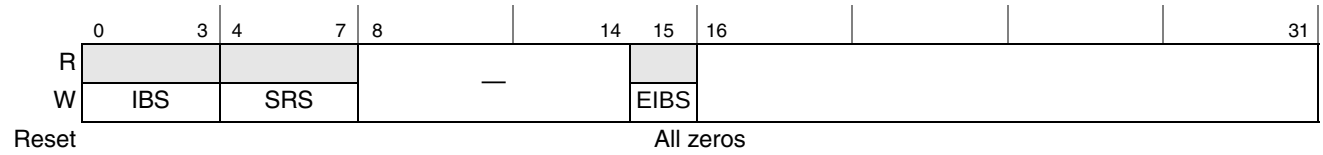


Figure 25-68. Shared Message Signaled Interrupt Index Register 1(MSIIR1)

1

Table 25-72 describes the bits of the MSIIR1*a*.

Table 25-73. MSIIR1*a* Field Descriptions

Bits	Name	Description
0–3	IBS	Interrupt bit select. Selects the bit to set in the MSIR. This is a five bit field combined with MSIIR1[EIBS]. The x below represents the most significant bit MSIIR1[EIBS] x0000 Set field SH0 or SH16 (bit 31 or bit 15) x0001 Set field SH1 or SH17 (bit 29 or bit 14) ... x1111 Set field SH15 or SH31 (bit 16 or bit 0) For example: 00000 Set field SH0 (bit 31) 10001 Set field SH17 (bit 14)
4–7	SRS	Shared interrupt register select. Selects the MSIR to be written. 0000 MSIR 0 0001 MSIR 1 0010 MSIR 2 ... 1111 MSIR 15
8–14	—	Reserved
15	EIBS	Interrupt bit select. Selects the bit to set in the MSIR. 0 Set field SH*0-15* (lower 16 bits of MSIR*n*) 1 Set field SH*16-31*(upper 16 bits of MSIR*n*)
16–31	—	Reserved

25.3.6.5 Shared Message Signaled Coalescing Registers (MSICR*a*0–MSICR*a*15)

Each bank of sixteen MSICRs indicates which of the up to 32 interrupt sources sharing a message register have coalescing setup. A non-zero write to this register enables coalescing on the corresponding MSI. A write of zero to the same register disables MSI coalescing on the corresponding MSI.

Offset MSICRA0: 0x4_4180; MSICRA1: 0x4_4184; MSICRA2: 0x4_4188; MSICRA3: 0x4_418C; MSICRA4: 0x4_4190; MSICRA5: 0x4_4194; MSICRA6: 0x4_4198; MSICRA7: 0x4_419C, MSICRA8: 0x4_41A0; MSICRA9: 0x4_41A4; MSICRA10: 0x4_41A8; MSICRA11: 0x4_41AC; MSICRA12: 0x4_41B0; MSICRA13: 0x4_41B4; MSICRA14: 0x4_41B8; MSICRA15: 0x4_41BC, MSICRB0: 0x4_5180; MSICRB1: 0x4_5184; MSICRB2: 0x4_5188; MSICRB3: 0x4_518C; MSICRB4: 0x4_5190; MSICRB5: 0x4_5194; MSICRB6: 0x4_5198; MSICRB7: 0x4_519C, MSICRB8: 0x4_51A0; MSICRB9: 0x4_51A4; MSICRB10: 0x4_51A8; MSICRB11: 0x4_51AC; MSICRB12: 0x4_51B0; MSICRB13: 0x4_51B4; MSICRB14: 0x4_51B8; MSICRB15: 0x4_51BC, MSICRC0: 0x4_6180; MSICRC1: 0x4_6184; MSICRC2: 0x4_6188; MSICRC3: 0x4_618C; MSICRC4: 0x4_6190; MSICRC5: 0x4_6194; MSICRC6: 0x4_6198; MSICRC7: 0x4_619C, MSICRC8: 0x4_61A0; MSICRC9: 0x4_61A4; MSICRC10: 0x4_61A8; MSICRC11: 0x4_61AC; MSICRC12: 0x4_61B0; MSICRC13: 0x4_61B4; MSICRC14: 0x4_61B8; MSICRC15: 0x4_61BC, MSICRD0: 0x4_7180; MSICRD1: 0x4_7184; MSICRD2: 0x4_7188; MSICRD3: 0x4_718C; MSICRD4: 0x4_7190; MSICRD5: 0x4_7194; MSICRD6: 0x4_7198; MSICRD7: 0x4_719C, MSICRD8: 0x4_71A0; MSICRD9: 0x4_71A4; MSICRD10: 0x4_71A8; MSICRD11: 0x4_71AC; MSICRD12: 0x4_71B0; MSICRD13: 0x4_71B4; MSICRD14: 0x4_71B8; MSICRD15: 0x4_71BC,

Access: Read/Write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	CB31	CB30	CB29	CB28	CB27	CB26	CB25	CB24	CB23	CB22	CB21	CB20	CB 19	CB 18	CB 17	CB 16
Reset	All zeros															

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R W	CB15	CB14	CB13	CB12	CB11	CB10	CB9	CB8	CB7	CB6	CB5	CB4	CB 3	CB 2	CB 1	CB 0
Reset	All zeros															

Figure 25-69. Shared Message Signaled Coalescing Registers (MSICR*an*)

Table 25-70 describes the bits of the MSIRs.

Table 25-74. MSICR*n* Field Descriptions

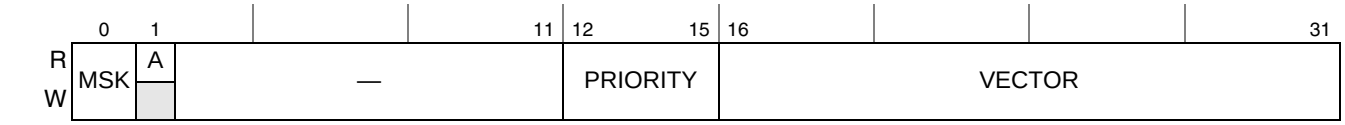
Bits	Name	Description
n	CB*n*	Message signaled interrupt *n* has a coalescing set. 0 = event *n* is not included in coalescing 1 = event *n* is included in coalescing

25.3.6.6 Shared Message Signaled Interrupt Vector/Priority Register *a* 0–15 (MSIVPR*a*0-MSIVPR*a*15)

The MSIVPR*an*s, shown in Figure 25-70, have the same fields and format as the GTVPRs.

Offset MSIVPRA0: 0x5_1C00; MSIVPRA1: 0x5_1C20; MSIVPRA2: 0x5_1C40; MSIVPRA3: 0x5_1C60; MSIVPRA4: 0x5_1C80; MSIVPRA5: 0x5_1CA0; MSIVPRA6: 0x5_1CC0; MSIVPRA7: 0x5_1CE0
MSIVPRB0: 0x5_1D00; MSIVPRB1: 0x5_1D20; MSIVPRB2: 0x5_1D40; MSIVPRB3: 0x5_1D60; MSIVPRB4: 0x5_1D80; MSIVPRB5: 0x5_1DA0; MSIVPRB6: 0x5_1DC0; MSIVPRB7: 0x5_1DE0
MSIVPRC0: 0x5_1E00; MSIVPRC1: 0x5_1E20; MSIVPRC2: 0x5_1E40; MSIVPRC3: 0x5_1E60; MSIVPRC4: 0x5_1E80; MSIVPRC5: 0x5_1EA0; MSIVPRC6: 0x5_1EC0; MSIVPRC7: 0x5_1EE0
MSIVPRD0: 0x5_1F00; MSIVPRD1: 0x5_1F20; MSIVPRD2: 0x5_1F40; MSIVPRD3: 0x5_1F60; MSIVPRD4: 0x5_1F80; MSIVPRD5: 0x5_1FA0; MSIVPRD16: 0x5_1FC0; MSIVPRD7: 0x5_1FE0
MSIVPRA8: 0x5_2000; MSIVPRA9: 0x5_2020; MSIVPRA10: 0x5_2040; MSIVPRA11: 0x5_2060; MSIVPRA12: 0x5_2080; MSIVPRA13: 0x5_20A0; MSIVPRA14: 0x5_20C0; MSIVPRA15: 0x5_20E0
MSIVPRB8: 0x5_2100; MSIVPRB9: 0x5_2120; MSIVPRB10: 0x5_2140; MSIVPRB11: 0x5_2160; MSIVPRB12: 0x5_2180; MSIVPRB13: 0x5_21A0; MSIVPRB14: 0x5_21C0; MSIVPRB15: 0x5_21E0
MSIVPRC8: 0x5_2200; MSIVPRC9: 0x5_2220; MSIVPRC10: 0x5_2240; MSIVPRC11: 0x5_2260; MSIVPRC12: 0x5_2280; MSIVPRC13: 0x5_22A0; MSIVPRC14: 0x5_22C0; MSIVPRC15: 0x5_22E0
MSIVPRD8: 0x5_2300; MSIVPRD9: 0x5_2320; MSIVPRD10: 0x5_2340; MSIVPRD11: 0x5_2360; MSIVPRD12: 0x5_2380; MSIVPRD13: 0x5_23A0; MSIVPRD14: 0x5_23C0; MSIVPRD15: 0x5_23E0

Access: Read/Write



Reset 1000_0000_0000_0000_0000_0000_0000_0000

Figure 25-70. Shared Message Signaled Interrupt Vector/Priority Register *a* (MSIVPR*a*s)

Table 25-75 describes the bits of the MSIVPRs.

Table 25-75. MSIVPRn Field Descriptions

Bits	Name	Description
0	MSK	Mask. Mask interrupts from this source. MSK affects interrupts routed to *int*, *cint*, *mcp*, *sie*[0:2], and $\overline{\text{IRQ\_OUT}}$. 0 An interrupt request is generated if the corresponding IPR bit is set. 1 Further interrupts from this source are disabled.
1	A	Activity. Indicates an interrupt has been requested or is in service. The VECTOR and PRIORITY values should not be changed while this bit is set. Affects only interrupts routed to *int*. 0 No current interrupt activity associated with this source. 1 The interrupt field for this source is set in the IPR or ISR.
2–11	—	Reserved
12–15	PRIORITY	Priority. Specifies the interrupt priority. The lowest priority is 0 and the highest priority is 15. A priority level of 0 inhibits signalling of this interrupt to the core. Affects only interrupts routed to *int*.
16–31	VECTOR	Vector (Affects only interrupts routed to *int*). Contains the value returned when IACK is read and this interrupt resides in the corresponding interrupt request register (IRR) for that core, as shown in Figure 25-85.

25.3.6.7 Shared Message Signaled Interrupt Destination Registers *a* 0–15 (MSIDR*a*0-MSIDR*a*15)

The MSIDRs, shown in Figure 25-71, contain the destination fields for shared message signaled interrupts. Only one destination bit may be set; otherwise, behavior is undefined.

Offset MSIDRA0: 0x5_1C10; MSIDRA1: 0x5_1C30; MSIDRA2: 0x5_1C50; MSIDRA3: 0x5_1C70; Access: Read/Write
MSIDRA4: 0x5_1C90; MSIDRA5: 0x5_1CB0; MSIDRA6: 0x5_1CD0; MSIDRA7: 0x5_1CF0
MSIDRB0: 0x5_1D10; MSIDRB1: 0x5_1D30; MSIDRB2: 0x5_1D50; MSIDRB3: 0x5_1D70;
MSIDRB4: 0x5_1D90; MSIDRB5: 0x5_1DB0; MSIDRB6: 0x5_1DD0; MSIDRB7: 0x5_1DF0
MSIDRC0: 0x5_1E10; MSIDRC1: 0x5_1E30; MSIDRC2: 0x5_1E50; MSIDRC3: 0x5_1E70;
MSIDRC4: 0x5_1E90; MSIDRC5: 0x5_1EB0; MSIDRC6: 0x5_1ED0; MSIDRC7: 0x5_1EF0
MSIDRD0: 0x5_1F10; MSIDRD1: 0x5_1F30; MSIDRD2: 0x5_1F50; MSIDRD3: 0x5_1F70;
MSIDRD4: 0x5_1F90; MSIDRD5: 0x5_1FB0; MSIDRD6: 0x5_1FD0; MSIDRD7: 0x5_1FF0
MSIDRA8: 0x5_2010; MSIDRA9: 0x5_2030; MSIDRA10: 0x5_2050; MSIDRA11: 0x5_2070;
MSIDRA12: 0x5_2090; MSIDRA13: 0x5_20B0; MSIDRA14: 0x5_20D0; MSIDRA15: 0x5_20F0
MSIDRB8: 0x5_2110; MSIDRB9: 0x5_2130; MSIDRB10: 0x5_2150; MSIDRB11: 0x5_2170;
MSIDRB12: 0x5_2190; MSIDRB13: 0x5_21B0; MSIDRB14: 0x5_21D0; MSIDRB15: 0x5_21F0
MSIDRC8: 0x5_2210; MSIDRC9: 0x5_2230; MSIDRC10: 0x5_2250; MSIDRC11: 0x5_2270;
MSIDRC12: 0x5_2290; MSIDRC13: 0x5_22B0; MSIDRC14: 0x5_22D0; MSIDRC15: 0x5_22F0
MSIDRD8: 0x5_2310; MSIDRD9: 0x5_2330; MSIDRD10: 0x5_2350; MSIDRD11: 0x5_2370;
MSIDRD12: 0x5_2390; MSIDRD13: 0x5_23B0; MSIDRD14: 0x5_23D0; MSIDRD15: 0x5_23F0

	0																								23	24	25	26	27	28	29	30	31
R W	—																									P7	P6	P5	P4	P3	P2	P1	P0
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	1

Figure 25-71. Shared Message Signaled Interrupt Destination Registers *a* (MSIDR*a*s)

Table 25-76 describes MSIDR*a*s fields.

Table 25-76. MSIDR*a*s Field Descriptions

Bits	Name	Description
0–23	—	Reserved
24	P7	Processor core 7. Indicates whether processor core 7 receives the interrupt. 0 Processor core 7 does not receive this interrupt. 1 Directs the interrupt to processor core 7.
25	P6	Processor core 6. Indicates whether processor core 6 receives the interrupt. 0 Processor core 6 does not receive this interrupt. 1 Directs the interrupt to processor core 6.
26	P5	Processor core 5. Indicates whether processor core 5 receives the interrupt. 0 Processor core 5 does not receive this interrupt. 1 Directs the interrupt to processor core 5.
27	P4	Processor core 4. Indicates whether processor core 4 receives the interrupt. 0 Processor core 4 does not receive this interrupt. 1 Directs the interrupt to processor core 4.
28	P3	Processor core 3. Indicates whether processor core 3 receives the interrupt. 0 Processor core 3 does not receive this interrupt. 1 Directs the interrupt to processor core 3.

Table 25-76. MSIDR*a*s Field Descriptions (continued)

Bits	Name	Description
29	P2	Processor core 2. Indicates whether processor core 2 receives the interrupt. 0 Processor core 2 does not receive this interrupt. 1 Directs the interrupt to processor core 2.
30	P1	Processor core 1. Indicates whether processor core 1 receives the interrupt. 0 Processor core 1 does not receive this interrupt. 1 Directs the interrupt to processor core 1.
31	P0	Processor core 0. Indicates whether processor core 0 receives the interrupt. 0 Processor core 0 does not receive this interrupt. 1 Directs the interrupt to processor core 0. The default destination is for core 0 to receive this shared message signaled interrupt after the MPIC is reset.

25.3.7 Global Interrupt Source Configuration Registers

The interrupt source configuration registers control the source and destinations of interrupts, specifying parameters such as the interrupting event, signal polarity, and relative priority.

The global timer, interprocessor, messaging, and shared message signaled destination register support only processor core options. That is, they cannot be routed to *cint, mcp, sie*[0:2] or to $\overline{\text{IRQ\_OUT}}$.

Only the global timer, interprocessor, and edge-triggered external interrupts are multicasting, so only these interrupts allow more than one destination bit to be specified.

Figure 25-72 shows a comparison of vector/priority registers.

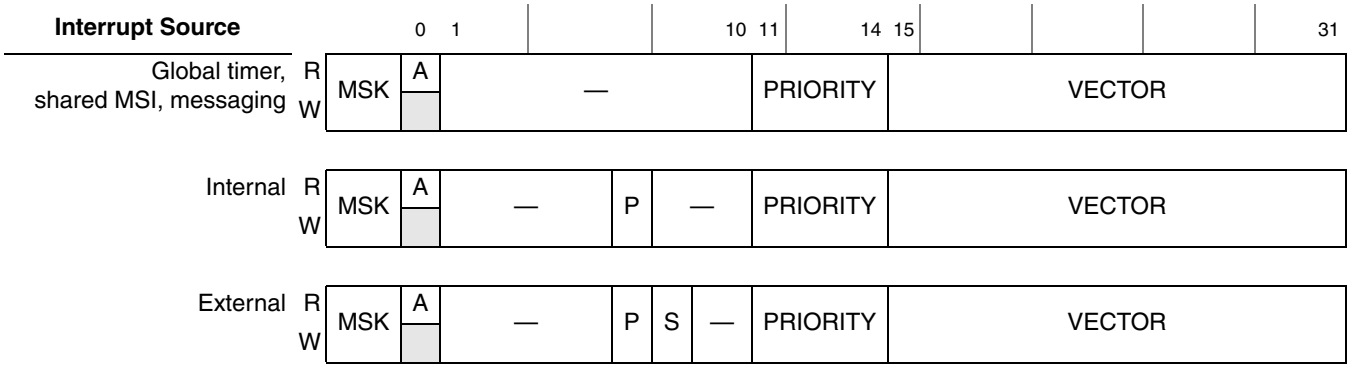


Figure 25-72. Vector/Priority Register Summary

Table 25-77 describes the vector priority fields. Note the following:

- The MSK, A, PRIORITY, and VECTOR fields are defined by the OpenPIC specification and have meaning only for interrupts routed to the *int* signal. However, for the internal and external interrupt sources, the MSK field is also repected for interrupts routed to the *int* , *cint*, *mcp*, *sie*[0:2], and $\overline{\text{IRQ\_OUT}}$ signals.
- The polarity field, P, is provided to indicate whether the signals from the corresponding source are active high or low.

- The sense field, S, is provided to allow external interrupt sources to be configured as level-sensitive so they can be routed to either *cint, mcp, sie*[0:2] or $\overline{\text{IRQ\_OUT}}$.

Table 25-77. Vector/Priority Register Field Descriptions

Bits	Name	Description
0	MSK	Mask. MSK affects interrupts routed to *int*, *cint*, *mcp*, *sie*[0:2], and $\overline{\text{IRQ\_OUT}}$. 0 An interrupt request (to *int*) is generated if the corresponding IPR bit is set. 1 Further interrupts routed to *int* from this source are disabled.
1	A	Activity. Indicates an interrupt has been requested or is in service. The VECTOR and PRIORITY values should not be changed while this bit is set. Affects only interrupts routed to *int*. 0 No current interrupt activity associated with this source. 1 The interrupt field for this source is set in the IPR or ISR.
2–7	—	Reserved
8	P	Polarity. (internal and external interrupts only) Specifies the polarity for the external and internal interrupt inputs. 0 Polarity is active-low or negative edge triggered. 1 Polarity is active-high or positive edge-triggered.
9	S	Sense. (external interrupts only) Specifies the sense for external interrupts. 0 The external interrupt is edge sensitive. 1 The external interrupt is level sensitive. This setting must be used to direct the interrupt to $\overline{\text{IRQ\_OUT}}$, *sie*, *cint,* or *mcp*.
10–11	—	Reserved
12–15	PRIORITY	Priority. Specifies the interrupt priority. The lowest priority is 0 and the highest priority is 15. A priority level of 0 inhibits signalling of this interrupt to the core. Affects only interrupts routed to *int*.
16–31	VECTOR	Vector (Affects only interrupts routed to *int*). Contains the value returned when IACK is read and this interrupt resides in the corresponding interrupt request register (IRR) for that core, as shown in Figure 25-86.

25.3.7.1 External Interrupt Vector/Priority Registers (EIVPR0–EIVPR11)

The EIVPRs, shown in Figure 25-73, contain polarity and sense fields for the external interrupts, that is, those caused by the assertion of any of IRQ[0:11].

Offset EIVPR0: 0x5_0000; EIVPR1: 0x5_0020; EIVPR2: 0x5_0040; EIVPR3: 0x5_0060; EIVPR4: 0x5_0080; EIVPR5: 0x5_00A0; EIVPR6: 0x5_00C0; EIVPR7: 0x5_00E0; EIVPR8: 0x5_0100; EIVPR9: 0x5_0120; EIVPR10: 0x5_0140; EIVPR11: 0x5_0160 — Access: Read/write

	0	1	2					7	8	9	10	11	12			15	16															31
R	MSK	A	—						P	S	—		PRIORITY				VECTOR															
W																																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 25-73. External Interrupt Vector/Priority Registers (EIVPR0—EIVPR11)

Table 25-78 describes the EIVPR fields.

Table 25-78. EIVPR*n* Field Descriptions

Bits	Name	Description
0	MSK	Mask. Mask interrupts from this source. MSK affects interrupts routed to *int*, *cint*, *mcp*, *sie*[0:2], and $\overline{\text{IRQ\_OUT}}$. 0 An interrupt request is generated if the corresponding IPR bit is set. 1 Further interrupts from this source are disabled.
1	A	Activity. Indicates an interrupt has been requested or is in service. The VECTOR and PRIORITY values should not be changed while this bit is set. Affects only interrupts routed to *int*. 0 No current interrupt activity associated with this source. 1 The interrupt field for this source is set in the IPR or ISR.
2–7	—	Reserved
8	P	Polarity. Specifies the polarity for the external interrupt. 0 Polarity is active-low or negative edge-triggered. 1 Polarity is active-high or positive edge-triggered.
9	S	Sense. Specifies the sense for external interrupts. 0 The external interrupt is edge sensitive. 1 The external interrupt is level sensitive. This setting must be used to direct the interrupt to $\overline{\text{IRQ\_OUT}}$, *sie, mcp* or *cint*. Note: If an IRQ*n* signal is used to receive INT*x* signals from one of the PCI Express ports as a root complex, the sense (S) must be set to be level-sensitive.
10–11	—	Reserved
12–15	PRIORITY	Priority. Specifies the interrupt priority. The lowest priority is 0 and the highest priority is 15. A priority level of 0 inhibits signalling of this interrupt to the core. Affects only interrupts routed to *int*.
16–31	VECTOR	Vector (Affects only interrupts routed to *int*). Contains the value returned when IACK is read and this interrupt resides in the corresponding interrupt request register (IRR) for that core, as shown in Figure 25-86.

25.3.7.2 External Interrupt Destination Registers (EIDR0–EIDR11)

The EIDRs, shown in Figure 25-74, control the destination of external interrupts caused by the assertion of any of IRQ[0:11]. If the EILR is programmed to assert *int* pins and EIVPR is programmed to edge sensitive, the EIDR*xn* bits can be set independently of each other and any or all can be set for this type of interrupt. Otherwise, if the interrupt is programmed to assert *int* pins and EIVPR is programmed to level sensitive, only one destination bit may be set; otherwise, behavior is undefined. If the EILR is programmed to assert *cint, or mcp* pins multiple destination pins may be set and system behavior must be managed by software.

Offset EIDR0: 0x5_0010; EIDR1: 0x5_0030; EIDR2: 0x5_0050; EIDR3: 0x5_0070; Access: Read/write
EIDR4: 0x5_0090; EIDR5: 0x5_00B0; EIDR6: 0x5_00D0; EIDR7: 0x5_00F0;
EIDR8: 0x5_0110; EIDR9: 0x5_0130; EIDR10: 0x5_0150; EIDR11: 0x5_0170

	0																				23	24	25	26	27	28	29	30	31
R	—																					P7	P6	P5	P4	P3	P2	P1	P0
W																													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 25-74. External Interrupt Destination Registers (EIDRs)

Table 25-79 describes the EIDR*n* fields.

Table 25-79. EIDR*n* Field Descriptions

Bits	Name	Description
0–17	—	Reserved
0–23	—	Reserved
24	P7	Processor core 7. This interrupt is multicasting for edge-triggered *int* or for *cint*, or *mcp* pins, so multiple bits can be set for any of these interrupts. 0 Processor core 7 does not receive this interrupt. 1 Directs the interrupt to processor core 7.
25	P6	Processor core 6. This interrupt is multicasting for edge-triggered *int* or for *cint*, or *mcp* pins, so multiple bits can be set for any of these interrupts. 0 Processor core 6 does not receive this interrupt. 1 Directs the interrupt to processor core 6.
26	P5	Processor core 5. This interrupt is multicasting for edge-triggered *int* or for *cint*, or *mcp* pins, so multiple bits can be set for any of these interrupts. 0 Processor core 5 does not receive this interrupt. 1 Directs the interrupt to processor core 5.
27	P4	Processor core 4. This interrupt is multicasting for edge-triggered *int* or for *cint*, or *mcp* pins, so multiple bits can be set for any of these interrupts. 0 Processor core 4 does not receive this interrupt. 1 Directs the interrupt to processor core 4.
28	P3	Processor core 3. This interrupt is multicasting for edge-triggered *int* or for *cint*, or *mcp* pins, so multiple bits can be set for any of these interrupts. 0 Processor core 3 does not receive this interrupt. 1 Directs the interrupt to processor core 3.
29	P2	Processor core 2. This interrupt is multicasting for edge-triggered *int* or for *cint*, or *mcp* pins, so multiple bits can be set for any of these interrupts. 0 Processor core 2 does not receive this interrupt. 1 Directs the interrupt to processor core 2.
30	P1	Processor core 1. This interrupt is multicasting for edge-triggered *int* or for *cint*, or *mcp* pins, so multiple bits can be set for any of these interrupts. 0 Processor core 1 does not receive this interrupt. 1 Directs the interrupt to processor core 1.
31	P0	Processor core 0. This interrupt is multicasting for edge-triggered *int* or for *cint*, or *mcp* pins, so multiple bits can be set for any of these interrupts. 0 Processor core 0 does not receive this interrupt. 1 Directs the interrupt to processor core 0. The default destination is for processor core 0 to receive this external interrupt after the MPIC is reset.

25.3.7.3 External Interrupt Level Registers (EILR0–EILR11)

The EILRs specify the actual signal to assert to the destination processor.

Offset EILR0: 0x5_0018; EILR1: 0x5_0038; EILR2: 0x5_0058; EILR3: 0x5_0078; EILR4: 0x5_0098; EILR5: 0x5_00B8; EILR6: 0x5_00D8; EILR7: 0x5_00F8; EILR8: 0x5_0118; EILR9: 0x5_0138; EILR10: 0x5_0158; EILR11: 0x5_0178 — Access: Read/Write

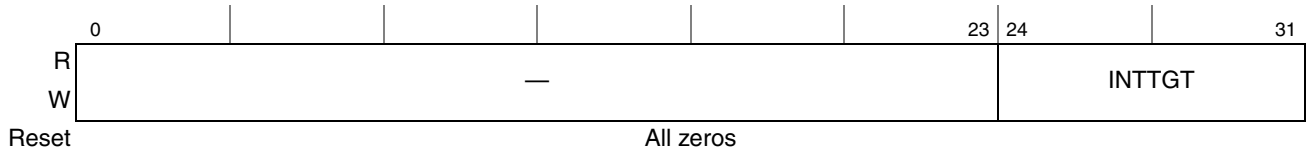


Figure 25-75. External Interrupt Level Registers (EILR*n*)

Table 25-80 describes the EILR*n* fields.

Table 25-80. EILR*n* Field Descriptions

Bits	Name	Description
0-23	—	Reserved
24–31	INTTGT	Interrupt Target. Specifies which signal is asserted 0x00 *int* 0x01 *cint* 0x02 *mcp* The target processor core for *int*, *cint*, and *mcp* is specified in the associated EIDR*n* 0xF0 *sie0* 0xF1 *sie1* 0xF2 *sie2* 0xFF $\overline{\text{IRQ\_OUT}}$ external output signal *sie*[0:2] and $\overline{\text{IRQ\_OUT}}$ override the processor specified in the associated EIDR. See the Interrupt Assignments chapter for *sie*[0:2] routing assignments. All other values are reserved.

25.3.7.4 Internal Interrupt Vector/Priority Registers (IIVPR0–IIVPR255)

The IIVPRs, shown in Figure 25-76, have the same fields and format as the GTVPRs, except that they apply to the internal interrupt sources listed in Table 25-4. These interrupts are all level-sensitive.

Offset IIVPR0–7 0x5_0200, 0x5_0220, 0x5_0240, 0x5_0260, 0x5_0280, 0x5_02A0, 0x5_02C0, 0x5_02E0
IIVPR8–15 0x5_0300, 0x5_0320, 0x5_0340, 0x5_0360, 0x5_0380, 0x5_03A0, 0x5_03C0, 0x5_03E0
.....IIVPR152–159 0x5_1500, 0x5_1520, 0x5_1540, 0x5_1560, 0x5_1580, 0x5_15A0, 0x5_15C0, 0x5_15E0
....
IIVPR248–255 0x5_3B00, 0x5_3B20, 0x5_3B40, 0x5_3B60, 0x5_3B80, 0x5_3BA0, 0x5_3BC0, 0x5_3BE0 — Access: Mixed

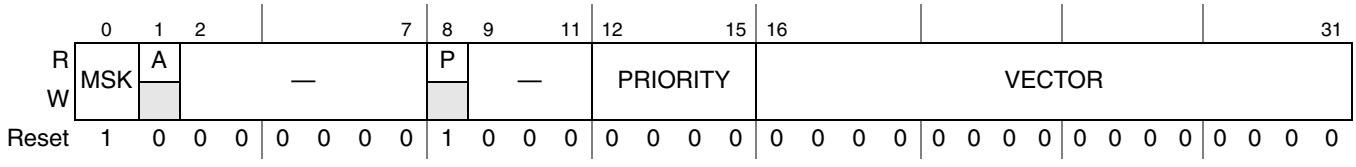


Figure 25-76. Internal Interrupt Vector/Priority Registers (IIVPRs)

Table 25-81 describes the IIVPR fields.

Table 25-81. IIVPR*n* Field Descriptions

Bits	Name	Description
0	MSK	Mask. Mask interrupts from this source. MSK affects interrupts routed to *int*, *cint*, *mcp*, *sie*[0:2], and IRQ_OUT. 0 An interrupt request is generated if the corresponding IPR bit is set. 1 Further interrupts from this source are disabled.
1	A	Activity. Indicates an interrupt has been requested or is in service. The VECTOR and PRIORITY values should not be changed while this bit is set. Affects only interrupts routed to *int*. 0 No current interrupt activity associated with this source. 1 The interrupt field for this source is set in the IPR or ISR.
2–7	—	Reserved
8	P	Polarity. All internal interrupts are active-high. **Note:** This bit is read-only and always set specifying the polarity as active high.
9–11	—	Reserved
12–15	PRIORITY	Priority. Specifies the interrupt priority. The lowest priority is 0 and the highest priority is 15. A priority level of 0 inhibits signalling of this interrupt to the core. Affects only interrupts routed to *int*.
16–31	VECTOR	Vector (Affects only interrupts routed to *int*). Contains the value returned when IACK is read and this interrupt resides in the corresponding interrupt request register (IRR) for that core, as shown in Figure 25-86.

25.3.7.5 Internal Interrupt Destination Registers (IIDR0–IIDR255)

The IIDRs, shown in Figure 25-77, have the same fields and format as EIVDRs, except that they apply to the internal interrupt sources listed in Table 25-4. If the IILR is programmed to assert *int* pins, only one destination bit may be set; otherwise, behavior is undefined. If the IILR is programmed to assert *cint* or *mcp* pins multiple destination pins may be set and system behavior must be managed by software.

Offset IIDR0–7 0x5_0210, 0x5_0230, 0x5_0250, 0x5_0270, 0x5_0290, 0x5_02B0, 0x5_02D0, 0x5_02F0 Access: Read/Write
IIDR8–15 0x5_0310, 0x5_0330, 0x5_0350, 0x5_0370, 0x5_0390, 0x5_03B0, 0x5_03D0, 0x5_03F0
.....IIDR152–159 0x5_1510, 0x5_1530, 0x5_1550, 0x5_1570, 0x5_1590, 0x5_15B0, 0x5_15D0, 0x5_15F0
...IIDR248–255 0x5_3B10, 0x5_3B30, 0x5_3B50, 0x5_3B70, 0x5_3B90, 0x5_3BB0, 0x5_3BD0, 0x5_3BF0

	0																						23	24	25	26	27	28	29	30	31
R W	—																							P7	P6	P5	P4	P3	P2	P1	P0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 25-77. Internal Interrupt Destination Registers (IIDRs)

Table 25-82 describes the IIDR fields.

Table 25-82. IIDR*n* Field Descriptions

Bits	Name	Description
0–23	—	Reserved
24	P7	Processor core 7. Indicates whether processor core 7 receives the interrupt. 0 Processor core 7 does not receive this interrupt. 1 Directs the interrupt to processor core 7.
25	P6	Processor core 6. Indicates whether processor core 6 receives the interrupt. 0 Processor core 6 does not receive this interrupt. 1 Directs the interrupt to processor core 6.
26	P5	Processor core 5. Indicates whether processor core 5 receives the interrupt. 0 Processor core 5 does not receive this interrupt. 1 Directs the interrupt to processor core 5.
27	P4	Processor core 4. Indicates whether processor core 4 receives the interrupt. 0 Processor core 4 does not receive this interrupt. 1 Directs the interrupt to processor core 4.
28	P3	Processor core 3. Indicates whether processor core 3 receives the interrupt. 0 Processor core 3 does not receive this interrupt. 1 Directs the interrupt to processor core 3.
29	P2	Processor core 2. Indicates whether processor core 2 receives the interrupt. 0 Processor core 2 does not receive this interrupt. 1 Directs the interrupt to processor core 2.
30	P1	Processor core 1. Indicates whether processor core 1 receives the interrupt. 0 Processor core 1 does not receive this interrupt. 1 Directs the interrupt to processor core 1.
31	P0	Processor core 0. Indicates whether processor core 0 receives the interrupt. 0 Processor core 0 does not receive this interrupt. 1 Directs the interrupt to processor core 0. The default destination is for processor core 0 to receive this interrupt after the MPIC is reset.

25.3.7.6 Internal Interrupt Level Registers (IILR0–IILR255)

The IILRs specify the actual signal to assert to the destination processor.

Offset IILR0–7 0x5_0218, 0x5_0238, 0x5_0258, 0x5_0278, 0x5_0298, 0x5_02B8, 0x5_02D8, 0x5_02F8 Access:
IILR8–15 0x5_0318, 0x5_0338, 0x5_0358, 0x5_0378, 0x5_0398, 0x5_03B8, 0x5_03D8, 0x5_03F8 Read/Write
.....
IILR152–159 0x5_1518, 0x5_1538, 0x5_1558, 0x5_1578, 0x5_1598, 0x5_15B8, 0x5_15D8, 0x5_15F8
...
IILR248–255 0x5_3B18, 0x5_3B38, 0x5_3B58, 0x5_3B78, 0x5_3B98, 0x5_3BB8, 0x5_3BD8, 0x5_3BF8

Reset All zeros

Figure 25-78. Internal Interrupt Level Registers (IILR*n*)

Table 25-83 describes the IILR*n* fields.

Table 25-83. IILR*n* Field Descriptions

Bits	Name	Description
0-23	—	Reserved
24–31	INTTGT	Interrupt Target. Specifies which signal is asserted 0x00 *int* 0x01 *cint* 0x02 *mcp* The target processor core for *int*, *cint*, and *msp* is specified in the associated IIDR*n* 0xF0 *sie0* 0xF1 *sie1* 0xF2 *sie2* 0xFF $\overline{\text{IRQ\_OUT}}$ external output signal *sie*[0:2] and $\overline{\text{IRQ\_OUT}}$ override the processor specified in the associated IIDR. See the Interrupt Assignments chapter for *sie*[0:2] routing assignments. All other values are reserved.

25.3.7.7 Messaging Interrupt Vector/Priority Registers (MIVPRA0–MIVPRA7, MIVPRB0–MIVPRB7)

The MIVPRs have the same fields and format as the GTVPRs, except they apply to messaging interrupts.

Offset MIVPRA0: 0x5_1600; MIVPRA1: 0x5_1620; MIVPRA2: 0x5_1640; MIVPRA3: 0x5_1660
MIVPRB0: 0x5_1680; MIVPRB1: 0x5_16A0; MIVPRB2: 0x5_16C0; MIVPRB3: 0x5_16E0
MIVPRA4: 0x5_1700; MIVPRA5: 0x5_1720; MIVPRA6: 0x5_1740; MIVPRA7: 0x5_1760
MIVPRB4: 0x5_1780; MIVPRB5: 0x5_17A0; MIVPRB6: 0x5_17C0; MIVPRB7: 0x5_17E0

Access: Read/Write

	0	1	2–11	12–15	16–31
R W	MSK	A	—	PRIORITY	VECTOR
Reset	1	0	0 0 0 0 0 0 0 0 0 0	0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Figure 25-79. Messaging Interrupt Vector/Priority Registers (MIVPR*n*)

Table 25-84 describes the MIVPR*n* fields.

Table 25-84. MIVPR*n* Field Descriptions

Bits	Name	Description
0	MSK	Mask. Mask interrupts from this source. MSK affects interrupts routed to *int*, *cint*, *mcp*, *sie*[0:2], and $\overline{\text{IRQ\_OUT}}$. 0 An interrupt request is generated if the corresponding IPR bit is set. 1 Further interrupts from this source are disabled.
1	A	Activity. Indicates an interrupt has been requested or is in service. The VECTOR and PRIORITY values should not be changed while this bit is set. Affects only interrupts routed to *int*. 0 No current interrupt activity associated with this source. 1 The interrupt field for this source is set in the IPR or ISR.
2–11	—	Reserved
12–15	PRIORITY	Priority. Specifies the interrupt priority. The lowest priority is 0 and the highest priority is 15. A priority level of 0 inhibits signalling of this interrupt to the core. Affects only interrupts routed to *int*.
16–31	VECTOR	Vector (Affects only interrupts routed to *int*). Contains value returned when IACK is read and this interrupt resides in the corresponding interrupt request register (IRR) for that core, as shown in Figure 25-86.

25.3.7.8 Messaging Interrupt Destination Registers (MIDRA0–MIDRA7, MIDRB0–MIDRB7)

The messaging interrupt destination registers (MIDRs), shown in Figure 25-80, control the destination for the messaging interrupts. Only one destination bit may be set; otherwise, behavior is undefined.

Offset MIDRA0: 0x5_1610; MIDRA1: 0x5_1630; MIDRA2: 0x5_1650; MIDRA3: 0x5_1670 Access: Read/Write
MIDRB0: 0x5_1690; MIDRB1: 0x5_16B0; MIDRB2: 0x5_16D0; MIDRB3: 0x5_16F0
MIDRA4: 0x5_1710; MIDRA5: 0x5_1730; MIDRA6: 0x5_1750; MIDRA7: 0x5_1770
MIDRB4: 0x5_1790; MIDRB5: 0x5_17B0; MIDRB6: 0x5_17D0; MIDRB7: 0x5_17F0

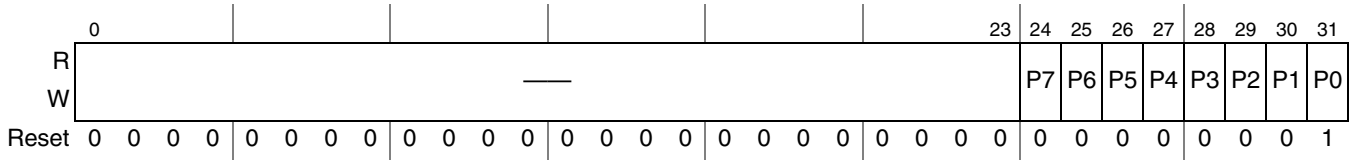


Figure 25-80. Messaging Interrupt Destination Registers (MIDR*n*)

Table 25-85 describes the MIDR*n* fields.

Table 25-85. MIDR*n* Field Descriptions

Bits	Name	Description
0–23	—	Reserved
24	P7	Processor core 7. Indicates whether processor core 7 receives the interrupt. 0 Processor core 7 does not receive this interrupt. 1 Directs the interrupt to processor core 7.
25	P6	Processor core 6. Indicates whether processor core 6 receives the interrupt. 0 Processor core 6 does not receive this interrupt. 1 Directs the interrupt to processor core 6.
26	P5	Processor core 5. Indicates whether processor core 5 receives the interrupt. 0 Processor core 5 does not receive this interrupt. 1 Directs the interrupt to processor core 5.
27	P4	Processor core 4. Indicates whether processor core 4 receives the interrupt. 0 Processor core 4 does not receive this interrupt. 1 Directs the interrupt to processor core 4.
28	P3	Processor core 3. Indicates whether processor core 3 receives the interrupt. 0 Processor core 3 does not receive this interrupt. 1 Directs the interrupt to processor core 3.
29	P2	Processor core 2. Indicates whether processor core 2 receives the interrupt. 0 Processor core 1 does not receive this interrupt. 1 Directs the interrupt to processor core 2.
30	P1	Processor core 1. Indicates whether processor core 1 receives the interrupt. 0 Processor core 1 does not receive this interrupt. 1 Directs the interrupt to processor core 1.
31	P0	Processor core 0. Indicates whether processor core 0 receives the interrupt. 0 Processor core 0 does not receive this interrupt. 1 Directs the interrupt to processor core 0. The default destination is for processor core 0 to receive this interrupt after the MPIC is reset.

25.3.8 Per-CPU (Private Access) Registers

The OpenPIC programming model supports multiprocessor systems of up to 32 separate processors. As such, the OpenPIC interface specification provides for coordinating both the requesting and servicing of interrupts among several processor cores within a single integrated device. To comply with the OpenPIC specification, the MPIC incorporates several of these multiprocessor capabilities.

NOTE

Note that these registers are meaningful only for interrupts routed to *int*.

The registers in Table 25-86 are called per-CPU registers because they are duplicated for each core in a multicore device. The OpenPIC interface specifies that a copy of these registers be available to each core at the same physical address by using the ID of the processor core that initiates the transaction to determine the set of per-CPU registers to access.

Table 25-86. Per-CPU Registers—Private Access Address Offsets

Register Name	Offset
Interprocessor 0 dispatch register (IPIDR0)	0x4_0040
Interprocessor 1 dispatch register (IPIDR1)	0x4_0050
Interprocessor 2 dispatch register (IPIDR2)	0x4_0060
Interprocessor 3 dispatch register (IPIDR3)	0x4_0070
Current task priority register (CTPR)	0x4_0080
Who am I register (WHOAMI)	0x4_0090
Interrupt acknowledge register (IACK)	0x4_00A0
End of interrupt register (EOI)	0x4_00B0

These addresses, shown in Table 25-86, appear in the memory map at the same offset for every processor in what is called the private access space. This duplication allows user code to execute correctly in an multiprocessor environment without needing to know which core it is running on. On a single-core device, each register has two addresses, one in the normal address space and one in the private access space. It is included on even single-core devices to simplify the porting of such code.

25.3.8.1 Interprocessor Interrupt Dispatch Register (IPIDR0–IPIDR3)

Figure 25-81 shows the four IPIDRs, one for each interprocessor interrupt channel. Writing to an IPIDR with a bit set causes a self interrupt for a single-core device. Because external bus masters can access the

these registers by using the per-CPU global access offsets, this feature can serve as a doorbell-type interrupt.

Offset Per-CPU private offsets: IPIDR0: 0x4_0040; IPIDR1: 0x4_0050; IPIDR2: 0x4_0060; IPIDR3: 0x4_0070 Access: Write only
Processor core 0: IPIDR0: 0x6_0040; IPIDR1: 0x6_0050; IPIDR2: 0x6_0060; IPIDR3: 0x6_0070
Processor core 1: IPIDR0: 0x6_1040; IPIDR1: 0x6_1050; IPIDR2: 0x6_1060; IPIDR3: 0x6_1070
Processor core 2: IPIDR0: 0x6_2040; IPIDR1: 0x6_2050; IPIDR2: 0x6_2060; IPIDR3: 0x6_2070
Processor core 3: IPIDR0: 0x6_3040; IPIDR1: 0x6_3050; IPIDR2: 0x6_3060; IPIDR3: 0x6_3070
Processor core 4: IPIDR0: 0x6_4040; IPIDR1: 0x6_4050; IPIDR2: 0x6_4060; IPIDR3: 0x6_4070
Processor core 5: IPIDR0: 0x6_5040; IPIDR1: 0x6_5050; IPIDR2: 0x6_5060; IPIDR3: 0x6_5070
Processor core 6: IPIDR0: 0x6_6040; IPIDR1: 0x6_6050; IPIDR2: 0x6_6060; IPIDR3: 0x6_6070
Processor core 7: IPIDR0: 0x6_7040; IPIDR1: 0x6_7050; IPIDR2: 0x6_7060; IPIDR3: 0x6_7070

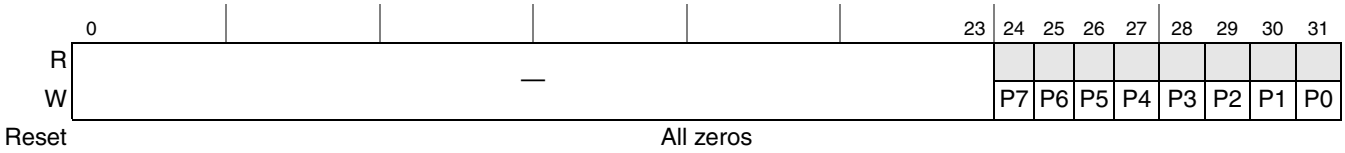


Figure 25-81. Interprocessor Interrupt Dispatch Registers (IPIDR0–IPIDR3)

Table 25-87 describes the IPIDR*n* fields.

Table 25-87. IPIDR*n* Field Descriptions

Bits	Name	Description
0–23	—	Reserved
24	P7	Processor core 7. Specifies if processor core 7 receives the interrupt. This interrupt is multicasting, so multiple bits can be set. 0 Processor core 7 does not receive the interrupt 1 Directs the interrupt to processor core 7
25	P6	Processor core 6. Specifies if processor core 6 receives the interrupt. This interrupt is multicasting, so multiple bits can be set. 0 Processor core 6 does not receive the interrupt 1 Directs the interrupt to processor core 6
26	P5	Processor core 5. Specifies if processor core 1 receives the interrupt. This interrupt is multicasting, so multiple bits can be set. 0 Processor core 5 does not receive the interrupt 1 Directs the interrupt to processor core 5
27	P4	Processor core 4. Specifies if processor core 4 receives the interrupt. This interrupt is multicasting, so multiple bits can be set. 0 Processor core 4 does not receive the interrupt 1 Directs the interrupt to processor core 4
28	P3	Processor core 3. Specifies if processor core 3 receives the interrupt. This interrupt is multicasting, so multiple bits can be set. 0 Processor core 3 does not receive the interrupt 1 Directs the interrupt to processor core 3
29	P2	Processor core 2. Specifies if processor core 2 receives the interrupt. This interrupt is multicasting, so multiple bits can be set. 0 Processor core 2 does not receive the interrupt 1 Directs the interrupt to processor core 2

Table 25-87. IPIDR*n* Field Descriptions (continued)

Bits	Name	Description
30	P1	Processor core 1. Specifies if processor core 1 receives the interrupt. This interrupt is multicasting, so multiple bits can be set. 0 Processor core 1 does not receive the interrupt 1 Directs the interrupt to processor core 1
31	P0	Processor core 0. Determines if processor core 0 receives the interrupt. 0 Processor core 0 does not receive the interrupt. 1 Directs the interrupt to processor core 0.

25.3.8.2 Processor Core Current Task Priority Registers 0–7 (CTPR0–CTPR7)

There is one CTPR per processor core on this device as shown in Figure 25-82.

Offset Per-CPU private offset: 0x4_0080
Processor core 0: CTPR0: 0x6_0080
Processor core 1: CTPR1: 0x6_1080
Processor core 2: CTPR2: 0x6_2080
Processor core 3: CTPR3: 0x6_3080
Processor core 4: CTPR4: 0x6_4080
Processor core 5: CTPR5: 0x6_5080
Processor core 6: CTPR6: 0x6_6080
Processor core 7: CTPR7: 0x6_7080.....
Access: Read/Write

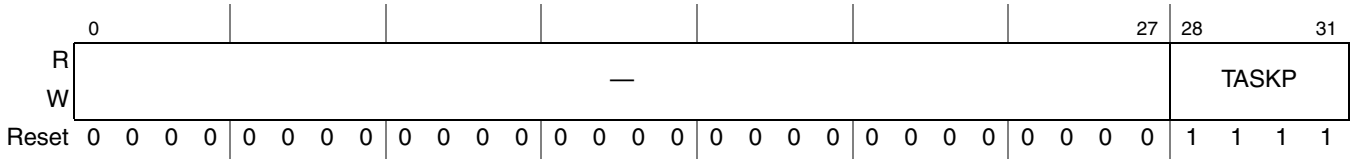


Figure 25-82. Processor Core Current Task Priority Registers (CTPR*n*)

NOTE

CTPR has meaning only for interrupts routed to *int*.

Software must write the priority of the current processor core task in the CTPR for each core. The MPIC uses this value for comparison with the priority of incoming interrupts. Given several concurrent incoming interrupts, the highest priority interrupt is asserted to that core if the following apply:

- The interrupt is not masked.
- The priority of the interrupt is higher than the values in the corresponding CTPR[TASKP] and ISR.

Table 25-88 describes the CTPR task priority field.

Table 25-88. CTPR*n* Field Descriptions

Bits	Name	Description
0–27	—	Reserved
28–31	TASKP	Task priority. Indicates the threshold that individual interrupt priorities must exceed for the interrupt request to be serviced. Task priority is meaningless (that is, a don't care) if the processor ID in WHOAMI[ID] is illegal. 0000–1111 *x*VPR*n*[PRIORITY] must exceed this value for the interrupt request to be serviced. Note the following special cases: 0000 Lowest priority. All interrupts except those whose priority are 0 can be serviced. 1111 Highest priority. No interrupts are signaled to that processor core. Hardware selects this value on a device hard reset or when the corresponding PIR[P*n*] is set.

25.3.8.3 Who Am I Registers 0–7 (WHOAMI0–WHOAMI7)

The processor core WHOAMI*n* register, shown in Figure 25-83, can be read by a processor core to determine its physical connection to the MPIC. The value returned when reading this register may be used to determine the value for the destination masks used for dispatching interrupts.

Offset Per-CPU private offset: 0x4_0090;
Processor core 0: WHOAMI0: 0x6_0090
Processor core 1: WHOAMI1: 0x6_1090
Processor core 2: WHOAMI2: 0x6_2090
Processor core 3: WHOAMI3: 0x6_3090
Processor core 4: WHOAMI4: 0x6_4090
Processor core 5: WHOAMI5: 0x6_5090
Processor core 6: WHOAMI6: 0x6_6090
Processor core 7: WHOAMI7: 0x6_7090
Access: Read only

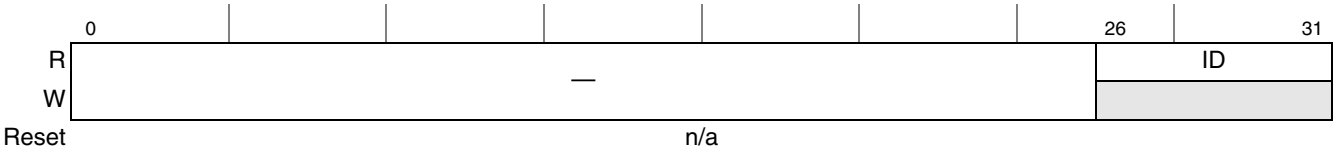


Figure 25-83. Processor Core Who Am I Registers (WHOAMI*n*)

Table 25-89 describes the WHOAMI*n* fields.

Table 25-89. WHOAMI*n* Field Descriptions

Bits	Name	Description
0–25	—	Reserved
26–31	ID	Returns the ID of the processor core reading this register. 000000 Processor core 0 000001 Processor core 1 000010 Processor core 2 000011 Processor core 3 000100 Processor core 4 000101 Processor core 5 000110 Processor core 6 000111 Processor core 7 **Note:** ID = 111111 indicates an illegal processor ID)

25.3.8.4 Processor Core Interrupt Acknowledge Registers 0–7 (IACK0–IACK7)

NOTE

IACK has meaning only for interrupts routed to *int* in mixed mode (GCR[M] = 01)and should not be accessed for interrupts routed to *cint, mcp, sie* or $\overline{\text{IRQ\_OUT}}$.

The external proxy facility mode (GCR[M] = 11), eliminates the need for the core to read the IACK register as the vector is automatically passed to the core's external proxy register (EPR) or guest external proxy register (GEPR).

In systems based on processors that implement the Power architecture, the interrupt acknowledge function occurs as an explicit read operation to a memory-mapped interrupt acknowledge register (IACK), shown in Figure 25-84. Each processor core has an IACK register assigned to it. Reading IACK returns the

interrupt vector corresponding to the highest priority pending interrupt. Reading IACK also has the following side effects:

- The associated field in the corresponding interrupt pending register (IPR) is cleared for edge-sensitive interrupts.
- The corresponding in-service register (ISR) is updated.
- The corresponding *int* output signal from the MPIC is negated.

Reading IACK when no interrupt is pending returns the spurious vector value, as described in Section 25.3.1.10, "Spurious Vector Register (SVR)."

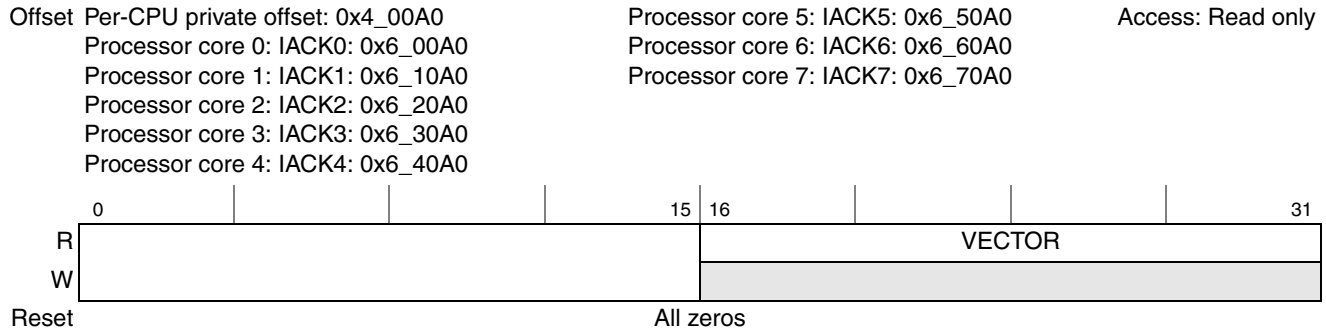


Figure 25-84. Processor Core Interrupt Acknowledge Registers (IACK*n*)

Table 25-90 describes the IACK*n* fields.

Table 25-90. IACK*n* Field Descriptions

Bits	Name	Description
0–15	—	Reserved
16–31	VECTOR	Interrupt vector. Vector of the highest pending interrupt (read only). This field is meaningless (that is, a don't care) if the processor ID in WHOAMI[ID] is illegal.

25.3.8.5 Processor Core End of Interrupt Registers 0–7 (EOI0–EOI7)

NOTE

EOI has meaning only for interrupts routed to *int* and should not be accessed for interrupts routed to *cint, mcp, sie* or $\overline{\text{IRQ\_OUT}}$.

Each core is assigned an EOI register, shown in Figure 25-85. Writing to EOI signals the end of processing for the highest-priority interrupt (routed to *int*) currently in service. It also updates the corresponding ISR*n* by retiring the highest priority interrupt. Data values written to EOI are ignored, and zero is assumed.

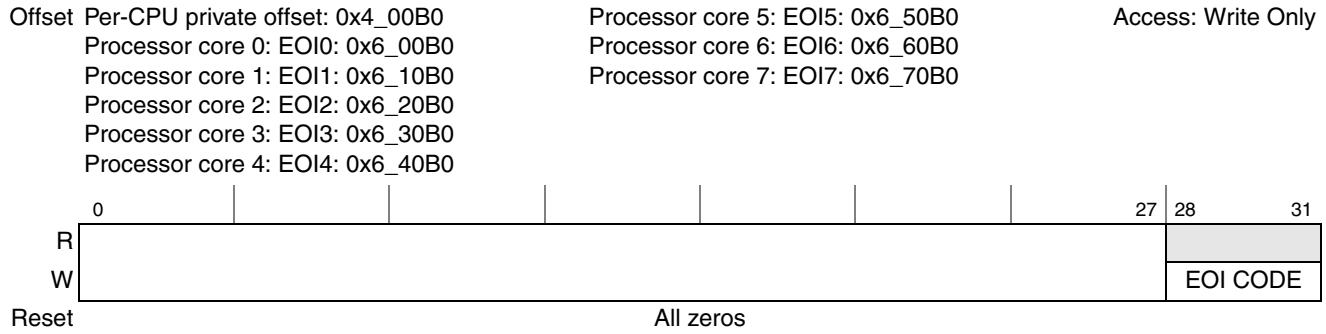


Figure 25-85. End of Interrupt Registers (EOI*n*)

Table 25-91 describes the EOI*n* fields.

Table 25-91. EOI*n* Field Descriptions

Bits	Name	Description
0–27	—	Reserved
28–31	EOI CODE	0000 (write only)

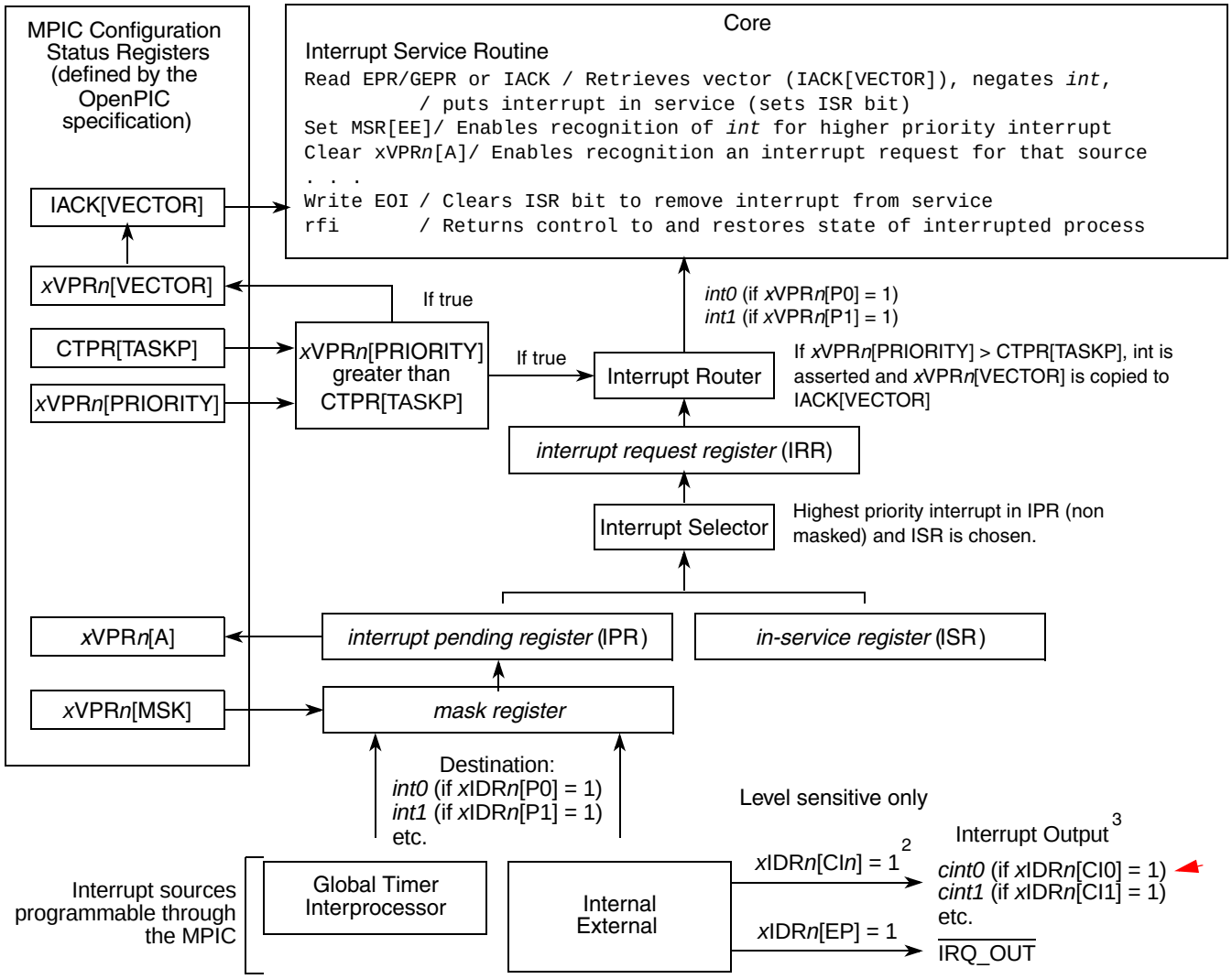
25.4 Functional Description

This section is a functional description of the MPIC.

25.4.1 Flow of Interrupt Control

Figure 25-86 shows the flow of interrupts directed by the MPIC to the *int*, *cint*, *sie* or and $\overline{\text{IRQ\_OUT}}$ outputs. Note that this diagram describes a conceptual model of an MPIC on a single processor. This logic is replicated for each implemented processor. This conceptual diagram does not fully represent all internal circuitry of the implementation

This figure focusses especially on the OpenPIC-defined logic and shows how the MPIC controls interrupt requests that target the *int* signal. The flow in Figure 25-86 is from the bottom to the top, and shows at the bottom how the destination register associated with each source determines the path.



1 If *cint*, *sie* or $\overline{\text{IRQ\_OUT}}$ is the destination, EIVPR*n*[S] must be set to configure the source as level sensitive.

2 If multiple destination register bits are set, MPIC behavior is undefined.

3 Although setting CI*n* directs the interrupt request to the critical interrupt output (*cintn*), integrated logic may connect this signal to a different interrupt input to the core.

Figure 25-86. MPIC Interrupt Processing Flow Diagram for Each Core (*n*)

25.4.1.1 Interrupts Routed to *cint, mcp, sie,* or $\overline{\text{IRQ\_OUT}}$

Interrupt requests routed to *cint*, *mcp*, *sie*, or $\overline{\text{IRQ\_OUT}}$ bypass the logic that is dedicated to interrupt sources that compete for *int*.

NOTES

Because interrupt sources routed to *cint*, *sie*, $\overline{\text{IRQ\_OUT}}$ and internal interrupt sources routed to *mcp* must be level sensitive, EIVPR[S] has no effect. External interrupt sources routed to *mcp* can be level-sensitive or edge-triggered since the core always uses edge detection logic. Consequently, EIVPR[S] has no effect in this case as well. See Section 25.3.7.1, "External Interrupt Vector/Priority Registers (EIVPR0–EIVPR11)."

Because these interrupts bypass the OpenPIC logic, it is especially important that handlers do not read IACK. Doing so causes a spurious interrupt. Likewise, they should not write EOI.

25.4.1.2 Interrupts Routed to *int*

As shown in Figure 25-86, the MPIC receives interrupt requests from external and internal sources and from within the MPIC itself. As Figure 25-86 shows, all of these interrupt sources can be routed to *int*; the global timer and timer processor interrupts can be directed only to *int*.

The sources' mask bits (*x*VPR*n*[MSK]) are tracked in the internal mask register. If a source's MSK bit is set, the mask register prevents the MPIC from asserting *int* (or *cint*, *mcp*, *sie*[0:2], $\overline{\text{IRQ\_OUT}}$) on its behalf.

NOTE

An edge-sensitive interrupt event is lost if the corresponding interrupt's mask bit is set or if the interrupt event is level-sensitive and it deasserts before the interrupt is unmasked.

Unmasked interrupt requests are captured in the interrupt pending register (IPR), an internal interrupt summary register with a bit for each source. If an interrupt request is multicast, a bit is set in the IPR for each targeted processor. Although the IPR cannot be read by software, when an IPR bit is set, the corresponding source's activity bit (*x*VPR*n*[A]) is automatically set.

The interrupt selector monitors the IPR and the in-service register (ISR), which tracks previously taken interrupts that were superseded by a higher-priority interrupt before the interrupt handler finished. The interrupt selector recognizes the highest priority unmasked interrupt request and latches it into the interrupt request register (IRR). The source's vector (*x*VPR*n*[VECTOR]) is automatically passed to the core's external proxy register (EPR) or guest external proxy register (GEPR) in external proxy facility mode (GCR[M] = 11) or it is copied to IACK[VECTOR] in mixed mode (GCR[M] = 01).

If the priority (*x*VPR*n*[PRIORITY]) of an interrupt latched in the IRR is higher than the value in the target processor's CTPR[TASKP], the interrupt router asserts the external interrupt signal (*int*), causing that processor core to vector to its external interrupt handler.

In mixed mode (CGR[M] = 01), the interrupt handler must acknowledge the interrupt by explicitly reading the corresponding IACK register, described in Section 25.3.8.4, "Processor Core Interrupt Acknowledge Registers 0–7 (IACK0–IACK7)." The MPIC interprets this read as an interrupt acknowledge (IACK) cycle. In external proxy facilty mode (GCR[M] = 11, the core automatically generates the IACK cycle

(when MSR[EE] = 1) to the MPIC, although from system software perspective, the core does not acknowledge the interrupt until the external interrupt is taken.

NOTE

In external proxy facility mode, interrupt handlers should retrieve the vector from the core's EPR (or GEPR) and not read the IACK register. Reading the IACK registers in external proxy facility mode causes a spurious interrupt.

The IACK cycle not only returns the source's vector, it also negates the *int* signal to the processor (making it possible for a higher priority interrupt to assert *int*) and sets the source's bit in the ISR, indicating that this interrupt has been put in service. An interrupt remains in service from the time until the corresponding end-of-interrupt register (EOI) is written, generating what the MPIC considers an EOI signal.

Figure 25-86 shows required elements in the interrupt handler

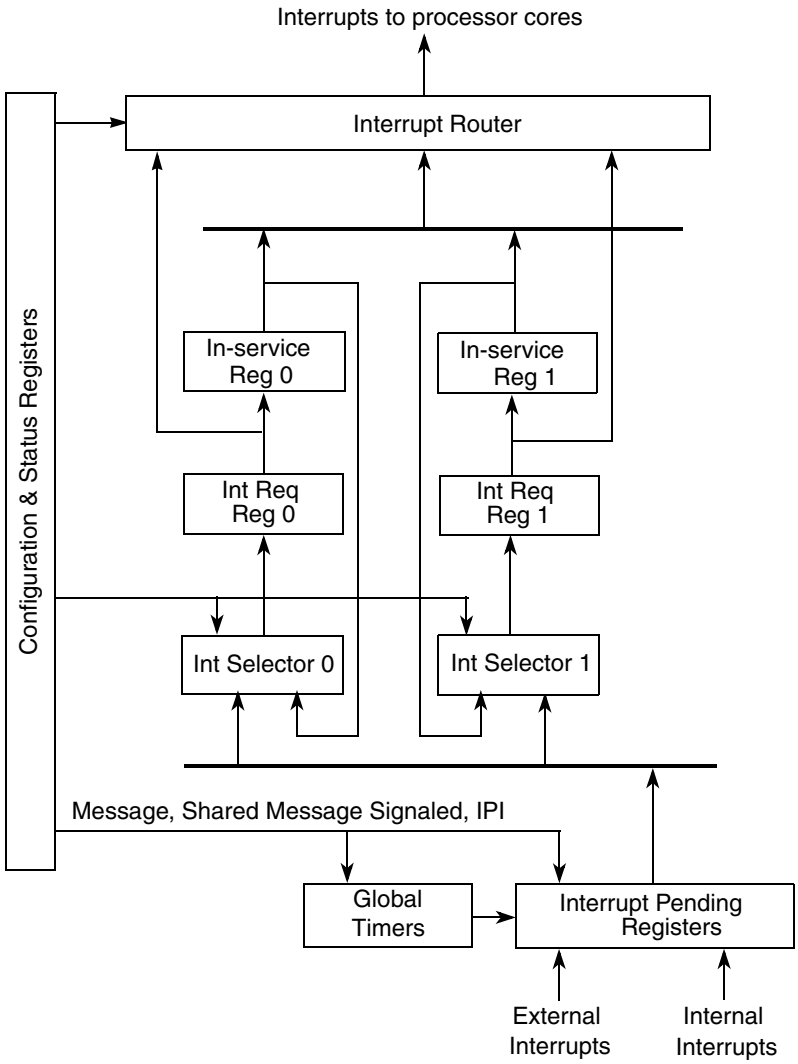


Figure 25-87. MPIC Block Diagram

25.4.1.2.1 Nesting of Interrupts

While an interrupt is being handled, if an interrupt request arrives with a higher *x*VPR*n*[PRIORITY] value, the interrupt being serviced is superseded. As described in Section 25.4.1.2, "Interrupts Routed to *int*," the MPIC asserts *int*, and the newer, higher priority interrupt is handled. This happens even if software, as part of its interrupt service routine, updates the corresponding CTPR with a lower value.

Thus, although several interrupts can be in service simultaneously (and tracked by the ISR), the highest priority interrupt by that processor is always the one actively handled. When the interrupt routine completes, it performs a write EOI cycle, a side effect of which is to take the current highest priority interrupt out of service (removes it from the ISR). At this point, the interrupt selector chooses the new highest priority interrupt request, and, assuming CTPR[TASKP] has not been updated to a value higher than the new interrupt, the MPIC asserts *int* on its behalf.

The next write EOI cycle takes the current highest priority interrupt out of service. An interrupt with lower priority than those in service is not started until all higher priority interrupts complete even if its priority is greater than the CTPR value.

25.4.1.2.2 Interrupt Source Priority

Each interrupt source routed to *int* is assigned a value through its *x*VPR*n*[PRIORITY] field. Priority values range from 0 to 15, where 15 is the highest. Interrupts are delivered only when the priority of the source is greater than the destination processor's CTPR[TASKP]. Therefore, setting *x*VPR*n*[PRIORITY] to zero inhibits that interrupt. Likewise, setting TASKP to 15 prevents the MPIC from delivering interrupts to that core through the *int* signal. Note that this is the reset value, preventing the MPIC from asserting *int* before the MPIC is configured.

The MPIC services simultaneous interrupts occurring with the same priority according to the following order:

1. MSGA0–MSGA7
2. MSGB0—MSGB7
3. MSIA0–MSIA15
4. MSIB0–MSIB15
5. MSIC0–MSIC15
6. MSID0-MSID15
7. IPI0–IPI3
8. Group A timer0–timer3
9. Group B timer0–timer3
10. IRQ[0:11]
11. Internal0–internal255

For example, if MSGA0, MSGA2, and IPI0 are all assigned the same priority and receive simultaneous interrupts, they are serviced in the following order:

1. MSGA0
2. MSGA2

3. IPI0

25.4.1.2.3 Interrupt Acknowledge

In mixed mode, when the MPIC causes *int* to be asserted, the external interrupt service routine acknowledges the request by reading that core's IACK register, which at this point holds the 16-bit vector value for the interrupt source that generated the request. This is the value programmed in that source's *x*VPR*n*[VECTOR]. Reading IACK has the following effects:

- The *int* signal for that core is negated, making it possible for another interrupt source to signal an external interrupt to the core, and more particularly, allowing the MPIC to signal a higher-priority interrupt, as described in Section 25.4.1.2.1, "Nesting of Interrupts."
- The source that caused that resource is represented in the internal in-service register (ISR).

The interrupt is then considered to be in service. It remains so until the processor core performs a write to the corresponding EOI. Writing to EOI is referred to as an EOI cycle.

In external proxy facility mode, the interrupt is automatically acknowledged with the same effects as reading the IACK register. Detailed description of the external proxy facility is described in the *e6500 Core Reference Manual*.

25.4.1.2.4 Spurious Vector Generation

Under certain circumstances, the MPIC has no valid vector to return to a processor core during an interrupt acknowledge cycle. In these cases, the spurious vector from the spurious vector register is returned. The following cases cause a spurious vector fetch:

- *int* (ipi_int_proc) is asserted in response to an externally or internally-sourced interrupt which is activated with level-sensitive logic, and the asserted level is negated before the interrupt is acknowledged.
- *int* (ipi_int_proc) is asserted for an interrupt source that is later masked (using the mask bit in the vector/priority register corresponding to that source) before the interrupt is acknowledged.
- *int* (ipi_int_proc) is asserted for an interrupt source that is later masked by an increase in the task priority level before the interrupt is acknowledged.
- An interrupt acknowledge cycle is performed by the processor core in spite of the fact that the MPIC has not asserted int (ipi_int_proc).

In all cases, a spurious vector is returned only if no pending interrupt has sufficient priority to signal an interrupt, otherwise, the vector for that interrupt source is returned.

NOTE

EOI should not be written in response to a spurious vector. Otherwise, a previously accepted interrupt might be cleared unintentionally.

25.4.2 Interprocessor Interrupts

Processors can generate interprocessor interrupts that target any processor. A self interrupt occurs when a core dispatches an interprocessor interrupt event to itself. Interrupts are initiated by writing one or several

of the P*n* bits in an interprocessor interrupt dispatch register (IPIDR0–IPIDR3) of one of the four IPI channels. Note that when initiating an interrupt event to more that one processor (by setting more than 1 P*n* bit in the IPIDR), each recipient processor must acknowledge, take delivery and EOI that interrupt. That is, the interrupt is delivered to each processor as a unique copy. Also note that if subsequent interprocessor interrupts from a given channel to a given target processor are initiated before the first is acknowledged, only one interrupt is generated.

25.4.3 Messaging Interrupts

25.4.3.1 Message Interrupts

The eight MSGRs, described in Section 25.3.5.1, "Message Registers (MSGRA0–MSGRA7, MSGRB0–MSGRB7)," can be used to send 32-bit messages to one or more processors. A messaging interrupt is generated by writing an MSGR if the corresponding MER bit is set and the interrupt is not masked. Reading a MSGR or writing a 1 to the status bit clears the interrupt.

25.4.3.2 Shared Message Signaled Interrupts

There are four sets of sixteen MSIRs (one set for each bank A, B, C, and D), described in Section 25.3.6.1, "Shared Message Signaled Interrupt Registers (MSIRa0–MSIRa15)," that indicate which of the interrupt sources sharing the MSI register have pending interrupts. Up to 32 sources can share any individual MSI register. A shared message signaled interrupt is generated by writing to Shared Message Signaled Interrupt Index Register (MSIIR or MSIIR1) fields SRS and IBS. The MSIIR supports up to eight registers (based on SRS field) and MSIIR1 supports up to sixteen. These registers are primarily intended to support PCI Express MSIs and each of the PCI Express ports can only use one MSI target at a time, either MSIIR or MSIIR1. The two fields, SRS and IBS, select one of the eight or sixteen registers in which an interrupt bit is to be set and the bit in the selected register, respectively. The result of the first eight MSIRs are OR of MSIIR and MSIIR1. This OR function is also valid for the Shared Message Signaled Status Register (MSISR). The corresponding interrupt needs to be unmasked for the interrupt to happen. A read to MSI register will clear the interrupt.

MSI also supports a coalescing feature. Each MSIR register has a MSI Coalescing register (MSICRa0-MSICRa15). The coalescing feature is enabled when MSICR has a non-zero value. When this feature is enabled, a MSI interrupt enters pending state only if all the events matching the coalescing register are set in MSIR. If an event occurs which does not have a matching coalescing bit set the resulting interrupt is 'coalesced' as if the bit is set.

25.4.4 PCI Express INTx

An INTx interrupt is signaled from the PCIe controller when it receives an in-band INTx message from a PCIe end point. These interrupts replace the out-of-band interrupts that historically were connected to the IRQ pins. Because the INTx interrupts and IRQ interrupts are generally mutually exclusive, the INTx signals are logically combined with the IRQ signals so that they share the same OpenPIC interrupt. See the Interrupt Assignments chapter for the association of INTx signals to IRQ signals.

25.4.5 Global Timers

There are appropriate clock prescalers and synchronizers to provide a time base for the internal MPIC timers. These 16 timers are organized as 2 groups of 8 timers each. The timers can be individually programmed to generate a processor core interrupt when they count down to zero and can be used to generate regular periodic interrupts. Each timer has the following four configuration and control registers:

- Global timer current count register (GTCCR*xn*)
- Global timer base count register (GTBCR*xn*)
- Global timer vector-priority register (GTVPR*xn*)
- Global timer destination register (GTDR*xn*)

The timer frequency should be written to the TFRR*xn*, described in Section 25.3.2.1, "Timer Frequency Reporting Register (TFRRA–TFRRB)."

Timer interrupts are all edge-triggered interrupts. If a timer period expires while a previous interrupt from the same source is pending or in service, the subsequent interrupt is lost.

The timer control register (TCR) provides users with the ability to create timers larger than the 31-bit global timers. The timer frequency can also be changed by setting the appropriate TCR fields, as described in Section 25.3.2.6, "Timer Control Registers (TCRA–TCRB)."

Note that when global timer interrupts are set to multicast (interrupt more that one processor by setting more than 1 P*n* bit in the GTDR*xn*), each recipient processor must acknowledge, take delivery and EOI that interrupt. That is, the interrupt is delivered to each processor as a unique copy.

25.4.6 Resets

This section describes the behavior of the MPIC at reset and the MPIC's ability to initiate processor resets.

25.4.7 Resetting the MPIC

The MPIC is reset by a device power-on reset (POR) or by software that sets GCR[RST], either of which causes the following:

- All pending and in-service interrupts are cleared.
- All interrupt mask bits are set.
- Polarity, sense, external signal, critical interrupt, and activity fields are reset to default values.
- PIR, TFRR, TCR, MER*x*, MSR*x*, and MSGR*x*0–MSGR*x*7 are cleared.
- MSG and timer destination fields are set.
- The interprocessor dispatch registers are cleared.
- All timer base count values are reset to zero with count inhibited.
- CTPR[TASKP] is reset to 0x000F, disabling delivery of interrupts that target *int*.
- The spurious interrupt vector resets to 0xFFFF.
- The PMMRs are reset to 0xFFFF.
- The MPIC defaults to the pass-through mode (GCR[M] = 00).

- All other registers remain at their pre-reset programmed values.

GCR[RST] is automatically cleared when the reset sequence is complete.

25.4.7.1 Processor Core Initialization

A software reset can be routed to any of the cores by writing to the processor core initialization register (PIR). This causes the assertion of the *hresetn* output signals from the MPIC. When this occurs, the corresponding CTPR also gets written to 0x000F to prevent delivery of any interrupts to *intn*.

25.4.8 Processor Core NMI

The processor core non-maskable interrupt (PNMIR) provides a way for software on another device to signal a non-maskable interrupt event to the processor cores by writing to PNMIR register. A *coren_nmi* signal to the cores is asserted when a one is written to the corresponding PNMIR field; these signals are held active until a zero is written to those same bits. This asserts the *core_nmi* input of the corresponding core.

25.5 Initialization/Application Information

This section contains initialization and application information for the MPIC.

25.5.1 Programming Guidelines

The following subsections contain information about programming MPIC registers.

25.5.1.1 MPIC Registers

Most MPIC control and status registers are readable and return the last value written. The exceptions to this rule are as follows:

- Interprocessor dispatch and EOI registers, which return zeros on reads.
- Activity bits (A) of the vector/priority registers reflect the status of the corresponding interrupt source.
- IACK, which returns the vector of the highest priority pending interrupt or the spurious vector (SVR[VECTOR]) if none is pending.
- Reserved fields always return 0.

When the MPIC is in mixed mode (GCR[M] = 01), the following guidelines are recommended:

- All MPIC registers must be located in a cache-inhibited, guarded area (configured through the core's MMU).
- The MPIC portion of the address map must be set up appropriately.

In addition, the following initialization sequence is recommended:

1. Write the vector, priority, and polarity values in each interrupt's vector/priority register, leaving their MSK (mask) bit set. This is required only if interrupts are used.

2. Clear CTPR (CTPR = All zeros).
3. Program the MPIC to mixed mode by setting GCR[M] = 01.
4. Clear the MSK bit in the vector/priority registers to be used.
5. Perform a software loop to clear all pending interrupts:
   — Load counter with FRR[NIRQ].
   — While counter > 0, read IACK and write EOI to guarantee all the IPR and ISR bits are cleared.
6. Set the processor core CTPR values to the desired values.

Depending on the interrupt system configuration, the MPIC may generate spurious interrupts to clear interrupts latched during power-up. A spurious or non-spurious vector is returned for an interrupt acknowledge cycle in this case.

25.5.1.2 Changing Interrupt Source Configuration

To change the vector, priority, polarity, sense, or destination of an active (unmasked) interrupt source, the following steps should be taken:

1. Mask the source using the mask (MSK) bit in the vector/priority register.
2. Wait for the activity (A) bit for that source to be cleared.
3. Make the desired changes.
4. Unmask the source.

Note that changing the destination from *int* to *cint, mcp, sie* or $\overline{\text{IRQ\_OUT}}$ makes the A, and PRIORITY fields meaningless; however, the MSK field does affect *cint*, *mcp*, *sie*[0:2], and $\overline{\text{IRQ\_OUT}}$.

25.5.1.3 Requirement

1. After issuing a reset to the core, there needs to be 15 EOIs before any new interrupt is sent to the core to make sure no interrupt is stuck in an in-service state. Note that each core has its own EOI register and this procedure must be performed for each core that has been reset.

Chapter 26
Interrupt Assignments

26.1 Introduction

This chapter describes the programmable interrupt controller (PIC) internal interrupt assignments for the T2080. These are the on-chip interrupt sources from peripheral logic within the integrated device.

26.2 Internal Interrupt Sources

Table 26-1. T2080 Internal Interrupt Assignments

Internal Interrupt Number	Interrupt Source
0	ORed error interrupt-See table below for individual assignments
1	WRS
2	Thermal monitor unit Alarm
3	Thermal monitor unit Critical Alarm
4	PCI Express 1
5	PCI Express 2
6	PCI Express 3
7	PCI Express 4
8	PAMU (access violations) **NOTE:** Interrupts from all PAMUs are ORed together and connected to this Interrupt
9	Integrated Flash controller (IFC) **NOTE:** All IFC interrupts are ORed together and connected to this Interrupt
...	-
12	DMA 1 channel 1
13	DMA 1 channel 2
14	DMA 1 channel 3
15	DMA 1 channel 4
16	DMA 2 channel 1

Table continues on the next page...

Table 26-1. T2080 Internal Interrupt Assignments (continued)

Internal Interrupt Number	Interrupt Source
17	DMA 2 channel 2
18	DMA 2 channel 3
19	DMA 2 channel 4
20	DUART 1
21	DUART 2
22	Dual I2C 1 (I2C1 and I2C2)
23	Dual I2C 2 (I2C3 and I2C4)
24	PCI Express 1 INTA
25	PCI Express 2 INTA
26	PCI Express 3 INTA
27	PCI Express 4 INTA
28	USB 1
29	USB 2
...	-
32	eSDHC
...	-
36	Performance monitor
37	eSPI
38	GPIO 2
39	GPIO 1
...	-
52	SATA 1
53	SATA 2
...	-
60	DMA 1 channel 5
61	DMA 1 channel 6
62	DMA 1 channel 7
63	DMA 1 channel 8
64	DMA 2 channel 5
65	DMA 2 channel 6
66	DMA 2 channel 7
67	DMA 2 channel 8
68	Event processing unit 1
69	Event processing unit 2
70	GPIO 3
71	GPIO 4
72	SEC 5.2 job queue 1
73	SEC 5.2 job queue 2
74	SEC 5.2 job queue 3

Table continues on the next page...

Table 26-1. T2080 Internal Interrupt Assignments (continued)

Internal Interrupt Number	Interrupt Source
75	SEC 5.2 job queue 4
76	SEC 5.2 global error
77	Security monitor
78	Event processing unit 3
79	Event processing unit 4
80	Frame manager
...	-
84	Ethernet management interface (MDIO 1) - 1G LT
85	Ethernet management interface (MDIO 2) - 10G LT
88	Queue manager portal 0
89	Buffer manager portal 0
90	Queue manager portal 1
91	Buffer manager portal 1
92	Queue manager portal 2
93	Buffer manager portal 2
94	Queue manager portal 3
95	Buffer manager portal 3
96	Queue manager portal 4
97	Buffer manager portal 4
98	Queue manager portal 5
99	Buffer manager portal 5
100	Queue manager portal 6
101	Buffer manager portal 6
102	Queue manager portal 7
103	Buffer manager portal 7
104	Queue manager portal 8
105	Buffer manager portal 8
106	Queue manager portal 9
107	Buffer manager portal 9
108	Queue manager portal 10
109	Buffer manager portal 10
110	Queue manager portal 11
111	Buffer manager portal 11
112	Queue manager portal 12
113	Buffer manager portal 12
114	Queue manager portal 13
115	Buffer manager portal 13
116	Queue manager portal 14
117	Buffer manager portal 14

Table continues on the next page...

Table 26-1. T2080 Internal Interrupt Assignments (continued)

Internal Interrupt Number	Interrupt Source
118	Queue manager portal 15
119	Buffer manager portal 15
120	Queue manager portal 16
121	Buffer manager portal 16
122	Queue manager portal 17
123	Buffer manager portal 17
...	-
240	DMA 3 channel 1
241	DMA 3 channel 2
242	DMA 3 channel 3
243	DMA 3 channel 4
244	DMA 3 channel 5
245	DMA 3 channel 6
246	DMA 3 channel 7
247	DMA 3 channel 8

The table below shows the ORed error interrupt sources (see EISR0) that map to internal interrupt 0.Each assignment identifies which bit position in the MPIC's EISR status registers is assigned to report errors from the given block.

Table 26-2. ORed Error Interrupt Sources

ORed Interrupt	Interrupt Source	
0	-	
1	Frame manager error	
2	Buffer manager error	
3	Queue manager error	
4	Data compression engine error	
5	Pattern matching engine error	
...	-	
9	L2 cache 1 error	
...	-	
11	Rapid I/O error/Rapid I/O message manager (RMan) error	
...	-	
23	DDR Memory controller error	Memory error detect (DDR_ERR_DETECT)
...	-	
27	CoreNet platform cache (CPC) error	Error Detection and Reporting
...	-	

Table continues on the next page...

Table 26-2. ORed Error Interrupt Sources (continued)

ORed Interrupt	Interrupt Source	
29	Internal RAM multi-bit ECC error (MBEE)	
30	PAMU hardware error **NOTE:** Errors from all PAMUs are ORed together.	PAMU Operation Error Status register 1 (PAMU_POES1) and ECC Error Detect Register (PAMU_EEDET)
31	CoreNet coherency fabric (CCF) error	CEDR

26.3 PCI Express INT*x*/External Interrupt IRQ*n* Sharing

The virtual INTA signals of the four PCI Express controllers are routed to dedicated internal interrupts (24, 25, 26, and 27) in the MPIC. The other INT*x* signals are logically combined with the IRQ*n* signals so that they share the external interrupt controlled by the associated EIVPR*n* and EIDR*n* registers. the table below details the association of PCI Express virtual INT*x* signals to MPIC external interrupt IRQ*n* signals.

Table 26-3. PCI Express INTx/External Interrupt IRQn Sharing

PCI Express Controller	INT*x*	IRQ*n*
PCI Express 1	INTA	-
	INTB	IRQ1
	INTC	IRQ2
	INTD	IRQ3
PCI Express 2	INTA	-
	INTB	IRQ5
	INTC	IRQ6
	INTD	IRQ7
PCI Express 3	INTA	-
	INTB	IRQ9
	INTC	IRQ10
	INTD	IRQ11
PCI Express 4	INTA	-
	INTB	IRQ0
	INTC	IRQ4
	INTD	IRQ8

26.4 SoC Interrupt Event Routing

The table below shows the routing of the MPIC SoC interrupt event outputs (*sien*), which are used to signal events to the specified blocks on the SoC.

Table 26-4. SoC Interrupt Event Routing

SoC Interrupt Event	xILRn[INTTGT]	Event Routing	Section/Page
sie0	0xF0	DDR controller. Triggers panic button/battery backup event	DDR SDRAM control configuration 2 (DDR_DDR_SDRAM_CFG_2)
sie1	0xF1	PCI Express controllers (all). Triggers generation of outbound MSI transaction.	
sie2	0xF2	Reserved	-

Chapter 27
Device Configuration and Pin Control

27.1 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.

27.2 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 or hard 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
- Set of Local I/O Device Number (LIODN) registers for programming the LIODN of legacy peripherals which did not originally contain this register
- Two small sets of scratch registers:

- One set of read / write scratch registers
- One set of write-once / read scratch registers

27.3 Device Configuration Memory Map

The table below shows the memory-mapped CCSR registers of the Device Config module and lists the offset, name, and a cross-reference to the complete description of each register. These registers only support 32-bit accesses.

DCFG_CCSR memory map

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E_0000	POR Status Register 1 (DCFG_CCSR_PORSR1)	32	R	See section	27.3.1/1550
E_0004	POR Status Register 2 (DCFG_CCSR_PORSR2)	32	R	See section	27.3.2/1552
E_0020	General-Purpose POR Configuration Register (DCFG_CCSR_GPPORCR1)	32	R	See section	27.3.3/1553
E_0028	Fuse Status Register (DCFG_CCSR_FUSESR)	32	R	0000_0000h	27.3.4/1554
E_0070	Device Disable Register 1 (DCFG_CCSR_DEVDISR1)	32	R/W	0000_0000h	27.3.5/1555
E_0074	Device Disable Register 2 (DCFG_CCSR_DEVDISR2)	32	R/W	0000_0000h	27.3.6/1557
E_0078	Device Disable Register 3 (DCFG_CCSR_DEVDISR3)	32	R/W	0000_0000h	27.3.7/1559
E_007C	Device Disable Register 4 (DCFG_CCSR_DEVDISR4)	32	R/W	0000_0000h	27.3.8/1561
E_0080	Device Disable Register 5 (DCFG_CCSR_DEVDISR5)	32	R/W	0000_0000h	27.3.9/1562
E_0094	Core Disable Register (DCFG_CCSR_COREDISR)	32	R/W	0000_0000h	27.3.10/ 1563
E_00A0	Processor Version Register (DCFG_CCSR_PVR)	32	R	8040_0020h	27.3.11/ 1565
E_00A4	System Version Register (DCFG_CCSR_SVR)	32	R		27.3.12/ 1566
E_00B0	Reset Control Register (DCFG_CCSR_RSTCR)	32	R/W	0000_0000h	27.3.13/ 1567
E_00B4	Reset Request Preboot Loader Status Register (DCFG_CCSR_RSTRQPBLSR)	32	w1c	0000_0000h	27.3.14/ 1568
E_00C0	Reset Request Mask Register (DCFG_CCSR_RSTRQMR1)	32	R/W	0000_4000h	27.3.15/ 1568
E_00C8	Reset Request Status Register (DCFG_CCSR_RSTRQSR1)	32	w1c	0000_0000h	27.3.16/ 1571
E_00D4	Reset Request WDT Mask Register (DCFG_CCSR_RSTRQWDTMR)	32	R/W	0000_0000h	27.3.17/ 1574
E_00DC	Reset Request WDT Status Register (DCFG_CCSR_RSTRQWDTSR)	32	w1c	0000_0000h	27.3.18/ 1576

Table continues on the next page...

DCFG_CCSR memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E_00E4	Boot Release Register (DCFG_CCSR_BRR)	32	R/W	0000_0000h	27.3.19/ 1578
E_0100	Reset Control Word Status Register n (DCFG_CCSR_RCWSR1)	32	R	See section	27.3.20/ 1579
E_0104	Reset Control Word Status Register n (DCFG_CCSR_RCWSR2)	32	R	See section	27.3.20/ 1579
E_0108	Reset Control Word Status Register n (DCFG_CCSR_RCWSR3)	32	R	See section	27.3.20/ 1579
E_010C	Reset Control Word Status Register n (DCFG_CCSR_RCWSR4)	32	R	See section	27.3.20/ 1579
E_0110	Reset Control Word Status Register n (DCFG_CCSR_RCWSR5)	32	R	See section	27.3.20/ 1579
E_0114	Reset Control Word Status Register n (DCFG_CCSR_RCWSR6)	32	R	See section	27.3.20/ 1579
E_0118	Reset Control Word Status Register n (DCFG_CCSR_RCWSR7)	32	R	See section	27.3.20/ 1579
E_011C	Reset Control Word Status Register n (DCFG_CCSR_RCWSR8)	32	R	See section	27.3.20/ 1579
E_0120	Reset Control Word Status Register n (DCFG_CCSR_RCWSR9)	32	R	See section	27.3.20/ 1579
E_0124	Reset Control Word Status Register n (DCFG_CCSR_RCWSR10)	32	R	See section	27.3.20/ 1579
E_0128	Reset Control Word Status Register n (DCFG_CCSR_RCWSR11)	32	R	See section	27.3.20/ 1579
E_012C	Reset Control Word Status Register n (DCFG_CCSR_RCWSR12)	32	R	See section	27.3.20/ 1579
E_0130	Reset Control Word Status Register n (DCFG_CCSR_RCWSR13)	32	R	See section	27.3.20/ 1579
E_0134	Reset Control Word Status Register n (DCFG_CCSR_RCWSR14)	32	R	See section	27.3.20/ 1579
E_0138	Reset Control Word Status Register n (DCFG_CCSR_RCWSR15)	32	R	See section	27.3.20/ 1579
E_013C	Reset Control Word Status Register n (DCFG_CCSR_RCWSR16)	32	R	See section	27.3.20/ 1579
E_0200	Scratch Read / Write Register n (DCFG_CCSR_SCRATCHRW1)	32	R/W	0000_0000h	27.3.21/ 1579
E_0204	Scratch Read / Write Register n (DCFG_CCSR_SCRATCHRW2)	32	R/W	0000_0000h	27.3.21/ 1579
E_0208	Scratch Read / Write Register n (DCFG_CCSR_SCRATCHRW3)	32	R/W	0000_0000h	27.3.21/ 1579
E_020C	Scratch Read / Write Register n (DCFG_CCSR_SCRATCHRW4)	32	R/W	0000_0000h	27.3.21/ 1579
E_0300	Scratch Read Register n (DCFG_CCSR_SCRATCHW1R1)	32	R/W	0000_0000h	27.3.22/ 1580

Table continues on the next page...

DCFG_CCSR memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E_0304	Scratch Read Register n (DCFG_CCSR_SCRATCHW1R2)	32	R/W	0000_0000h	27.3.22/ 1580
E_0308	Scratch Read Register n (DCFG_CCSR_SCRATCHW1R3)	32	R/W	0000_0000h	27.3.22/ 1580
E_030C	Scratch Read Register n (DCFG_CCSR_SCRATCHW1R4)	32	R/W	0000_0000h	27.3.22/ 1580
E_0400	Core Reset Status Register n (DCFG_CCSR_CRSTSR)	32	R/W	0000_0000h	27.3.23/ 1580
E_0520	USB n Logical I/O Device Number register (DCFG_CCSR_USB1LIODNR)	32	R/W	0000_0000h	27.3.24/ 1583
E_0524	USB n Logical I/O Device Number register (DCFG_CCSR_USB2LIODNR)	32	R/W	0000_0000h	27.3.24/ 1583
E_0530	SD/MMC Logical I/O Device Number register (DCFG_CCSR_SDMMCLIODNR)	32	R/W	0000_0000h	27.3.25/ 1584
E_0550	SATA n Logical I/O Device Number register (DCFG_CCSR_SATA1LIODNR)	32	R/W	0000_0000h	27.3.26/ 1584
E_0554	SATA n Logical I/O Device Number register (DCFG_CCSR_SATA2LIODNR)	32	R/W	0000_0000h	27.3.26/ 1584
E_0580	DMA n Logical I/O Device Number register (DCFG_CCSR_DMA1LIODNR)	32	R/W	0000_0000h	27.3.27/ 1585
E_0584	DMA n Logical I/O Device Number register (DCFG_CCSR_DMA2LIODNR)	32	R/W	0000_0000h	27.3.27/ 1585
E_0588	DMA n Logical I/O Device Number register (DCFG_CCSR_DMA3LIODNR)	32	R/W	0000_0000h	27.3.27/ 1585
E_0600	Preboot Loader Status Register (DCFG_CCSR_PBLSR)	32	R	0400_0000h	27.3.28/ 1586
E_0604	PAMU Bypass Enable Register (DCFG_CCSR_PAMUBYPENR)	32	R	See section	27.3.29/ 1587
E_0608	DMA Control Register (DCFG_CCSR_DMACR1)	32	R/W	0000_0000h	27.3.30/ 1588
E_0704	DCSR Control Register (DCFG_CCSR_DCSRCR)	32	R/W	0000_0000h	27.3.31/ 1589
E_0740	Topology Initiator Type n Register (DCFG_CCSR_TP_ITYP0)	32	R	See section	27.3.32/ 1590
E_0744	Topology Initiator Type n Register (DCFG_CCSR_TP_ITYP1)	32	R	See section	27.3.32/ 1590
E_0748	Topology Initiator Type n Register (DCFG_CCSR_TP_ITYP2)	32	R	See section	27.3.32/ 1590
E_074C	Topology Initiator Type n Register (DCFG_CCSR_TP_ITYP3)	32	R	See section	27.3.32/ 1590
E_0750	Topology Initiator Type n Register (DCFG_CCSR_TP_ITYP4)	32	R	See section	27.3.32/ 1590
E_0754	Topology Initiator Type n Register (DCFG_CCSR_TP_ITYP5)	32	R	See section	27.3.32/ 1590

Table continues on the next page...

DCFG_CCSR memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E_0758	Topology Initiator Type n Register (DCFG_CCSR_TP_ITYP6)	32	R	See section	27.3.32/ 1590
E_075C	Topology Initiator Type n Register (DCFG_CCSR_TP_ITYP7)	32	R	See section	27.3.32/ 1590
E_0760	Topology Initiator Type n Register (DCFG_CCSR_TP_ITYP8)	32	R	See section	27.3.32/ 1590
E_0764	Topology Initiator Type n Register (DCFG_CCSR_TP_ITYP9)	32	R	See section	27.3.32/ 1590
E_0768	Topology Initiator Type n Register (DCFG_CCSR_TP_ITYP10)	32	R	See section	27.3.32/ 1590
E_076C	Topology Initiator Type n Register (DCFG_CCSR_TP_ITYP11)	32	R	See section	27.3.32/ 1590
E_0770	Topology Initiator Type n Register (DCFG_CCSR_TP_ITYP12)	32	R	See section	27.3.32/ 1590
E_0774	Topology Initiator Type n Register (DCFG_CCSR_TP_ITYP13)	32	R	See section	27.3.32/ 1590
E_0778	Topology Initiator Type n Register (DCFG_CCSR_TP_ITYP14)	32	R	See section	27.3.32/ 1590
E_077C	Topology Initiator Type n Register (DCFG_CCSR_TP_ITYP15)	32	R	See section	27.3.32/ 1590
E_0780	Topology Initiator Type n Register (DCFG_CCSR_TP_ITYP16)	32	R	See section	27.3.32/ 1590
E_0784	Topology Initiator Type n Register (DCFG_CCSR_TP_ITYP17)	32	R	See section	27.3.32/ 1590
E_0788	Topology Initiator Type n Register (DCFG_CCSR_TP_ITYP18)	32	R	See section	27.3.32/ 1590
E_078C	Topology Initiator Type n Register (DCFG_CCSR_TP_ITYP19)	32	R	See section	27.3.32/ 1590
E_0790	Topology Initiator Type n Register (DCFG_CCSR_TP_ITYP20)	32	R	See section	27.3.32/ 1590
E_0794	Topology Initiator Type n Register (DCFG_CCSR_TP_ITYP21)	32	R	See section	27.3.32/ 1590
E_0798	Topology Initiator Type n Register (DCFG_CCSR_TP_ITYP22)	32	R	See section	27.3.32/ 1590
E_079C	Topology Initiator Type n Register (DCFG_CCSR_TP_ITYP23)	32	R	See section	27.3.32/ 1590
E_07A0	Topology Initiator Type n Register (DCFG_CCSR_TP_ITYP24)	32	R	See section	27.3.32/ 1590
E_07A4	Topology Initiator Type n Register (DCFG_CCSR_TP_ITYP25)	32	R	See section	27.3.32/ 1590
E_07A8	Topology Initiator Type n Register (DCFG_CCSR_TP_ITYP26)	32	R	See section	27.3.32/ 1590
E_07AC	Topology Initiator Type n Register (DCFG_CCSR_TP_ITYP27)	32	R	See section	27.3.32/ 1590

Table continues on the next page...

DCFG_CCSR memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E_07B0	Topology Initiator Type n Register (DCFG_CCSR_TP_ITYP28)	32	R	See section	27.3.32/ 1590
E_07B4	Topology Initiator Type n Register (DCFG_CCSR_TP_ITYP29)	32	R	See section	27.3.32/ 1590
E_07B8	Topology Initiator Type n Register (DCFG_CCSR_TP_ITYP30)	32	R	See section	27.3.32/ 1590
E_07BC	Topology Initiator Type n Register (DCFG_CCSR_TP_ITYP31)	32	R	See section	27.3.32/ 1590
E_07C0	Topology Initiator Type n Register (DCFG_CCSR_TP_ITYP32)	32	R	See section	27.3.32/ 1590
E_07C4	Topology Initiator Type n Register (DCFG_CCSR_TP_ITYP33)	32	R	See section	27.3.32/ 1590
E_07C8	Topology Initiator Type n Register (DCFG_CCSR_TP_ITYP34)	32	R	See section	27.3.32/ 1590
E_07CC	Topology Initiator Type n Register (DCFG_CCSR_TP_ITYP35)	32	R	See section	27.3.32/ 1590
E_07D0	Topology Initiator Type n Register (DCFG_CCSR_TP_ITYP36)	32	R	See section	27.3.32/ 1590
E_07D4	Topology Initiator Type n Register (DCFG_CCSR_TP_ITYP37)	32	R	See section	27.3.32/ 1590
E_07D8	Topology Initiator Type n Register (DCFG_CCSR_TP_ITYP38)	32	R	See section	27.3.32/ 1590
E_07DC	Topology Initiator Type n Register (DCFG_CCSR_TP_ITYP39)	32	R	See section	27.3.32/ 1590
E_07E0	Topology Initiator Type n Register (DCFG_CCSR_TP_ITYP40)	32	R	See section	27.3.32/ 1590
E_07E4	Topology Initiator Type n Register (DCFG_CCSR_TP_ITYP41)	32	R	See section	27.3.32/ 1590
E_07E8	Topology Initiator Type n Register (DCFG_CCSR_TP_ITYP42)	32	R	See section	27.3.32/ 1590
E_07EC	Topology Initiator Type n Register (DCFG_CCSR_TP_ITYP43)	32	R	See section	27.3.32/ 1590
E_07F0	Topology Initiator Type n Register (DCFG_CCSR_TP_ITYP44)	32	R	See section	27.3.32/ 1590
E_07F4	Topology Initiator Type n Register (DCFG_CCSR_TP_ITYP45)	32	R	See section	27.3.32/ 1590
E_07F8	Topology Initiator Type n Register (DCFG_CCSR_TP_ITYP46)	32	R	See section	27.3.32/ 1590
E_07FC	Topology Initiator Type n Register (DCFG_CCSR_TP_ITYP47)	32	R	See section	27.3.32/ 1590
E_0800	Topology Initiator Type n Register (DCFG_CCSR_TP_ITYP48)	32	R	See section	27.3.32/ 1590
E_0804	Topology Initiator Type n Register (DCFG_CCSR_TP_ITYP49)	32	R	See section	27.3.32/ 1590

Table continues on the next page...

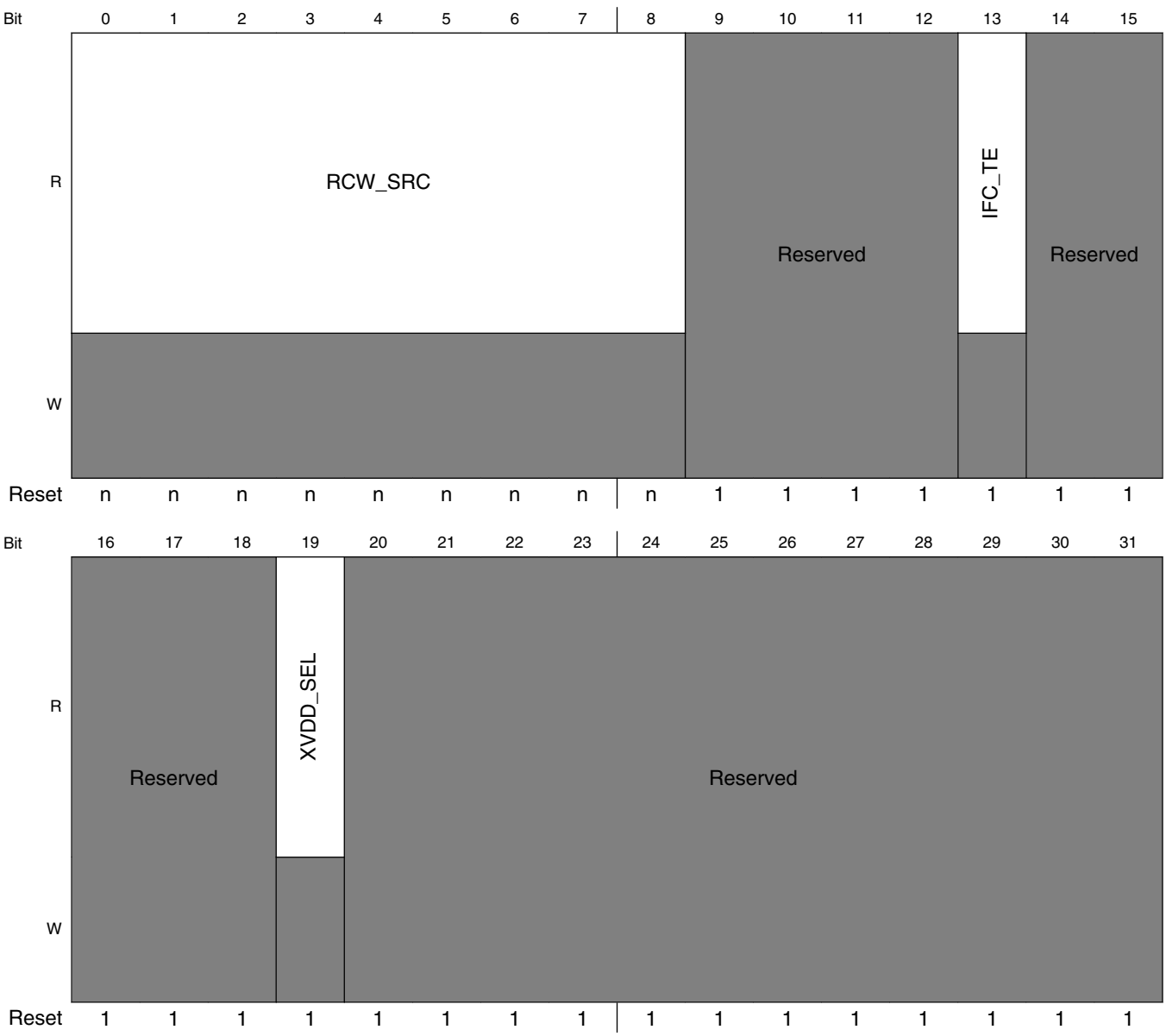
DCFG_CCSR memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
E_0808	Topology Initiator Type n Register (DCFG_CCSR_TP_ITYP50)	32	R	See section	27.3.32/ 1590
E_080C	Topology Initiator Type n Register (DCFG_CCSR_TP_ITYP51)	32	R	See section	27.3.32/ 1590
E_0810	Topology Initiator Type n Register (DCFG_CCSR_TP_ITYP52)	32	R	See section	27.3.32/ 1590
E_0814	Topology Initiator Type n Register (DCFG_CCSR_TP_ITYP53)	32	R	See section	27.3.32/ 1590
E_0818	Topology Initiator Type n Register (DCFG_CCSR_TP_ITYP54)	32	R	See section	27.3.32/ 1590
E_081C	Topology Initiator Type n Register (DCFG_CCSR_TP_ITYP55)	32	R	See section	27.3.32/ 1590
E_0820	Topology Initiator Type n Register (DCFG_CCSR_TP_ITYP56)	32	R	See section	27.3.32/ 1590
E_0824	Topology Initiator Type n Register (DCFG_CCSR_TP_ITYP57)	32	R	See section	27.3.32/ 1590
E_0828	Topology Initiator Type n Register (DCFG_CCSR_TP_ITYP58)	32	R	See section	27.3.32/ 1590
E_082C	Topology Initiator Type n Register (DCFG_CCSR_TP_ITYP59)	32	R	See section	27.3.32/ 1590
E_0830	Topology Initiator Type n Register (DCFG_CCSR_TP_ITYP60)	32	R	See section	27.3.32/ 1590
E_0834	Topology Initiator Type n Register (DCFG_CCSR_TP_ITYP61)	32	R	See section	27.3.32/ 1590
E_0838	Topology Initiator Type n Register (DCFG_CCSR_TP_ITYP62)	32	R	See section	27.3.32/ 1590
E_083C	Topology Initiator Type n Register (DCFG_CCSR_TP_ITYP63)	32	R	See section	27.3.32/ 1590
E_0844	Core Cluster n Topology Register (DCFG_CCSR_TP_CLUSTER1)	32	R	See section	27.3.33/ 1591
E_0A00	QMBM Warm Reset Control Register (DCFG_CCSR_QMBM_WARMRST)	32	W	See section	27.3.34/ 1592
E_0E60	DDR Clock Disable Register (DCFG_CCSR_DDRCLKDR)	32	R/W	0000_0000h	27.3.35/ 1593
E_0E68	IFC Clock Disable Register (DCFG_CCSR_IFCCLKDR)	32	R/W	0000_0000h	27.3.36/ 1595

27.3.1 POR Status Register 1 (DCFG_CCSR_PORSR1)

PORSR1 captures the values of the device's POR configuration pins.

Address: 0h base + E_0000h offset = E_0000h



DCFG_CCSR_PORSR1 field descriptions

Field	Description
0–8 RCW_SRC	Reset Configuration Word Source. Indicates which Flash memory contains the RCW information. Loaded with the value of CFG_RCW_SRC[0:8].

Table continues on the next page...

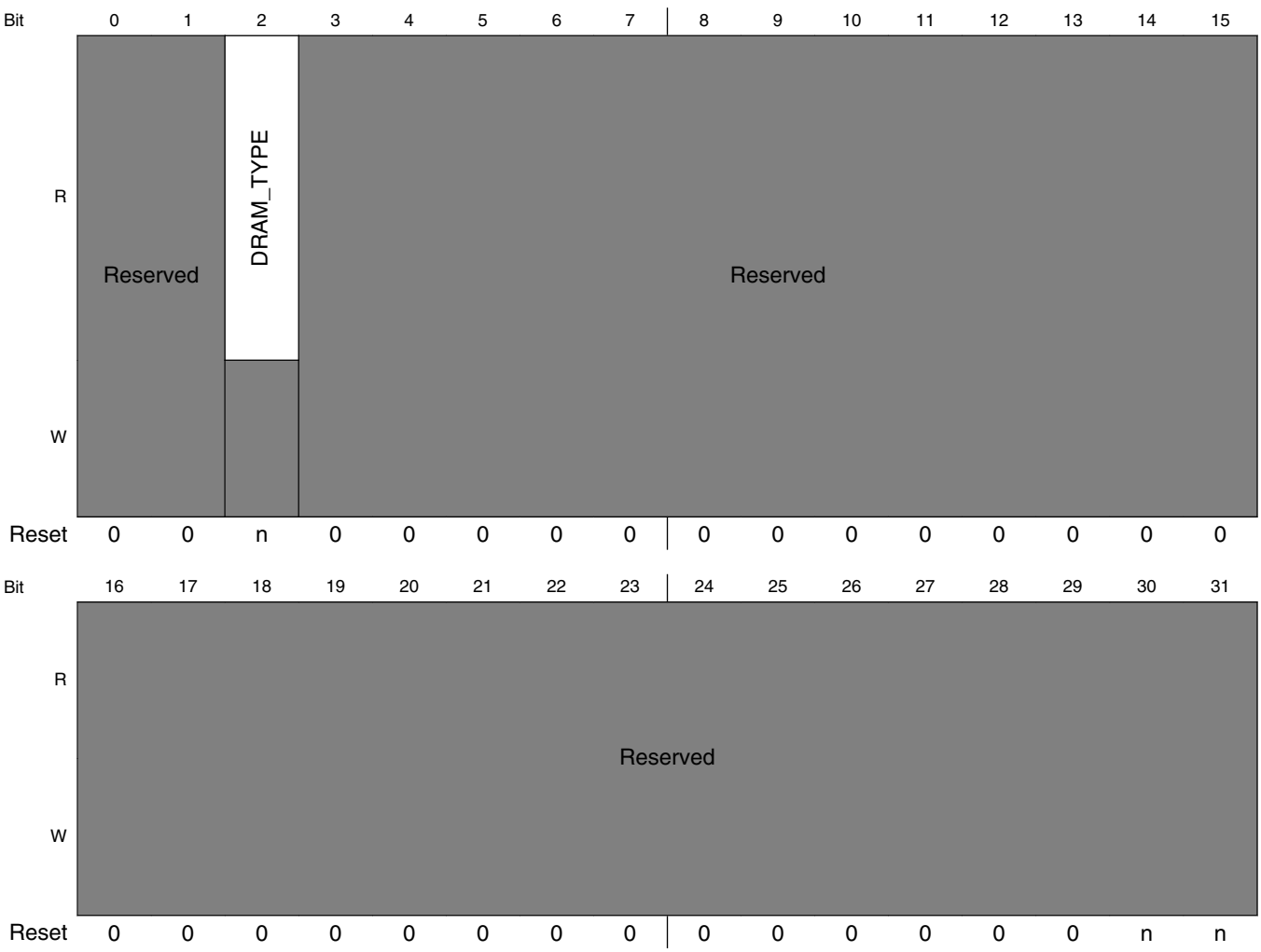
DCFG_CCSR_PORSR1 field descriptions (continued)

Field	Description
9–12 -	This field is reserved. Reserved
13 IFC_TE	IFC External Transceive Enable Polarity Selection. Loaded with the value of the CFG_IFC_TE pin at power-on reset. 0 Transceiver Enable polarity is configured active high 1 Transceiver Enable polarity is configured active low
14–18 -	This field is reserved. Reserved
19 XVDD_SEL	SerDes XVDD voltage select 0 Reserved 1 1.35 V
20–31 -	This field is reserved. Reserved

27.3.2 POR Status Register 2 (DCFG_CCSR_PORSR2)

PORSR2 captures the values of the device's POR configuration pins.

Address: 0h base + E_0004h offset = E_0004h



DCFG_CCSR_PORSR2 field descriptions

Field	Description
0–1 -	This field is reserved. reserved
2 DRAM_TYPE	DRAM Type Selector. 0 DDR3 technology (1.5 V) 1 DDR3L technology (1.35 V)
3–31 -	This field is reserved. Reserved

27.3.3 General-Purpose POR Configuration Register (DCFG_CCSR_GPPORCR1)

GPPORCR1 stores the value sampled from the integrated Flash controller address/data signals, IFC_AD[0:7], using sampling logic similar to that for the configuration pins It is provided for customer use and always samples 8-bits, independent of the width of an SoC flash controller's LAD bus. Software can use this value to inform the operating system about initial system configuration. Typical interpretations include circuit board type, board ID number, or a list of available peripherals.

Address: 0h base + E_0020h offset = E_0020h

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	POR_CFG_VEC								Reserved																							
W																																
Reset	*	*	*	*	*	*	*	*	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

* Notes:
- POR_CFG_VEC field: Reset value supplied by cfg_gpinput[0:7].

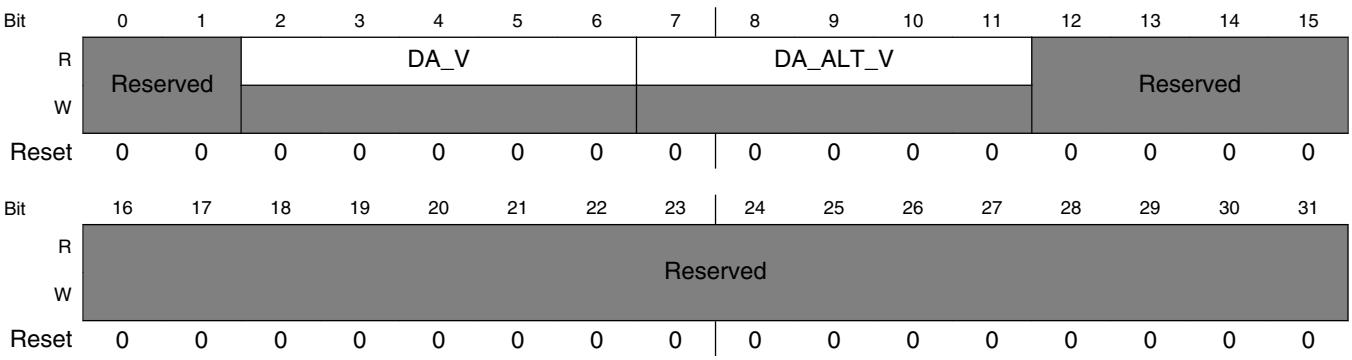
DCFG_CCSR_GPPORCR1 field descriptions

Field	Description
0–7 POR_CFG_VEC	General-purpose POR configuration vector sampled from integrated Flash controller address/data signals, IFC_AD[0:7] signals (uppermost 8-bits) at the negation of PORESET. Note that if nothing is driven on these signals during reset, the value of this register is indeterminate. **NOTE:** Reset value supplied by cfg_gpinput[0:7].
8–31 -	This field is reserved. Reserved

27.3.4 Fuse Status Register (DCFG_CCSR_FUSESR)

The fuse status register provides the values from on-chip voltage ID efuses programmed at the factory. These values define the voltage requirements for the chip. Boot software reads FUSESR and translates the values into the appropriate commands to set the voltage output value of an external voltage regulator.

Address: 0h base + E_0028h offset = E_0028h



DCFG_CCSR_FUSESR field descriptions

Field	Description
0–1 -	This field is reserved. Reserved
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. 00000 unused (default) 00001 0.9875 V 00010 0.9750 V 00011 0.9625 V 00100 0.9500 V 00101 0.9375 V 00110 0.9250 V 00111 0.9125 V 01000 0.9000 V 01001 0.8875 V 01010 0.8750 V 01011 0.8625 V 01100 0.8500 V 01101 0.8375 V 01110 0.8250 V 01111 0.8125 V 10000 1.0000 V 10001 1.0125 V

Table continues on the next page...

DCFG_CCSR_FUSESR field descriptions (continued)

Field	Description
	10010 1.0250 V 10011 1.0375 V 10100 1.0500 V 10101 1.0625 V 10110 1.0750 V 10111 1.0875 V 11000 1.1000 V 11001 Reserved 11010 Reserved 11011 Reserved 11100 Reserved 11101 Reserved 11110 Reserved 11111 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 -	This field is reserved. Reserved

27.3.5 Device Disable Register 1 (DCFG_CCSR_DEVDISR1)

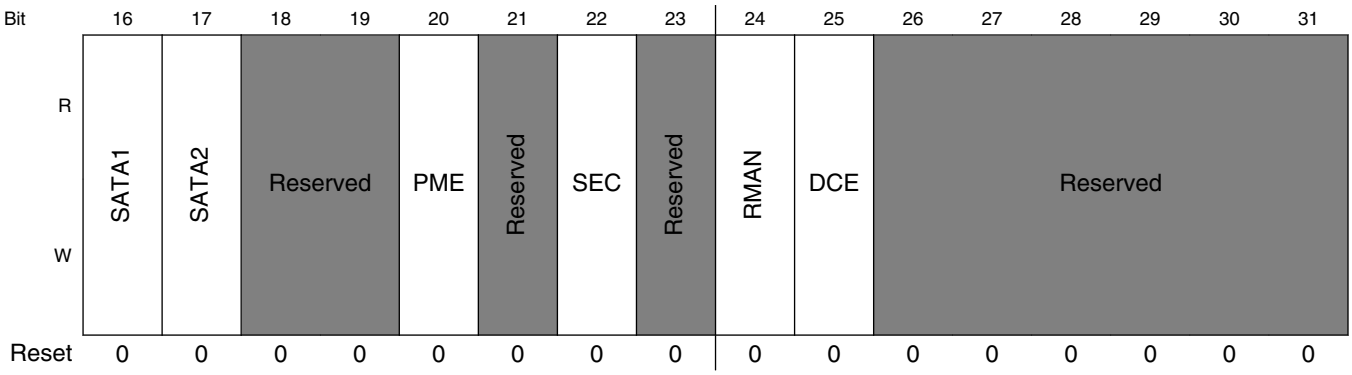
A given application may not use all the peripherals on the device. In this case, it may be desirable to disable unused peripherals. DEVDISR1 provides a mechanism for gating clocks to IP blocks that are not used when running an application.

NOTE

IP Blocks disabled by setting the corresponding bit in the DEVDISR1 register must not be re-enabled.

Address: 0h base + E_0070h offset = E_0070h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	PBL	PMAN	ESDHC	Reserved					DMA1	DMA2	DMA3	Reserved	USB1	USB2	Reserved	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



DCFG_CCSR_DEVDISR1 field descriptions

Field	Description
0 PBL	Pre-boot loader disable. 0 Module is enabled 1 Module is disabled
1 PMAN	Prefetch manager disable. 0 Module is enabled 1 Module is disabled
2 ESDHC	eSDHC controller disable. 0 Module is enabled 1 Module is disabled
3–7 -	This field is reserved. Reserved
8 DMA1	DMA controller 1 disable. 0 Module is enabled 1 Module is disabled
9 DMA2	DMA controller 2 disable. 0 Module is enabled 1 Module is disabled
10 DMA3	DMA controller 3 disable. 0 Module is enabled 1 Module is disabled
11 -	This field is reserved. Reserved
12 USB1	USB controller 1 disable. 0 Module is enabled 1 Module is disabled
13 USB2	USB controller 2 disable. 0 Module is enabled 1 Module is disabled

Table continues on the next page...

DCFG_CCSR_DEVDISR1 field descriptions (continued)

Field	Description
14–15 -	This field is reserved. Reserved
16 SATA1	SATA controller 1 disable. 0 Module is enabled 1 Module is disabled
17 SATA2	SATA controller 2 disable. 0 Module is enabled 1 Module is disabled
18–19 -	This field is reserved. Reserved
20 PME	Pattern matching engine disable. 0 Module is enabled 1 Module is disabled
21 -	This field is reserved. Reserved
22 SEC	SEC module disable. 0 Module is enabled 1 Module is disabled
23 -	This field is reserved. Reserved
24 RMAN	RapidIO message manager disable. 0 Module is enabled 1 Module is disabled
25 DCE	Data compression engine disable. 0 Module is enabled 1 Module is disabled
26–31 -	This field is reserved. Reserved

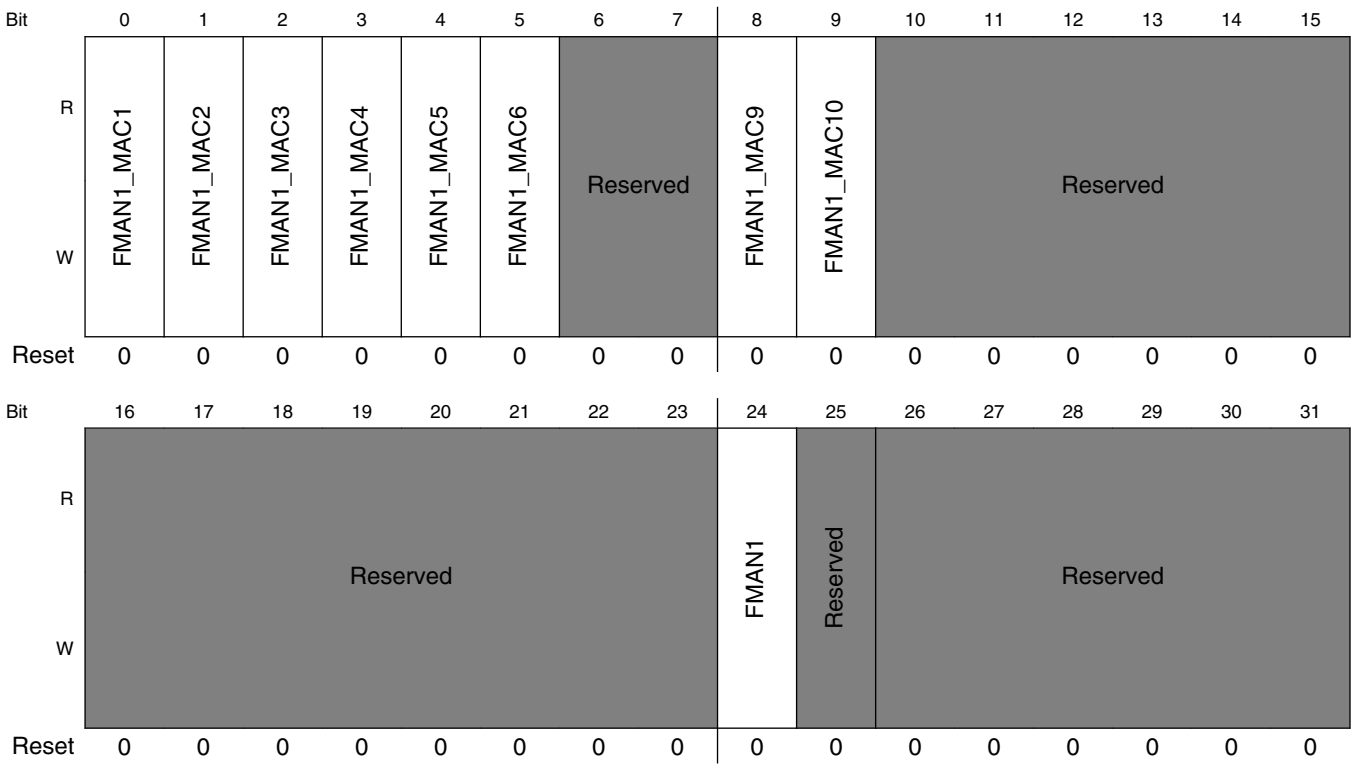
27.3.6 Device Disable Register 2 (DCFG_CCSR_DEVDISR2)

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.

NOTE

IP Blocks disabled by setting the corresponding bit in the DEVDISR2 register must not be re-enabled.

Address: 0h base + E_0074h offset = E_0074h



DCFG_CCSR_DEVDISR2 field descriptions

Field	Description
0 FMAN1_MAC1	Frame manager 1, MAC1 disable. 0 Module is enabled 1 Module is disabled
1 FMAN1_MAC2	Frame manager 1, MAC2 disable. 0 Module is enabled 1 Module is disabled
2 FMAN1_MAC3	Frame manager 1, MAC3 disable. 0 Module is enabled 1 Module is disabled
3 FMAN1_MAC4	Frame manager 1, MAC4 disable. 0 Module is enabled 1 Module is disabled
4 FMAN1_MAC5	Frame manager 1, MAC5 disable. 0 Module is enabled 1 Module is disabled
5 FMAN1_MAC6	Frame manager 1, MAC6 disable. 0 Module is enabled 1 Module is disabled

Table continues on the next page...

DCFG_CCSR_DEVDISR2 field descriptions (continued)

Field	Description
6–7 -	This field is reserved. Reserved
8 FMAN1_MAC9	Frame manager 1, MAC9 disable. 0 Module is enabled 1 Module is disabled
9 FMAN1_MAC10	Frame manager 1, MAC10 disable. 0 Module is enabled 1 Module is disabled
10–23 -	This field is reserved. Reserved
24 FMAN1	Frame manager 1 disable. 0 Module is enabled 1 Module is disabled
25 -	This field is reserved. Reserved
26–31 -	This field is reserved. Reserved

27.3.7 Device Disable Register 3 (DCFG_CCSR_DEVDISR3)

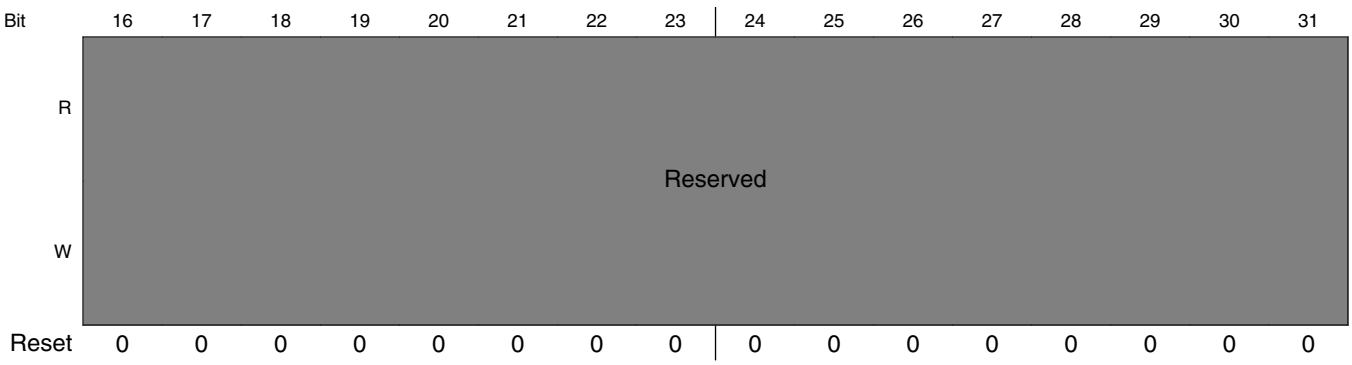
A given application may not use all the peripherals on the device. In this case, it may be desirable to disable unused peripherals. DEVDISR3 provides a mechanism for gating clocks to IP blocks that are not used when running an application.

NOTE

IP Blocks disabled by setting the corresponding bit in the DEVDISR3 register must not be re-enabled.

Address: 0h base + E_0078h offset = E_0078h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	PEX1	PEX2	PEX3	PEX4	SRIO1	SRIO2	Reserved						QMAN	BMAN	Reserved	Reserved
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



DCFG_CCSR_DEVDISR3 field descriptions

Field	Description
0 PEX1	PCI Express controller 1 disable. 0 Module is enabled 1 Module is disabled
1 PEX2	PCI Express controller 2 disable. 0 Module is enabled 1 Module is disabled
2 PEX3	PCI Express controller 3 disable. 0 Module is enabled 1 Module is disabled
3 PEX4	PCI Express controller 4 disable. 0 Module is enabled 1 Module is disabled
4 SRIO1	Serial RapidIO controller 1 disable. 0 Module is enabled 1 Module is disabled
5 SRIO2	Serial RapidIO controller 2 disable. 0 Module is enabled 1 Module is disabled
6–11 -	This field is reserved. Reserved
12 QMAN	Queue manager disable. 0 Module is enabled 1 Module is disabled
13 BMAN	Buffer manager disable. 0 Module is enabled 1 Module is disabled
14 -	This field is reserved. Reserved

Table continues on the next page...

DCFG_CCSR_DEVDISR3 field descriptions (continued)

Field	Description
15–31 -	This field is reserved. Reserved

27.3.8 Device Disable Register 4 (DCFG_CCSR_DEVDISR4)

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.

Address: 0h base + E_007Ch offset = E_007Ch

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	I2C1	I2C2	DUART1	DUART2	ESPI	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

DCFG_CCSR_DEVDISR4 field descriptions

Field	Description
0 I2C1	Dual I2C module 1 (I2C1 and I2C2) disable. 0 Module is enabled 1 Module is disabled
1 I2C2	Dual I2C module 2 (I2C3 and I2C4) disable. 0 Module is enabled 1 Module is disabled
2 DUART1	DUART1 module disable. 0 Module is enabled 1 Module is disabled

Table continues on the next page...

DCFG_CCSR_DEVDISR4 field descriptions (continued)

Field	Description
3 DUART2	DUART2 module disable. 0 Module is enabled 1 Module is disabled
4 ESPI	eSPI module disable. 0 Module is enabled 1 Module is disabled
5–31 -	This field is reserved. Reserved

27.3.9 Device Disable Register 5 (DCFG_CCSR_DEVDISR5)

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.

Address: 0h base + E_0080h offset = E_0080h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DDR	Reserved			CPC	Reserved			IFC	GPIO	DBG	NAL	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

DCFG_CCSR_DEVDISR5 field descriptions

Field	Description
0 DDR	DDR Controller disable 0 Module is enabled 1 Module is disabled
1–3 -	This field is reserved. Reserved
4 CPC	CoreNet platform cache 1 disable. 0 Module is enabled 1 Module is disabled

Table continues on the next page...

DCFG_CCSR_DEVDISR5 field descriptions (continued)

Field	Description
5–7 -	This field is reserved. Reserved
8 IFC	Integrated Flash controller disable. 0 Module is enabled 1 Module is disabled
9 GPIO	GPIO disable. **NOTE:** This field disables all GPIO modules. 0 Module is enabled 1 Module is disabled
10 DBG	Debug module disable. 0 Module is enabled 1 Module is disabled
11 NAL	Nexus/Aurora link layer disable. 0 Module is enabled 1 Module is disabled
12–31 -	This field is reserved. Reserved

27.3.10 Core Disable Register (DCFG_CCSR_COREDISR)

COREDISR provides a mechanism for gating clocks to any cores on the device that are not used when running an application.

COREDISR register should only be configured in the following conditions:

- Before system ready, COREDISR register can be programmed by the external debugger or Pre-Boot Initialization.
- After system ready, a COREDISR register bit can be programmed for the corresponding core by the external debugger or embedded software while the core is in boot-holdoff mode.

NOTE

Cores that have an interrupt pending will not be disabled by COREDISR until the interrupt is cleared.

NOTE

Cores disabled by setting the corresponding bit in the COREDISR register must not be re-enabled via software. The only supported mechanism for re-enabling a Core disabled in this manner is via power-on, hard reset, or core reset.

NOTE

The LSB, bit 31, is associated with Core 0.

Address: 0h base + E_0094h offset = E_0094h

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												CD3	CD2	CD1	CD0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DCFG_CCSR_COREDISR field descriptions

Field	Description
0–27 -	This field is reserved. Reserved
28 CD3	Core 3 Disable. 0 Selected Core is enabled 1 Selected Core is disabled
29 CD2	Core 2 Disable. 0 Selected Core is enabled 1 Selected Core is disabled
30 CD1	Core 1 Disable. 0 Selected Core is enabled 1 Selected Core is disabled
31 CD0	Core 0 Disable. 0 Selected Core is enabled 1 Selected Core is disabled

27.3.11 Processor Version Register (DCFG_CCSR_PVR)

The PVR contains the processor version number. It is a memory-mapped copy of the PVR in core 0 (and is therefore accessible to external devices).

Address: 0h base + E_00A0h offset = E_00A0h

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

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

DCFG_CCSR_PVR field descriptions

Field	Description
0–3 MFR_ID	Manufacturer ID
4–5 -	This field is reserved. Reserved
6–9 PROC_TYPE	Processor type
10–15 PROC_ID	Processor ID
16–23 MAJOR_REV	Major Revision Number
24–31 MINOR_REV	Minor Revision Number

27.3.12 System Version Register (DCFG_CCSR_SVR)

The SVR contains the system version number for the device. This value can also be read though the SVR SPR of the e6500 core.

Address: 0h base + E_00A4h offset = E_00A4h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	MFR_ID				SOC_DEV_ID								SEC	VAR_PER		
W																
Reset	1	0	0	0	0	1	0	1	0	0	1	1	n	0	0	n

Bit	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	n	0	0	0	0	1	0	0	0	0

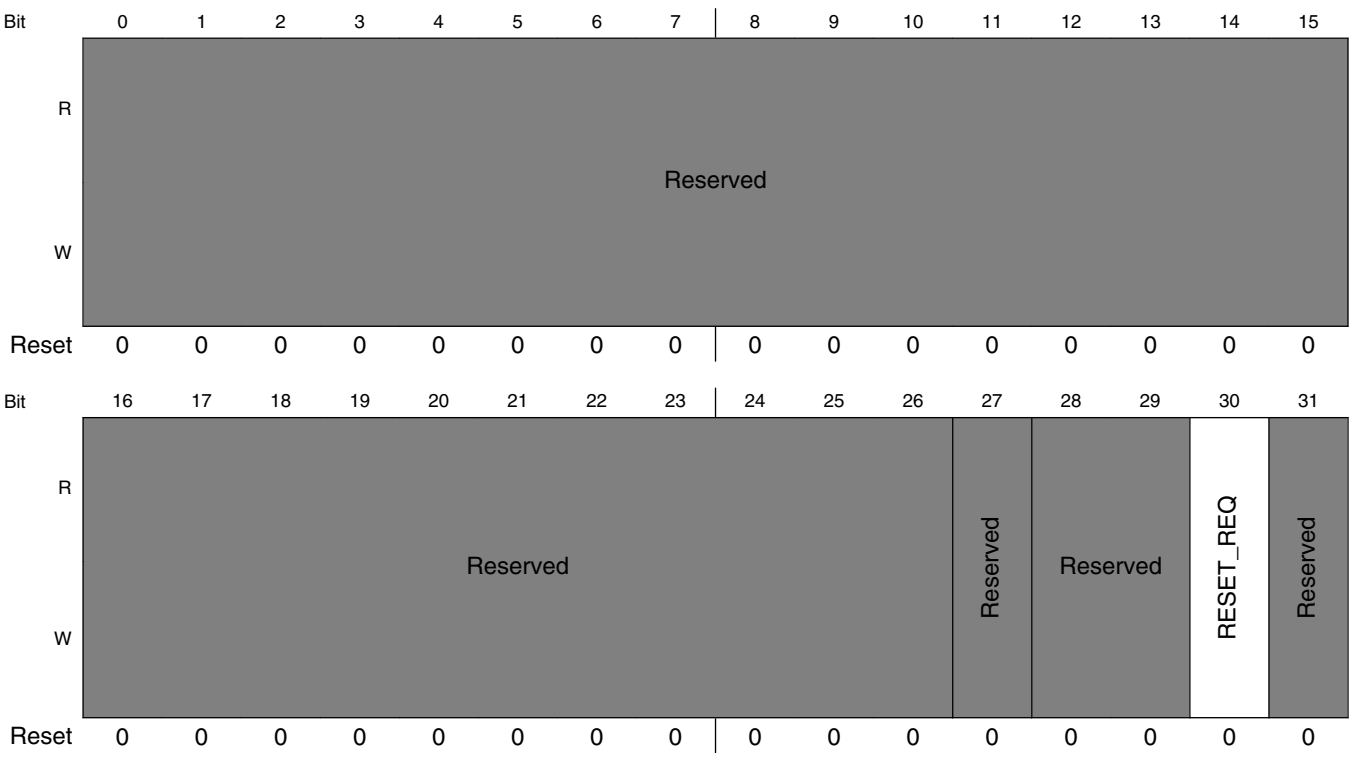
DCFG_CCSR_SVR field descriptions

Field	Description
0–3 MFR_ID	Manufacturer ID
4–11 SOC_DEV_ID	SoC Device ID
12 SEC	Security 0 No security 1 Security enabled
13–23 VAR_PER	Various Personalities 000_0000_0000 T2080 001_0000_0000 T2081
24–27 MAJOR_REV	Major Revision Number 0001 - Rev 1.0
28–31 MINOR_REV	Minor Revision Number 0000

27.3.13 Reset Control Register (DCFG_CCSR_RSTCR)

The RSTCR allows software to control reset functions.

Address: 0h base + E_00B0h offset = E_00B0h



DCFG_CCSR_RSTCR field descriptions

Field	Description
0–26 -	This field is reserved. Reserved
27 -	This field is reserved. Reserved
28–29 -	This field is reserved. Reserved
30 RESET_REQ	Hardware reset request. External hardware may then decide to issue the desired reset signal (PORESET_B or HRESET_B) to the device. 0 No reset request initiated. 1 Hardware reset request initiated by software.
31 -	This field is reserved. Reserved

27.3.14 Reset Request Preboot Loader Status Register (DCFG_CCSR_RSTRQPBLSR)

The RSTRQPBLSR contains status bits to record the reasons for RESET_REQ_B assertion. Excludes core watchdog timer sources.

NOTE

This register's code is valid only when RSTRQSR[PBL_RR] is set.

NOTE

ECC errors detected during NAND flash loading during RCW and PBI phases are reported in RSTRQSR[IFC_RR]. See the description of this bit for more details.

Address: 0h base + E_00B4h offset = E_00B4h

Bit	0–7	8–14	15	16–31
R	Reserved	ERR_CODE		Reserved
W		w1c		
Reset	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0	0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

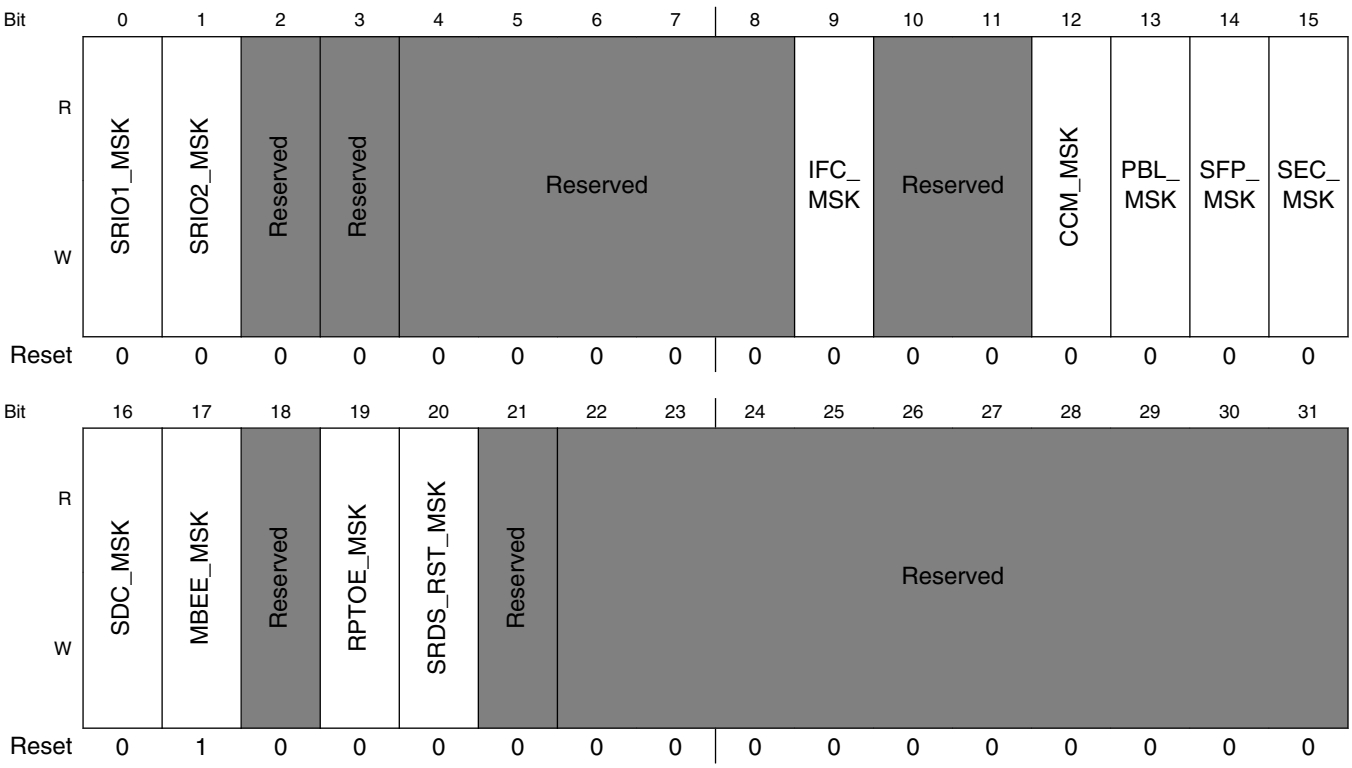
DCFG_CCSR_RSTRQPBLSR field descriptions

Field	Description
0–7 -	This field is reserved. 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 the PBL chapter for details on the PBL error encodings.
15–31 -	This field is reserved. Reserved

27.3.15 Reset Request Mask Register (DCFG_CCSR_RSTRQMR1)

The RSTRQMR contains mask bits for optional masking of RESET_REQ_B sources to prevent generation of such a reset request. Excludes core watchdog timer sources.

Address: 0h base + E_00C0h offset = E_00C0h



DCFG_CCSR_RSTRQMR1 field descriptions

Field	Description
0 SRIO1_MSK	Rapid I/O 1 reset request mask 0 Rapid I/O 1 error event can cause a reset request 1 Rapid I/O 1 error event cannot cause a reset request
1 SRIO2_MSK	Rapid I/O 2 reset request mask 0 Rapid I/O 2 error event can cause a reset request 1 Rapid I/O 2 error event cannot cause a reset request
2 -	This field is reserved. Reserved
3 -	This field is reserved. Reserved
4–8 -	This field is reserved. Reserved
9 IFC_MSK	Integrated Flash Controller reset request event mask 0 IFC error event can cause a reset request 1 IFC error event cannot cause a reset request
10–11 -	This field is reserved. Reserved
12 CCM_MSK	Coherency Fabric reset request event mask

Table continues on the next page...

DCFG_CCSR_RSTRQMR1 field descriptions (continued)

Field	Description
	0 Coherency Module write miss event can cause a reset request 1 Coherency Module write miss event cannot cause a reset request
13 PBL_MSK	PBL error reset request event mask. 0 PBL error event can cause a reset request 1 PBL error event cannot cause a reset request
14 SFP_MSK	Security Fuse Processor error during POR fuse process reset mask. 0 Security Fuse Processor error event can cause a reset request 1 Security Fuse Processor error event cannot cause a reset request
15 SEC_MSK	Security Controller reset request event mask 0 Security Controller Reset event can cause a reset request 1 Security Controller Reset event cannot cause a reset request
16 SDC_MSK	Security Debug Controller error reset request mask 0 SDC error event can cause a reset request 1 SDC error event cannot cause a reset request
17 MBEE_MSK	Multi-bit ECC error reset request mask 0 Multi-bit ECC error event can cause a reset request 1 Multi-bit ECC error event cannot cause a reset request
18 -	This field is reserved. Reserved 0 POR BIST error event can cause a reset request 1 POR BIST error event cannot cause a reset request
19 RPTOE_MSK	RCPM Time Out reset request event mask 0 RCPM Time Out event can cause a reset request 1 RCPM Time Out event cannot cause a reset request
20 SRDS_RST_ MSK	SerDes reset request event mask 0 SerDes reset event can cause a reset request 1 SerDes reset event cannot cause a reset request
21 -	This field is reserved. Reserved
22–31 -	This field is reserved. Reserved

27.3.16 Reset Request Status Register (DCFG_CCSR_RSTRQSR1)

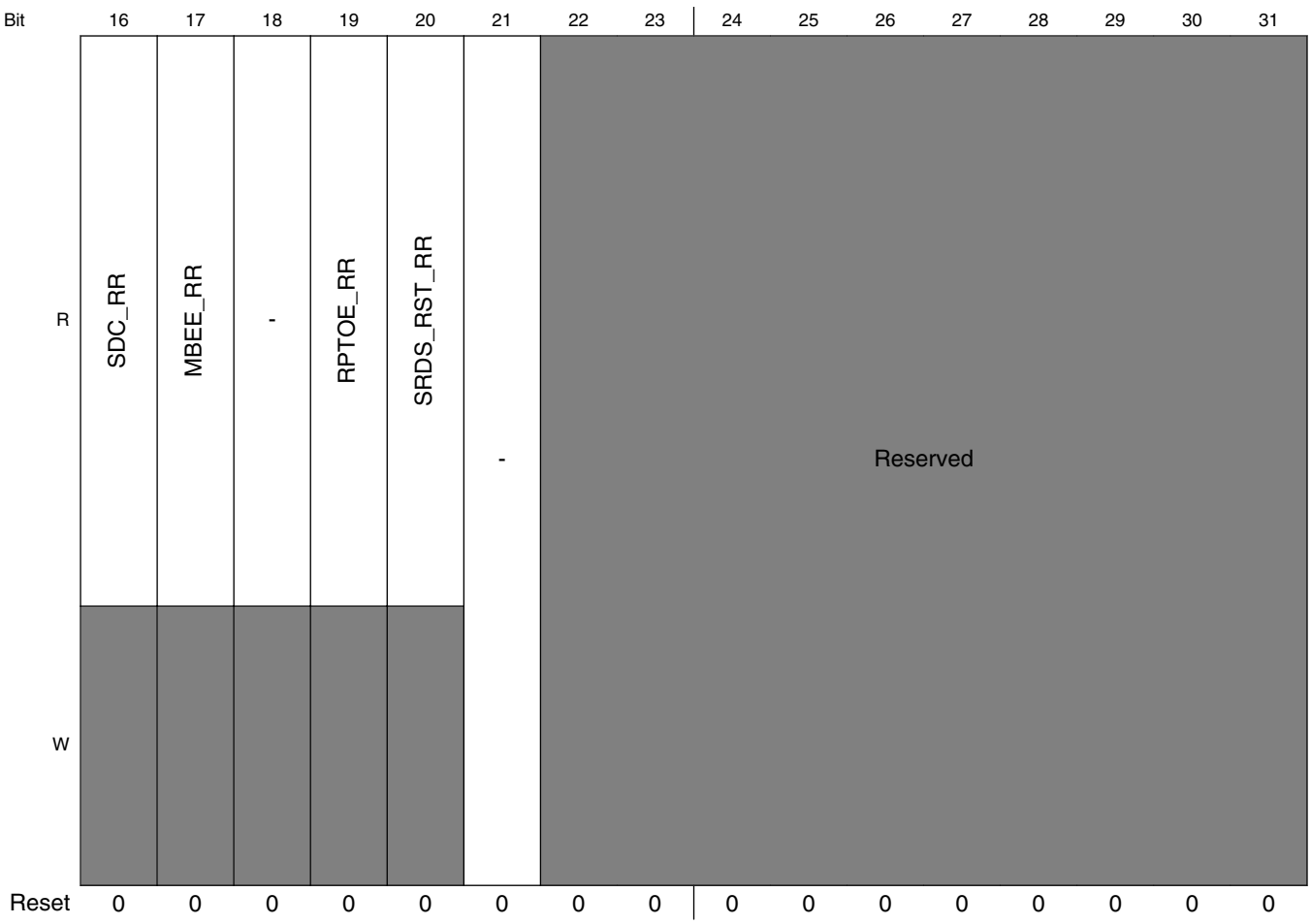
The RSTRQSR contains status bits to record the reasons for RESET_REQ_B assertion. The bits here are set independent of the RSTRQMR. This means if a reset reason occurs and is masked by RSTRQMR, it will still be recoreded in RSTRQSR. Excludes core watchdog timer sources (see Reset Request WDT Status Register (DCFG_CCSR_RSTRQWDTSR)).

NOTE

For the different sources captured in this register, some of these can be serviced with either PORESET_B or HRESET_B and some of these must be serviced with PORESET_B only.

Address: 0h base + E_00C8h offset = E_00C8h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	SRIO1_RR	SRIO2_RR	Reserved	Reserved	Reserved					IFC_RR	WDT_RR	SW_RR	CCM_RR	PBL_RR	SFP_RR	SEC_RR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



DCFG_CCSR_RSTRQSR1 field descriptions

Field	Description
0 SRIO1_RR	Rapid I/O 1 reset request requires device level PORESET_B or HRESET_B. 0 Rapid I/O 1 reset request event not active. 1 Rapid I/O 1 reset request event active.
1 SRIO2_RR	Rapid I/O 2 reset request requires device level PORESET_B or HRESET_B. 0 Rapid I/O 2 reset request event not active. 1 Rapid I/O 2 reset request event active.
2 -	This field is reserved. Reserved
3 -	This field is reserved. Reserved
4–8 -	This field is reserved. Reserved
9 IFC_RR	Integrated Flash Controller reset request event This bit is set if an ECC error occured in NAND Flash preload for the RCW phase or the Preboot Initialization phase

Table continues on the next page...

DCFG_CCSR_RSTRQSR1 field descriptions (continued)

Field	Description
	NOTE: After a PORESET, RCWSRn registers can be read. If RSTRQSR[IFC_RR] is set after a PORESET and RCWSRn does not contain RCW values, then the failure occurred during RCW. If the bit is set and RCWSRn contains valid values, then the failure occurred during PBI. 0 IFC reset event reset request event not active 1 IFC reset event reset request event active
10 WDT_RR	Watchdog Timer reset requires device level PORESET_B or HRESET_B. 0 All bits in RSTRQWDTSRL and RSTRQWDTSRU are cleared 1 At least one bit in RSTRQWDTSRL or RSTRQWDTSRU is set
11 SW_RR	Software settable reset requested. (See Reset Control Register (DCFG_CCSR_RSTCR) .) 0 Software reset request event not active 1 Software reset request event active
12 CCM_RR	Coherency Fabric write miss on all local access windows during reset's PBL phase requires device level PORESET_B or HRESET_B. 0 Coherency Module write miss reset request event not active 1 Coherency Module write miss reset request event active
13 PBL_RR	PBL error reset request requires device level PORESET_B or HRESET_B. 0 PBL reset request event not active 1 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 0 Security Fuse Processor reset request event not active 1 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 . 0 Security Monitor reset request event not active 1 Security Monitor reset request event active
16 SDC_RR	Security Debug Controller reset request Security Debug Controller requires device level PORESET_B . 0 SDC reset request event not active 1 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 or HRESET_B. 0 Multi-bit ECC error reset request event not active 1 Multi-bit ECC error reset request event active
18 -	Reserved 0 POR BIST error reset request event not active 1 POR BIST error reset request event active

Table continues on the next page...

DCFG_CCSR_RSTRQSR1 field descriptions (continued)

Field	Description
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 or HRESET_B. 0 RCPM Time Out event reset request event not active 1 RCPM Time Out event reset request event active
20 SRDS_RST_RR	SerDes reset event. Occurs if any enabled SerDes PLL does not lock. 0 SerDes reset request event not active 1 SerDes reset request event active
21 -	Reserved
22–31 -	This field is reserved. Reserved

27.3.17 Reset Request WDT Mask Register (DCFG_CCSR_RSTRQWDTMR)

The RSTRQWDTMR contains mask bits for optional masking of RESET_REQ_B generation on thread watchdog timer expiration (WRC/WRS=10) from the individual threads. Excludes device sources (see Reset Request WDT Mask Register (DCFG_CCSR_RSTRQWDTMR)).

WDT field *n*	T2080 Thread
0	Core 0 Thread 0
1	Core 0 Thread 1
2	Core 1 Thread 0
3	Core 1 Thread 1
4	Core 2 Thread 0
5	Core 2 Thread 1
6	Core 3 Thread 0
7	Core 3 Thread 1

Address: 0h base + E_00D4h offset = E_00D4h

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	Reserved								WDT_MSK7	WDT_MSK6	WDT_MSK5	WDT_MSK4	WDT_MSK3	WDT_MSK2	WDT_MSK1	WDT_MSK0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DCFG_CCSR_RSTRQWDTMR field descriptions

Field	Description
0–23 -	This field is reserved. Reserved
24 WDT_MSK7	WDT Reset Request Mask. 0 Associated WDT can cause a reset request 1 Associated WDT cannot cause a reset request
25 WDT_MSK6	WDT Reset Request Mask. 0 Associated WDT can cause a reset request 1 Associated WDT cannot cause a reset request
26 WDT_MSK5	WDT Reset Request Mask. 0 Associated WDT can cause a reset request 1 Associated WDT cannot cause a reset request
27 WDT_MSK4	WDT Reset Request Mask. 0 Associated WDT can cause a reset request 1 Associated WDT cannot cause a reset request
28 WDT_MSK3	WDT Reset Request Mask. 0 Associated WDT can cause a reset request 1 Associated WDT cannot cause a reset request
29 WDT_MSK2	WDT Reset Request Mask. 0 Associated WDT can cause a reset request 1 Associated WDT cannot cause a reset request
30 WDT_MSK1	WDT Reset Request Mask. 0 Associated WDT can cause a reset request 1 Associated WDT cannot cause a reset request
31 WDT_MSK0	WDT Reset Request Mask. 0 Associated WDT can cause a reset request 1 Associated WDT cannot cause a reset request

27.3.18 Reset Request WDT Status Register (DCFG_CCSR_RSTRQWDTSR)

The RSTRQWDTSR contains status bits to record the reasons for RESET_REQ_B assertion due to a CPU watchdog timer when configured for WRC/WRS=10. Excludes device sources (see Reset Request Status Register (DCFG_CCSR_RSTRQSR1)).

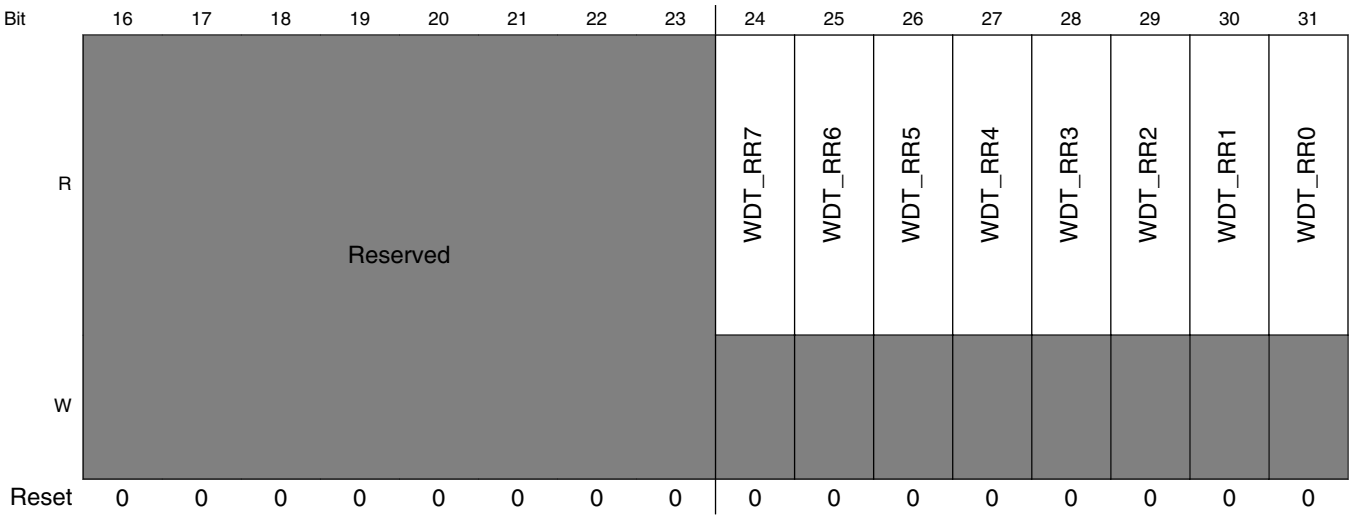
NOTE

The LSB, bit 31, is associated with Thread 0.

WDT field *n*	T2080 Thread
0	Core 0 Thread 0
1	Core 0 Thread 1
2	Core 1 Thread 0
3	Core 1 Thread 1
4	Core 2 Thread 0
5	Core 2 Thread 1
6	Core 3 Thread 0
7	Core 3 Thread 1

Address: 0h base + E_00DCh offset = E_00DCh

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



DCFG_CCSR_RSTRQWDTSR field descriptions

Field	Description
0–23 -	This field is reserved. Reserved
24 WDT_RR7	WDT Reset Request Status. 0 Associated WDT configured for WRC/WRS=10 has not caused a reset request 1 Associated WDT configured for WRC/WRS=10 has caused a reset request
25 WDT_RR6	WDT Reset Request Status. 0 Associated WDT configured for WRC/WRS=10 has not caused a reset request 1 Associated WDT configured for WRC/WRS=10 has caused a reset request
26 WDT_RR5	WDT Reset Request Status. 0 Associated WDT configured for WRC/WRS=10 has not caused a reset request 1 Associated WDT configured for WRC/WRS=10 has caused a reset request
27 WDT_RR4	WDT Reset Request Status. 0 Associated WDT configured for WRC/WRS=10 has not caused a reset request 1 Associated WDT configured for WRC/WRS=10 has caused a reset request
28 WDT_RR3	WDT Reset Request Status. 0 Associated WDT configured for WRC/WRS=10 has not caused a reset request 1 Associated WDT configured for WRC/WRS=10 has caused a reset request
29 WDT_RR2	WDT Reset Request Status. 0 Associated WDT configured for WRC/WRS=10 has not caused a reset request 1 Associated WDT configured for WRC/WRS=10 has caused a reset request
30 WDT_RR1	WDT Reset Request Status. 0 Associated WDT configured for WRC/WRS=10 has not caused a reset request 1 Associated WDT configured for WRC/WRS=10 has caused a reset request
31 WDT_RR0	WDT Reset Request Status.

Table continues on the next page...

DCFG_CCSR_RSTRQWDTSR field descriptions (continued)

Field	Description
	0 Associated WDT configured for WRC/WRS=10 has not caused a reset request 1 Associated WDT configured for WRC/WRS=10 has caused a reset request

27.3.19 Boot Release Register (DCFG_CCSR_BRR)

The BRR contains control bits for enabling boot for each core. On exiting HRESET or PORESET, the RCW BOOT_HO field optionally allows for logical core 0 to be released for booting or to remain in boot holdoff. All other cores remain in boot holdoff until their corresponding bit is set.

NOTE

The LSB, bit 31, is associated with Core 0.

NOTE

If a bit is changed from 1 to 0 outside of warm reset, results are boundedly undefined for that core.

Address: 0h base + E_00E4h offset = E_00E4h

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												CR3	CR2	CR1	CR0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DCFG_CCSR_BRR field descriptions

Field	Description
0–27 -	This field is reserved. Reserved
28 CR3	Core 3 Release. 0 Core is in Boot Holdoff and not released for Booting 1 Core released for Booting
29 CR2	Core 2 Release. 0 Core is in Boot Holdoff and not released for Booting 1 Core released for Booting
30 CR1	Core 1 Release. 0 Core is in Boot Holdoff and not released for Booting 1 Core released for Booting

Table continues on the next page...

DCFG_CCSR_BRR field descriptions (continued)

Field	Description
31 CR0	Core 0 Release. 0 Core is in Boot Holdoff and not released for Booting 1 Core released for Booting

27.3.20 Reset Control Word Status Register n (DCFG_CCSR_RCWSR*n*)

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.

NOTE

After a PORESET, RCWSRn registers can be read. If RSTRQSR[IFC_RR] is set after a PORESET and RCWSRn does not contain RCW values, then the failure occurred during RCW. If the bit is set and RCWSRn contains valid values, then the failure occurred during PBI.

Address: 0h base + E_0100h offset + (4d × i), where i=0d to 15d

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	RCW																															
W																																
Reset	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n

DCFG_CCSR_RCWSR*n* field descriptions

Field	Description
0–31 RCW	Read-only value of RCW bits (n-1)*32 : (n*32)-1 loaded in power-on reset's Reset Configuration stage.

27.3.21 Scratch Read / Write Register n (DCFG_CCSR_SCRATCHRW*n*)

The SCRATCHRWn provides read / write scratch register locations available to the user.

NOTE

When performing secure boot, these registers are defined as follows:

- SCRATCHRW1 - ESBC Pointer Register (Primary Boot Image)
- SCRATCHRW2 - Failure Code if Secure Boot Fails (Primary Boot Image)
- SCRATCHRW3 - ESBC Pointer Register (Alternate Boot Image)
- SCRATCHRW4 - Failure Code if Secure Boot Fails (Alternate Boot Image)

Address: 0h base + E_0200h offset + (4d × i), where i=0d to 3d

Bit	0–31
R W	VAL
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

DCFG_CCSR_SCRATCHRW*n* field descriptions

Field	Description
0–31 VAL	32-bit scratch contents

27.3.22 Scratch Read Register n (DCFG_CCSR_SCRATCHW1R*n*)

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.

Address: 0h base + E_0300h offset + (4d × i), where i=0d to 3d

Bit	0–31
R W	VAL
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

DCFG_CCSR_SCRATCHW1R*n* field descriptions

Field	Description
0–31 VAL	32-bit scratch contents

27.3.23 Core Reset Status Register n (DCFG_CCSR_CRSTSR)

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

A hard reset of the device causes the following to occur:

- RST_PORST remains unchanged (PORST resources are not affected by Hard Reset)
- RST_HRST is set
- All other bits are cleared

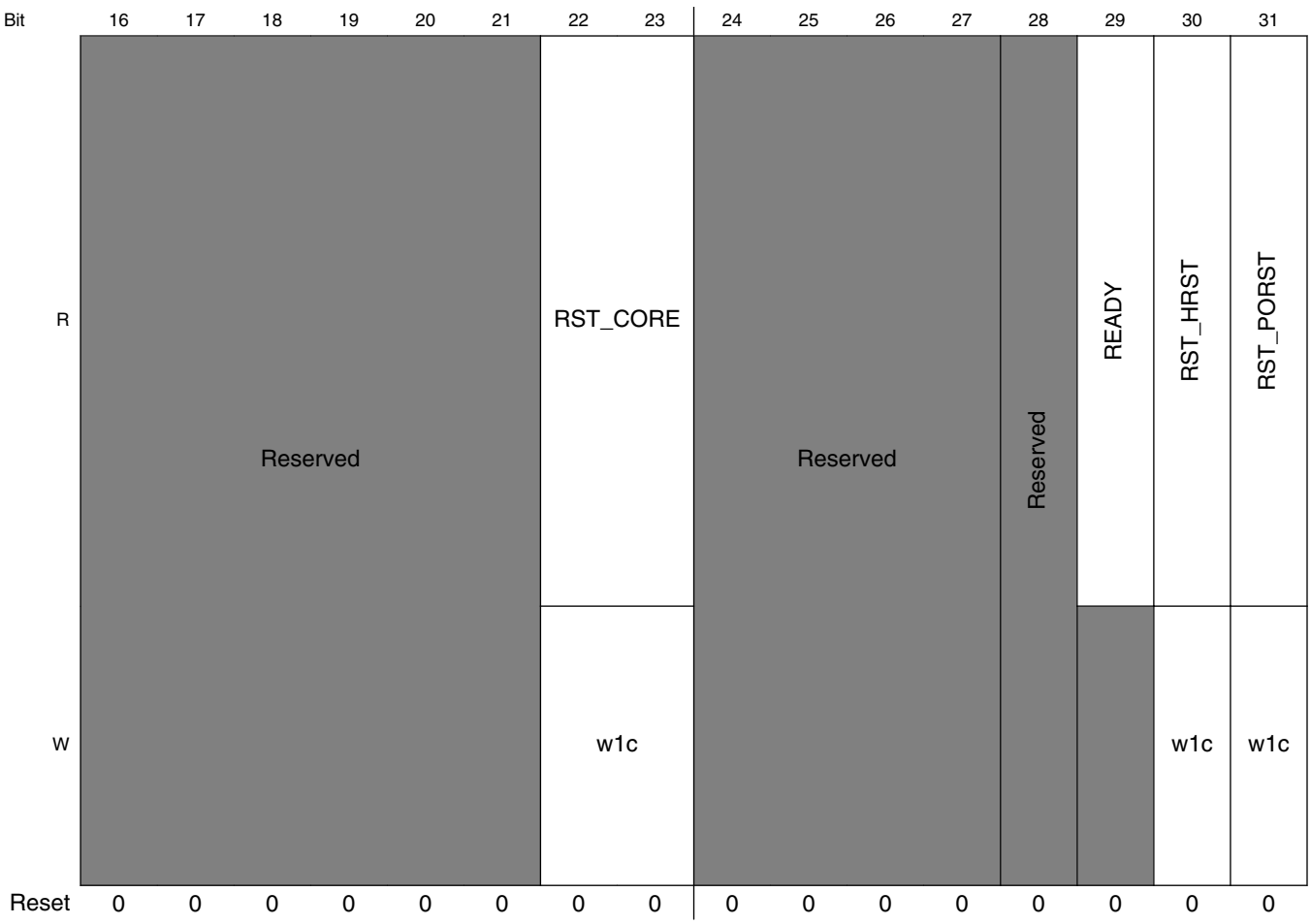
If an application prefers that RST_PORST is not set after hard reset, then a write-1-clear must be done to CRSTSRn[RST_PORST] upon exiting power-on reset.

Ready Bit Functionality

This bit is cleared on a device power-on or hard reset. Upon completion of power-on or hard reset processing, this bit may be automatically set for a core if none of the conditions specified in the READY bit definition are true.

Address: 0h base + E_0400h offset = E_0400h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	Reserved						RST_WRT		Reserved						RST_MPIC	
W							w1c								w1c	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



DCFG_CCSR_CRSTSR field descriptions

Field	Description
0–5 -	This field is reserved. Reserved
6–7 RST_WRT	Core was reset in response to watchdog timer expiration 0 No reset 1 Reset occurred and WRC/WRS=11
8–13 -	This field is reserved. Reserved
14–15 RST_MPIC	Core was reset in response to MPIC reset request 0 No reset 1 Reset occurred
16–21 -	This field is reserved. Reserved
22–23 RST_CORE	Core was reset in response to internal core request to reset itself by setting bit DBCR0[RST] register 0 No reset 1 Reset occurred

Table continues on the next page...

DCFG_CCSR_CRSTSR field descriptions (continued)

Field	Description
24–27 -	This field is reserved. Reserved
28 -	This field is reserved. 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 PH10 state • Core PH15 state • Core held in reset via a warm reset from the MPIC This bit reflects what is driven on the READY_P0 external signal. 0 Core 0 not ready 1 Core 0 ready
30 RST_HRST	Core was reset due to an HRESET (note a PORESET causes an HRESET, but a PORESET will not cause this bit to be set)
31 RST_PORST	Core was reset due to a PORESET

27.3.24 USB n Logical I/O Device Number register (DCFG_CCSR_USB*n*LIODNR)

The LIODNR provides the Logical I/O Device Number for initiator peripherals on the device which do not have a dedicated internal register for this.

Address: 0h base + E_0520h 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	Reserved																LDEVNUM															
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

DCFG_CCSR_USB*n*LIODNR field descriptions

Field	Description
0–15 -	This field is reserved. Reserved
16–31 LDEVNUM	Logical Device Number

27.3.25 SD/MMC Logical I/O Device Number register (DCFG_CCSR_SDMMCLIODNR)

The LIODNR provides the Logical I/O Device Number for initiator peripherals on the device which do not have a dedicated internal register for this.

Address: 0h base + E_0530h offset = E_0530h

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																LDEVNUM															
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

DCFG_CCSR_SDMMCLIODNR field descriptions

Field	Description
0–15 -	This field is reserved. Reserved
16–31 LDEVNUM	Logical Device Number

27.3.26 SATA n Logical I/O Device Number register (DCFG_CCSR_SATA*n*LIODNR)

The LIODNR provides the Logical I/O Device Number for initiator peripherals on the device which do not have a dedicated internal register for this.

Address: 0h base + E_0550h 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	Reserved																LDEVNUM															
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

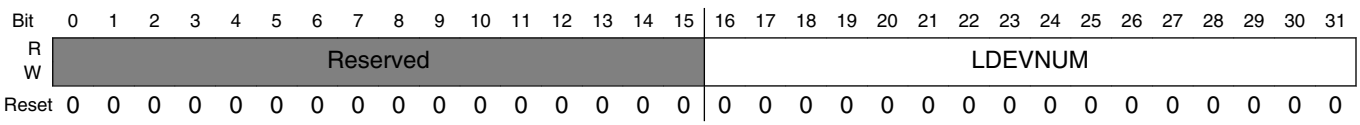
DCFG_CCSR_SATA*n*LIODNR field descriptions

Field	Description
0–15 -	This field is reserved. Reserved
16–31 LDEVNUM	Logical Device Number

27.3.27 DMA n Logical I/O Device Number register (DCFG_CCSR_DMA*n*LIODNR)

The LIODNR provides the Logical I/O Device Number for initiator peripherals on the device which do not have a dedicated internal register for this.

Address: 0h base + E_0580h offset + (4d × i), where i=0d to 2d



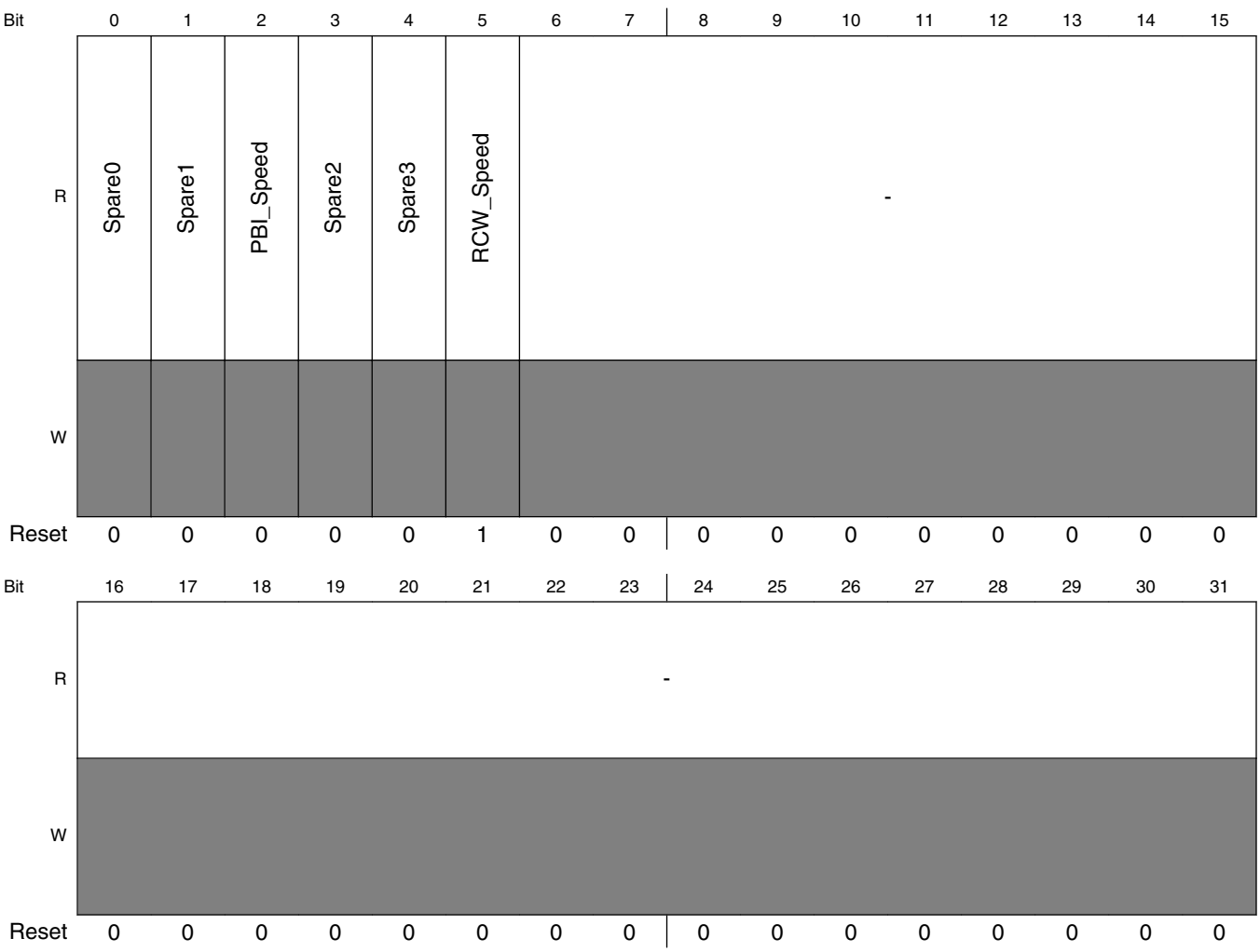
DCFG_CCSR_DMA*n*LIODNR field descriptions

Field	Description
0–15 -	This field is reserved. Reserved
16–31 LDEVNUM	Logical Device Number

27.3.28 Preboot Loader Status Register (DCFG_CCSR_PBLSR)

The PBLSR contains the status bits for PBL's PBI and RCW operating frequencies. This register is PORESET resetable and not HRESET resetable. This register upon PORESET loads in the same strapped values that PBLCR loads in.

Address: 0h base + E_0600h offset = E_0600h



DCFG_CCSR_PBLSR field descriptions

Field	Description
0 Spare0	-
1 Spare1	-

Table continues on the next page...

DCFG_CCSR_PBLSR field descriptions (continued)

Field	Description
2 PBI_Speed	PBL speed during the PBI Phase. 0 Low Speed 1 High Speed
3 Spare2	-
4 Spare3	-
5 RCW_Speed	PBL speed during the RCW Phase. 0 Low Speed 1 High Speed
6–31 -	Reserved

27.3.29 PAMU Bypass Enable Register (DCFG_CCSR_PAMUBYPENR)

The PAMU contains control bits for enabling bypass mode on each PAMU. On exiting RCW the register will be loaded with the inverse of secure_boot_en.

Address: 0h base + E_0604h offset = E_0604h

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	PBYP								Reserved								Reserved															
Reset	n*	n*	n*	n*	n*	n*	n*	n*	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:
- PBYP field: If secure boot is enabled (either by RCW[SB_EN] or the ITS fuse in the SFP block), these bits reset to 0; if secure boot is not enabled, these bits reset to 1.

DCFG_CCSR_PAMUBYPENR field descriptions

Field	Description
0–7 PBYP	PAMU *n* bypass, where *n*=1-8. 0 PAMU *n* is not bypassed 1 PAMU *n* is bypassed
8–15 -	This field is reserved. Reserved
16–31 -	This field is reserved. Reserved

27.3.30 DMA Control Register (DCFG_CCSR_DMACR1)

The DMACR1 contains bits for allowing DMA transactions from internal sources on the device.

Address: 0h base + E_0608h offset = E_0608h

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R W	DMA1_0		DMA1_1		DMA1_2		DMA1_3		DMA2_0		DMA2_1		DMA2_2		DMA2_3	
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	DMA1_4		DMA1_5		DMA1_6		DMA1_7		DMA2_4		DMA2_5		DMA2_6		DMA2_7	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DCFG_CCSR_DMACR1 field descriptions

Field	Description
0–1 DMA1_0	DMA 1, Channel 0. 00 DMA's Channel may be initiated by SoC's DMA pins 01 DMA's Channel may be initiated by Peripheral #1 10 DMA's Channel may be initiated by Peripheral #2 11 DMA's Channel may be initiated by the EPU: • Even DMA channels are initiated by the EPU's "trigger DMA request 0" action (EPACRn register) • Odd DMA channels are initiated by the EPU's "trigger DMA request 1" action (EPACRn register)
2–3 DMA1_1	DMA 1, Channel 1, following bit definition for DMA1_0.
4–5 DMA1_2	DMA 1, Channel 2, following bit definition for DMA1_0.
6–7 DMA1_3	DMA 1, Channel 3, following bit definition for DMA1_0.
8–9 DMA2_0	DMA 2, Channel 0, following bit definition for DMA1_0.
10–11 DMA2_1	DMA 2, Channel 1, following bit definition for DMA1_0.
12–13 DMA2_2	DMA 2, Channel 2, following bit definition for DMA1_0.
14–15 DMA2_3	DMA 2, Channel 3, following bit definition for DMA1_0.
16–17 DMA1_4	DMA 1, Channel 4, following bit definition for DMA1_0.
18–19 DMA1_5	DMA 1, Channel 5, following bit definition for DMA1_0.

Table continues on the next page...

DCFG_CCSR_DMACR1 field descriptions (continued)

Field	Description
20–21 DMA1_6	DMA 1, Channel 6, following bit definition for DMA1_0.
22–23 DMA1_7	DMA 1, Channel 7, following bit definition for DMA1_0.
24–25 DMA2_4	DMA 2, Channel 4, following bit definition for DMA1_0.
26–27 DMA2_5	DMA 2, Channel 5, following bit definition for DMA1_0.
28–29 DMA2_6	DMA 2, Channel 6, following bit definition for DMA1_0.
30–31 DMA2_7	DMA 2, Channel 7, following bit definition for DMA1_0.

27.3.31 DCSR Control Register (DCFG_CCSR_DCSRCR)

The DCSR controls parameters which affect accesses to the DCSR address space.

Address: 0h base + E_0704h offset = E_0704h

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														DCSR_SZ	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

DCFG_CCSR_DCSRCR field descriptions

Field	Description
0–29 -	This field is reserved. Reserved
30–31 DCSR_SZ	DCSR Address Space Size. This bit specifies the usable size of the DCSR space and the corresponding alignment required when setting up a Local Access Window. **NOTE:** This bit only affects accesses performed through CoreNet and does not affect accesses performed through the bypass path on the MISRVD bus. **NOTE:** When ever this bit is changed, it must be immediately followed with a read of this register where the contents of this read are not guaranteed to contain the value of this register. Similarly after modifying this bit and reading back the register, the appropriate LAW should be modified for the new base address and size.

Table continues on the next page...

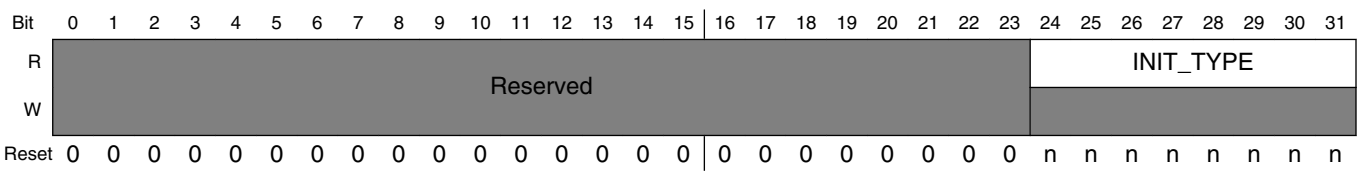
DCFG_CCSR_DCSRCR field descriptions (continued)

Field	Description
	00 The size of the DCSR address space configured as 4 MB (normal use) so that DCSR is allocated on a 4 MB boundary 01 Reserved 10 Reserved 11 The size of the DCSR address space configured as 1 GB (lab use) so that DCSR is allocated on a 1 GB boundary

27.3.32 Topology Initiator Type n Register (DCFG_CCSR_TP_ITYP*n*)

Each Initiator Type Topology Register provides one entry of a 64 entry lookup table.

Address: 0h base + E_0740h offset + (4d × i), where i=0d to 63d



DCFG_CCSR_TP_ITYP*n* field descriptions

Field	Description
0–23 -	This field is reserved. 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. 00h Simple initiatior 02h e6500 (disabled) 03h e6500 (enabled)

27.3.33 Core Cluster n Topology Register (DCFG_CCSR_TP_CLUSTER*n*)

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.

Address: 0h base + E_0844h offset + (8d × i), where i=0d to 0d

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	EOC		IT_IDX_PC4						Reserved		IT_IDX_PC3					
W																
Reset	n	n	n	n	n	n	n	n	0	0	n	n	n	n	n	n

Bit	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	Reserved		IT_IDX_PC2						Reserved		IT_IDX_PC1					
W																
Reset	0	0	n	n	n	n	n	n	0	0	n	n	n	n	n	n

DCFG_CCSR_TP_CLUSTER*n* field descriptions

Field	Description
0–1 EOC	End of Clusters - if the EOC field is non-zero, the register contains the information on the last cluster in the SoC. 2'b00 - not the last cluster 2'b01,10,11 - the last cluster in the SoC
2–7 IT_IDX_PC4	Initiator Type Index for this cluster's fourth 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.
8–9 -	This field is reserved. 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 -	This field is reserved. reserved
18–23 IT_IDX_PC2	Initiator Type Index for this cluster secind 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 -	This field is reserved. 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.

27.3.34 QMBM Warm Reset Control Register (DCFG_CCSR_QMBM_WARMRST)

Software writes bit location to initiate warm reset to specific device.

Software polls bit location to know when warm reset has completed.

NOTE

Before initiating warm reset software must:

Shut down software portal interfaces (stop de-queueing new work from QMAN and stops enqueueing new work to QMAN).

Quiesce FMAN

NOTE

After setting the warm reset bits software must:

Poll QMBM_WARMRST to know when warm reset has completed.

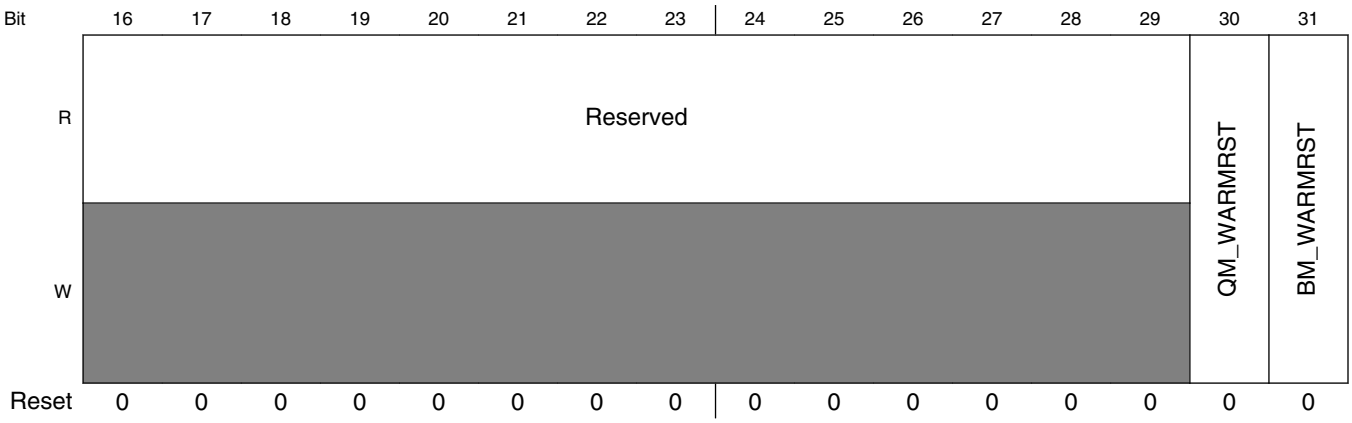
Re-initialize QMAN/BMAN. This may include clearing out any history it has of QMAN portal ring state as well as forgetting about buffers it may not have previously released.

Restart FMAN.

For more details please refer the DPAA Appendices.

Address: 0h base + E_0A00h offset = E_0A00h

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



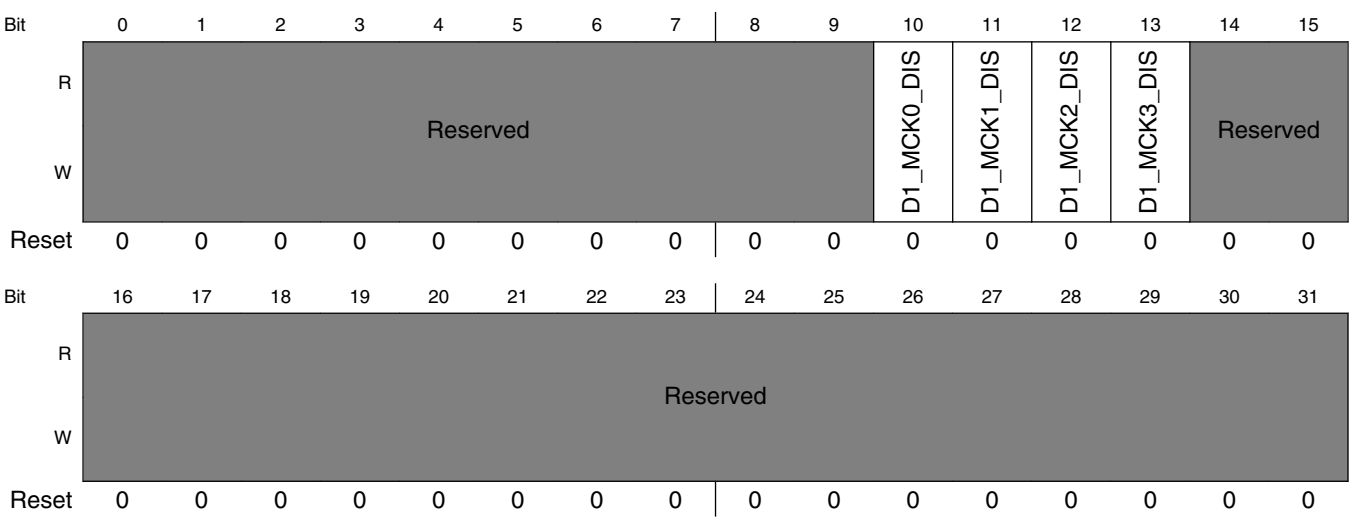
DCFG_CCSR_QMBM_WARMRST field descriptions

Field	Description
0–29 Reserved	This field is reserved.
30 QM_WARMRST	Write 1 to inititiate QMan warm reset. Write 0 has no effect. Read of 1 indicates QMan warm reset in progress.
31 BM_WARMRST	Write 1 to inititiate BMan warm reset. Write 0 has no effect. Read of 1 indicates BMan warm reset in progress.

27.3.35 DDR Clock Disable Register (DCFG_CCSR_DDRCLKDR)

DDRCLKDR allows for specific, unused clocks of both of the DDR Controllers' interfaces to be released to high impedance, thereby reducing power consumption.

Address: 0h base + E_0E60h offset = E_0E60h



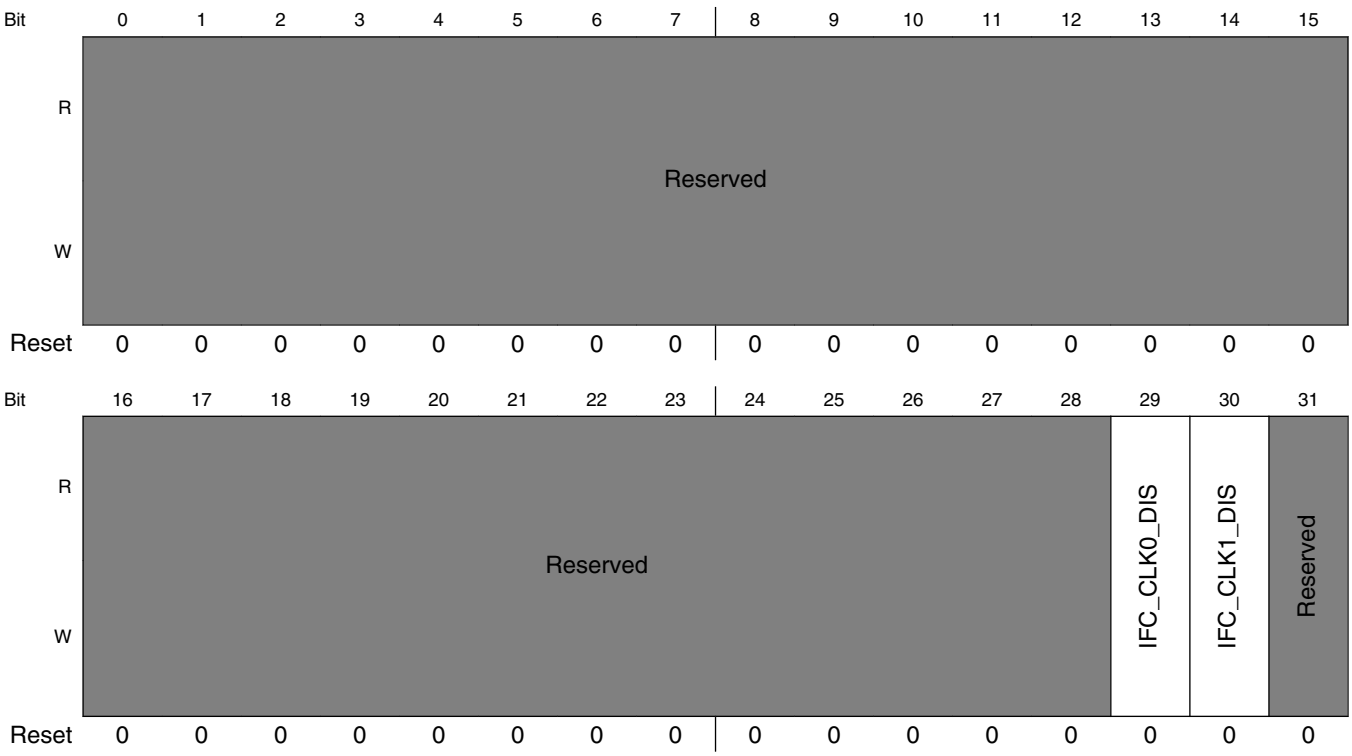
DCFG_CCSR_DDRCLKDR field descriptions

Field	Description
0–9 -	This field is reserved. Reserved
10 D1_MCK0_DIS	DDR Controller 1 clock 0 disable. The output is tri-stated, when disabled. 0 D1_MCK[0] is enabled 1 D1_MCK[0] is disabled
11 D1_MCK1_DIS	DDR Controller 1 clock 1 disable. The output is tri-stated, when disabled. 0 D1_MCK[1] is enabled 1 D1_MCK[1] is disabled
12 D1_MCK2_DIS	DDR Controller 1 clock 2 disable. The output is tri-stated, when disabled. 0 D1_MCK[2] is enabled 1 D1_MCK[2] is disabled
13 D1_MCK3_DIS	DDR Controller 1 clock 3 disable. The output is tri-stated, when disabled. 0 D1_MCK[3] is enabled 1 D1_MCK[3] is disabled
14–31 -	This field is reserved. Reserved

27.3.36 IFC Clock Disable Register (DCFG_CCSR_IFCCLKDR)

The IGCCLKDR allows for specific, unused clocks of the IFC interface to be not driven/ high-impedance, thereby reducing power consumption.

Address: 0h base + E_0E68h offset = E_0E68h



DCFG_CCSR_IFCCLKDR field descriptions

Field	Description
0–28 -	This field is reserved. Reserved
29 IFC_CLK0_DIS	IFC clock 0 disable. The output is not driven and in a high impedance state, when disabled. 0 IFC_CLK0 is enabled 1 IFC_CLK0 is disabled
30 IFC_CLK1_DIS	IFC clock 1 disable. The output is not driven and in a high impedance state, when disabled. 0 IFC_CLK1 is enabled 1 IFC_CLK1 is disabled
31 -	This field is reserved. Reserved

Chapter 28
General Purpose I/O (GPIO)

28.1 GPIO overview

This chapter describes the general-purpose I/O module, including signal descriptions, register settings and interrupt capabilities.

This figure shows the block diagram of the GPIO.

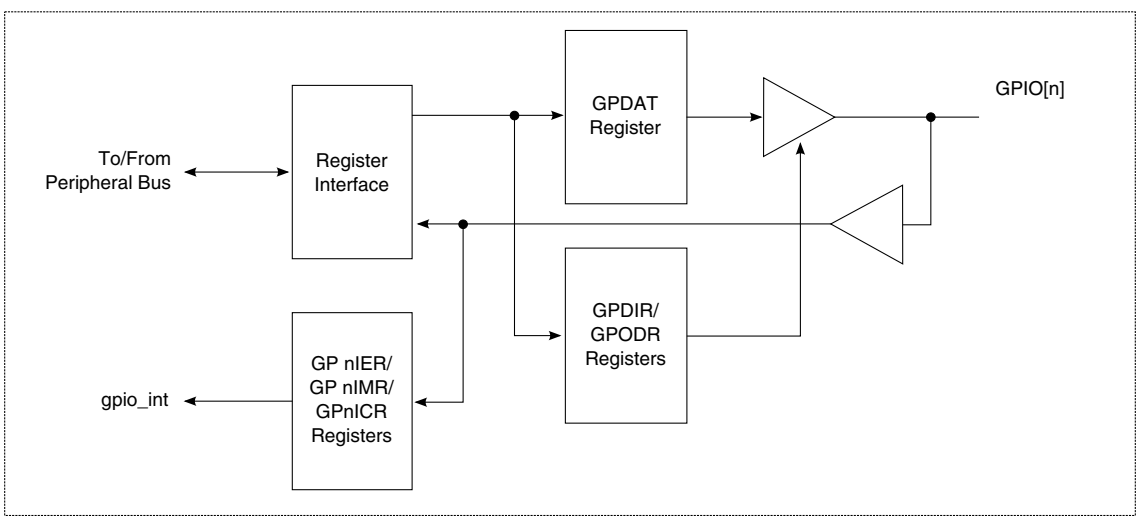


Figure 28-1. GPIO module block diagram

In general, GPIO module supports general-purpose I/O ports. Each port can be configured as an input or as an output. 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.

28.2 GPIO features summary

The GPIO unit includes the following features:

- Open-drain capability on all ports
- All ports can optionally generate an interrupt upon changing their state

28.3 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 28-1. GPIO external signals-detailed signal descriptions

Signal	I/O	Description	
GPIO1[9:31], GPIO2[0:15], GPIO2[25:31], GPIO3[0:31], GPIO4[0:9], GPIO4[24:25], GPIO4[27:31]	I/O	General Purpose I/O. Each pin can be individually set to act as input or output, according to application needs.	
		State Meaning	Asserted/Negated-Defined per application.
		Timing	Assertion/Negation-Inputs can be asserted completely asynchronously. Outputs are asynchronous to any externally visible clock.

28.4 GPIO Memory Map/Register Definition

The programmable register map for GPIO module occupies 24 bytes of memory-mapped space. The full register address is comprised of the CCSR window base address, specified in CCSRBAR, plus the functional block base address, plus the specific register's offset within that block. 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.

GPIO memory map

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
13_0000	GPIO direction register (GPIO1_GPDIR)	32	R/W	0000_0000h	28.4.1/1599
13_0004	GPIO open drain register (GPIO1_GPODR)	32	R/W	0000_0000h	28.4.2/1600
13_0008	GPIO data register (GPIO1_GPDAT)	32	R/W	0000_0000h	28.4.3/1600
13_000C	GPIO interrupt event register (GPIO1_GPIER)	32	w1c	0000_0000h	28.4.4/1601
13_0010	GPIO interrupt mask register (GPIO1_GPIMR)	32	R/W	0000_0000h	28.4.5/1601
13_0014	GPIO interrupt control register (GPIO1_GPICR)	32	R/W	0000_0000h	28.4.6/1602

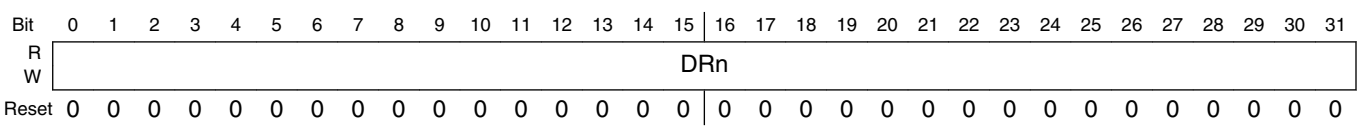
GPIO memory map (continued)

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
13_1000	GPIO direction register (GPIO2_GPDIR)	32	R/W	0000_0000h	28.4.1/1599
13_1004	GPIO open drain register (GPIO2_GPODR)	32	R/W	0000_0000h	28.4.2/1600
13_1008	GPIO data register (GPIO2_GPDAT)	32	R/W	0000_0000h	28.4.3/1600
13_100C	GPIO interrupt event register (GPIO2_GPIER)	32	w1c	0000_0000h	28.4.4/1601
13_1010	GPIO interrupt mask register (GPIO2_GPIMR)	32	R/W	0000_0000h	28.4.5/1601
13_1014	GPIO interrupt control register (GPIO2_GPICR)	32	R/W	0000_0000h	28.4.6/1602
13_2000	GPIO direction register (GPIO3_GPDIR)	32	R/W	0000_0000h	28.4.1/1599
13_2004	GPIO open drain register (GPIO3_GPODR)	32	R/W	0000_0000h	28.4.2/1600
13_2008	GPIO data register (GPIO3_GPDAT)	32	R/W	0000_0000h	28.4.3/1600
13_200C	GPIO interrupt event register (GPIO3_GPIER)	32	w1c	0000_0000h	28.4.4/1601
13_2010	GPIO interrupt mask register (GPIO3_GPIMR)	32	R/W	0000_0000h	28.4.5/1601
13_2014	GPIO interrupt control register (GPIO3_GPICR)	32	R/W	0000_0000h	28.4.6/1602
13_3000	GPIO direction register (GPIO4_GPDIR)	32	R/W	0000_0000h	28.4.1/1599
13_3004	GPIO open drain register (GPIO4_GPODR)	32	R/W	0000_0000h	28.4.2/1600
13_3008	GPIO data register (GPIO4_GPDAT)	32	R/W	0000_0000h	28.4.3/1600
13_300C	GPIO interrupt event register (GPIO4_GPIER)	32	w1c	0000_0000h	28.4.4/1601
13_3010	GPIO interrupt mask register (GPIO4_GPIMR)	32	R/W	0000_0000h	28.4.5/1601
13_3014	GPIO interrupt control register (GPIO4_GPICR)	32	R/W	0000_0000h	28.4.6/1602

28.4.1 GPIO direction register (GPIOx_GPDIR)

The GPIO direction register (GPDIR) defines the direction of the individual ports.

Address: Base address + 0h offset



GPIOx_GPDIR field descriptions

Field	Description
0–31 DRn	Direction. Indicates whether a pin is used as an input or an output. 0 The corresponding pin is an input. 1 The corresponding pin is an output.

28.4.2 GPIO open drain register (GPIOx_GPODR)

The GPIO open drain register (GPODR) defines the way individual ports drive their output.

Address: Base address + 4h 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	ODn																															
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

GPIOx_GPODR field descriptions

Field	Description
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. 0 The corresponding signal is actively driven as an output. 1 The corresponding signal is an open-drain driver. As an output, the signal is driven active-low, otherwise it is not driven (high impedance).

28.4.3 GPIO data register (GPIOx_GPDAT)

The GPIO data register (GPDAT) carries the data in/out for the individual ports.

Address: Base address + 8h 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	Dn																															
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

GPIOx_GPDAT field descriptions

Field	Description
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. Read operations always return the data at the pin except for output only pins that always return zero.

28.4.4 GPIO interrupt event register (GPIOx_GPIER)

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.

Address: Base address + Ch 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	EVn																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPIOx_GPIER field descriptions

Field	Description
0–31 EVn	Interrupt events. Indicates whether an interrupt event occurred on the corresponding GPIO signal. 0 No interrupt event occurred on the corresponding GPIO signal. 1 An interrupt event occurred on the corresponding GPIO signal.

28.4.5 GPIO interrupt mask register (GPIOx_GPIMR)

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.

Address: Base address + 10h 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	IMn																															
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

GPIOx_GPIMR field descriptions

Field	Description
0–31 IMn	Interrupt mask. Indicates whether an interrupt event is masked or not masked for the corresponding GPIO signal. 0 The input interrupt signal is masked (disabled). 1 The input interrupt signal is not masked (enabled).

28.4.6 GPIO interrupt control register (GPIOx_GPICR)

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.

Address: Base address + 14h 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	EDn																															
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

GPIOx_GPICR field descriptions

Field	Description
0–31 EDn	Edge detection mode. The corresponding port line asserts an interrupt request according to the following: 0 Any change on the state of the port generates an interrupt request. 1 High-to-low change on the port generates an interrupt request.

Chapter 29
Thermal Monitoring Unit (TMU)

29.1 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.

29.1.1 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.

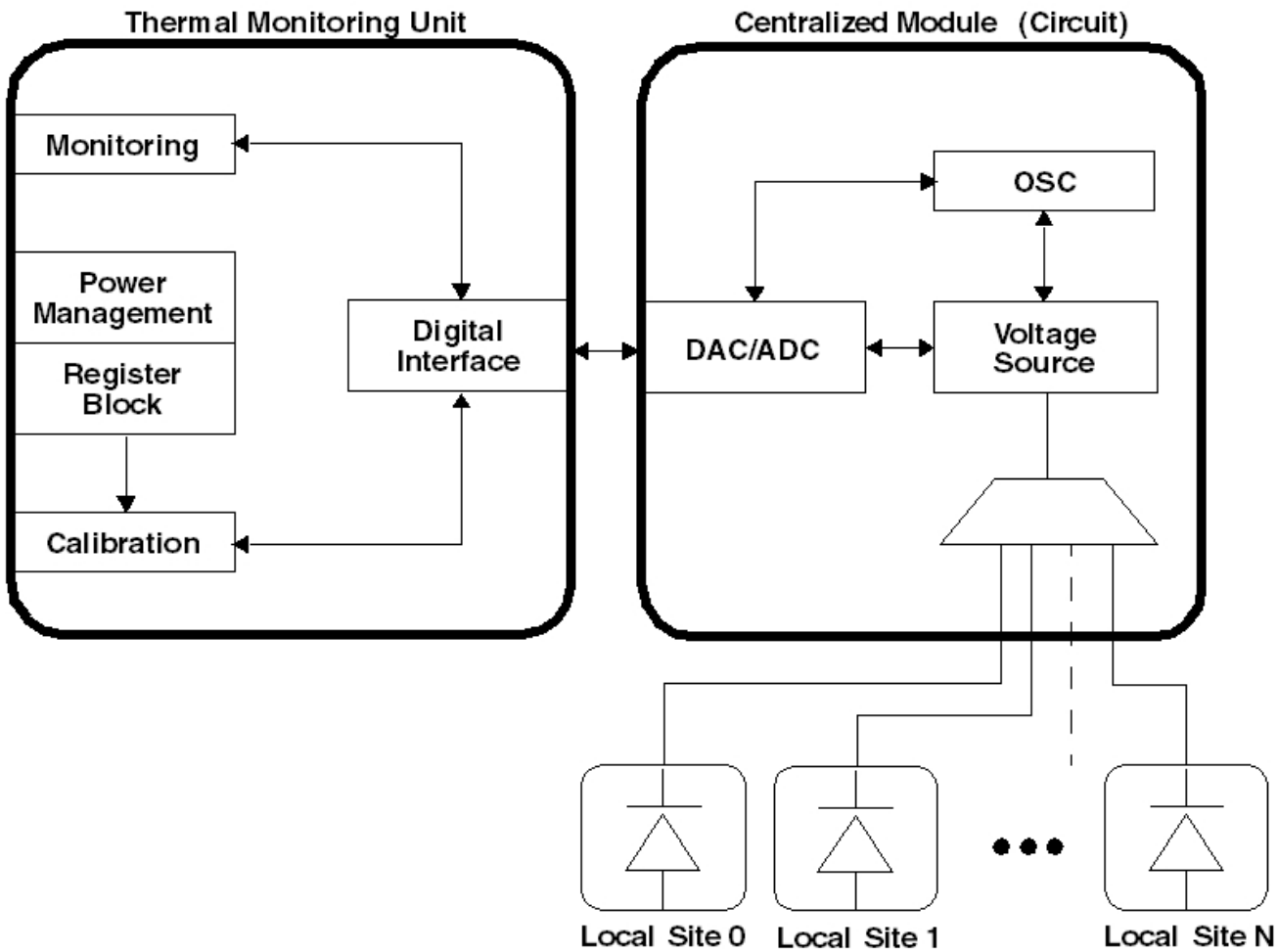


Figure 29-1. Thermal Monitoring Unit Block Diagram

29.1.2 Features

The temperature management unit features:

- Temperature measurement range 0-125 degrees Celsius
- 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

29.1.3 Modes of Operation

The TMU has one mode of the 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 29-1. TMU Mode Bit Setting

Modes with Features	TMR[ME]
Monitoring mode disabled	0
Monitoring mode enabled	1

29.1.4 Local Temperature Sensor Placement

The table below shows the placement of the local temperature sensor:

Table 29-2. Local Temperature Sensor Placement in SoC

Temperature Sensor ID	Placement
0	DDR
1	Cores cluster
2	Network SerDes Complex
3-15	Reserved

29.2 TMU Memory Map/Register Definition

This section provides a detailed description of all accessible TMU memory and registers. The table below lists the TMU registers. Note that the full register address is comprised of the programmable CCSRBAR together with the offset listed.

TMU memory map

Offset address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
F_0000	TMU mode register (TMU_TMR)	32	R/W	0000_0000h	29.2.1/1607
F_0004	TMU status register (TMU_TSR)	32	R	0000_0000h	29.2.2/1608
F_0008	TMU monitor temperature measurement interval register (TMU_TMTMIR)	32	R/W	0000_0000h	29.2.3/1610
F_0020	TMU interrupt enable register (TMU_TIER)	32	R/W	0000_0000h	29.2.4/1611
F_0024	TMU interrupt detect register (TMU_TIDR)	32	w1c	0000_0000h	29.2.5/1612
F_0028	TMU interrupt site capture register (TMU_TISCR)	32	R/W	0000_0000h	29.2.6/1613
F_002C	TMU interrupt critical site capture register (TMU_TICSCR)	32	R/W	0000_0000h	29.2.7/1613
F_0040	TMU monitor high temperature capture register (TMU_TMHTCRH)	32	R	0000_0000h	29.2.8/1614
F_0044	TMU monitor low temperature capture register (TMU_TMHTCRL)	32	R	0000_0000h	29.2.9/1615
F_0050	TMU monitor high temperature immediate threshold register (TMU_TMHTITR)	32	R/W	0000_0000h	29.2.10/ 1616
F_0054	TMU monitor high temperature average threshold register (TMU_TMHTATR)	32	R/W	0000_0000h	29.2.11/ 1617
F_0058	TMU monitor high temperature average critical threshold register (TMU_TMHTACTR)	32	R/W	0000_0000h	29.2.12/ 1618
F_0080	TMU temperature configuration register (TMU_TTCFGR)	32	R/W	0000_0000h	29.2.13/ 1618
F_0084	TMU sensor configuration register (TMU_TSCFGR)	32	R/W	0000_0000h	29.2.14/ 1619
F_0100	TMU report immediate temperature site register n (TMU_TRITSR)	32	R	0000_0000h	29.2.15/ 1620
F_0104	TMU report average temperature site register n (TMU_TRATSR)	32	R	0000_0000h	29.2.16/ 1621
F_0BF8	IP block revision register 0 (TMU_IPBRR0)	32	R	0190_0100h	29.2.17/ 1621
F_0BFC	IP block revision register 1 (TMU_IPBRR1)	32	R	0000_00F0h	29.2.18/ 1622

29.2.1 TMU mode register (TMU_TMR)

The TMU mode register allows software to control the operation of the thermal monitoring.

Address: F_0000h base + 0h offset = F_0000h

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

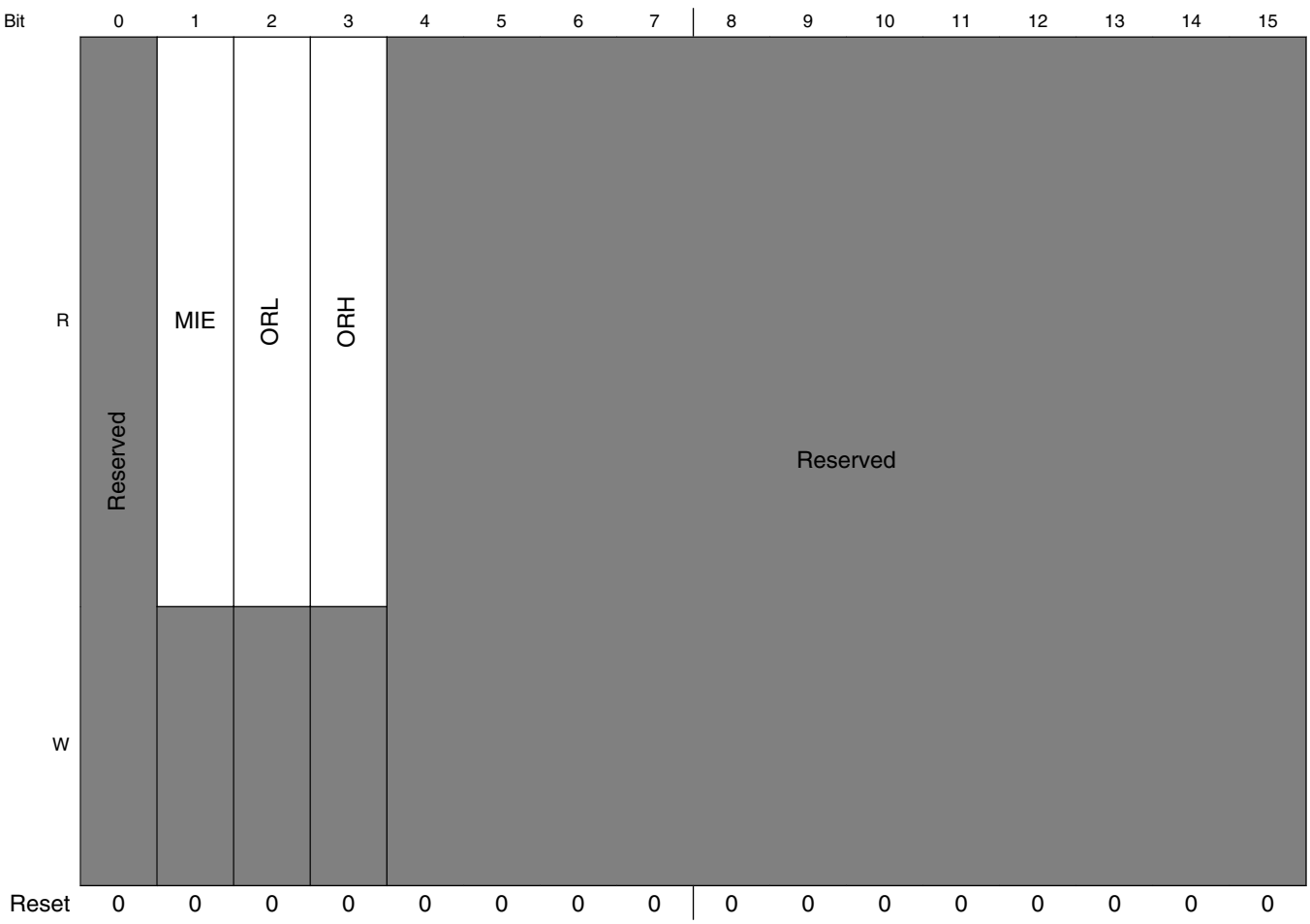
TMU_TMR field descriptions

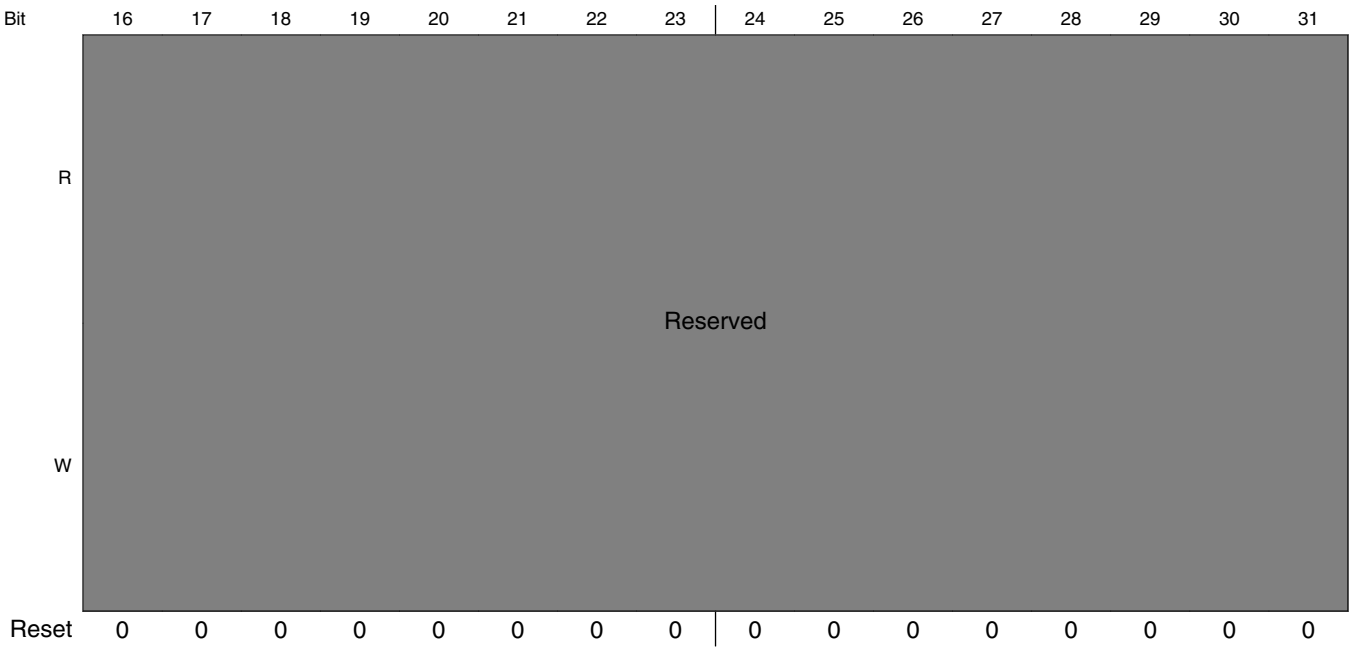
Field	Description
0 ME	Monitor mode enable. Before enabling the TMU for monitoring, the calibration table must be initialized. Failure to properly initialize the calibration table may result in boundedly undefined behavior. 0 No monitoring. Power saving mode. 1 Monitoring of sites as defined by MSITE.
1–3 -	This field is reserved. Reserved
4–5 ALPF	Average low pass filter setting. The average temperature is calculated as: ALPF x Current_Temp + (1 - ALPF) x Average_Temp. For proper operation, this field should only change when monitoring is disabled. 00 1.0 01 0.5 10 0.25 11 0.125
6–15 -	This field is reserved. Reserved
16–31 MSITE	Monitoring site select. 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. Bit - Site [0] - 0 DDR [1] - 1 Cores cluster [2] - 2 SerDes complex

29.2.2 TMU status register (TMU_TSR)

The TMU status register reports the monitoring status during operation.

Address: F_0000h base + 4h offset = F_0004h





TMU_TSR field descriptions

Field	Description
0 -	This field is reserved. Reserved
1 MIE	Monitoring interval exceeded. This bit will clear automatically when TMU monitoring is (re-)enabled or the monitoring interval register, TMTMIR, is written. 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.
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 125 degrees Celsius. This bit will clear automatically when TMU monitoring is (re-)enabled.
4–31 -	This field is reserved. Reserved

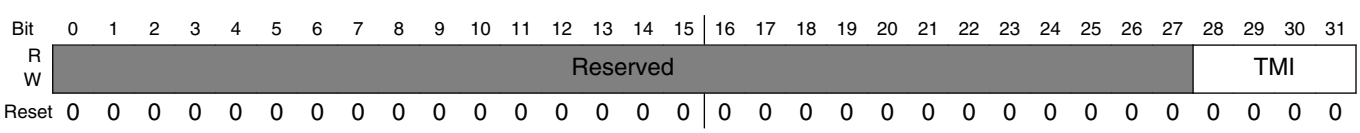
29.2.3 TMU monitor temperature measurement interval register (TMU_TMTMIR)

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.

Table 29-6. Platform Clock Frequency

TMI Bit Field Setting	Platform Clock Frequency	
	400MHz	667MHz
0000	0.02 s	0.015 s
0001	0.04 s	0.03 s
0010	0.08 s	0.05 s
0011	0.17 s	0.10 s
0100	0.34 s	0.20 s
0101	0.67 s	0.40 s
0110	1.34 s	0.80 s
0111	2.7 s	1.6 s
1000	5.4 s	3.2 s
1001	10.7 s	6.4 s
1010	21.5 s	12.9 s
1011	42.9 s	25.8 s
1100	85.9 s	51.5 s
1101	171.8 s	103.0 s
1110	343.6 s	206.1 s
1111	Disabled	

Address: F_0000h base + 8h offset = F_0008h



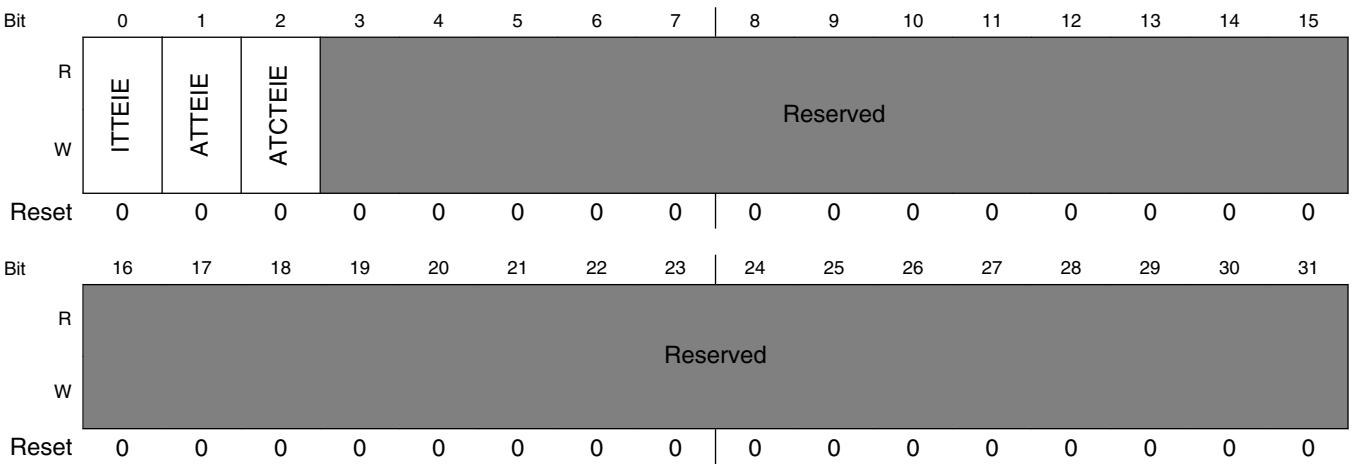
TMU_TMTMIR field descriptions

Field	Description
0–27 -	This field is reserved. Reserved
28–31 TMI	Temperature monitoring interval in seconds. For proper operation, this field should only change when monitoring is disabled, TMR[ME]=0. See table above for bit field settings.

29.2.4 TMU interrupt enable register (TMU_TIER)

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.

Address: F_0000h base + 20h offset = F_0020h



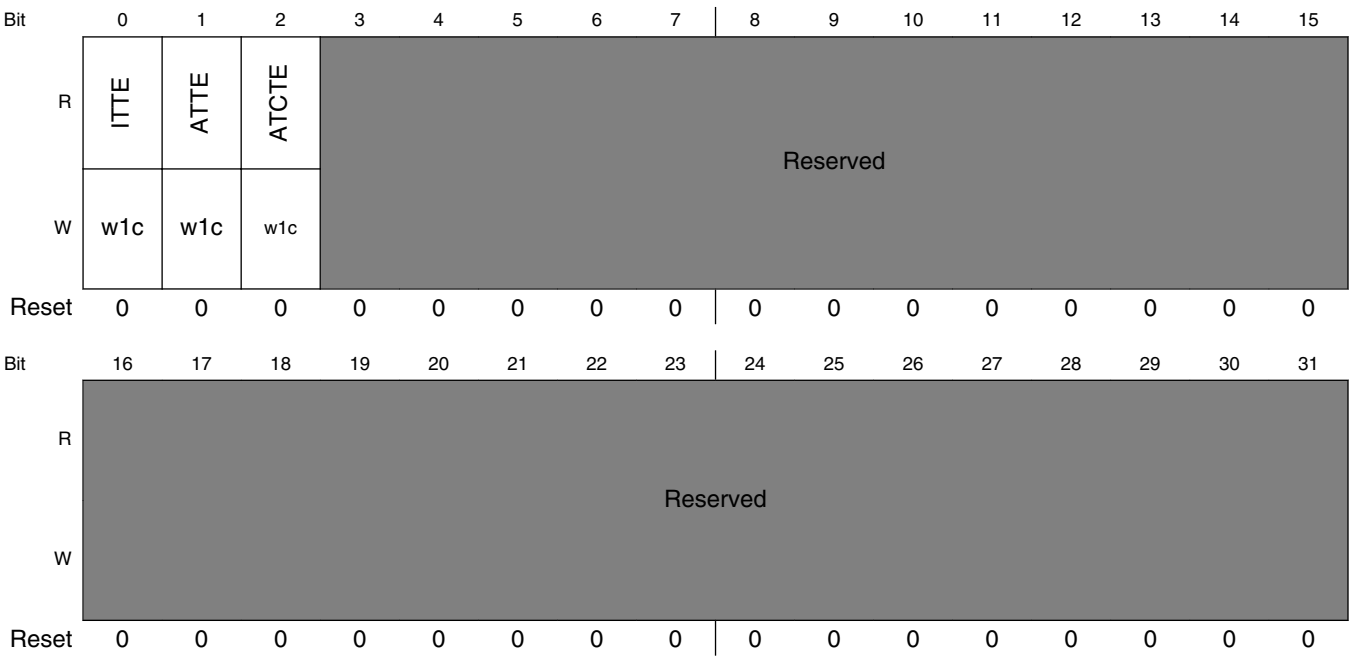
TMU_TIER field descriptions

Field	Description
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 -	This field is reserved. Reserved

29.2.5 TMU interrupt detect register (TMU_TIDR)

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.

Address: F_0000h base + 24h offset = F_0024h



TMU_TIDR field descriptions

Field	Description
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 125C. The sites which has exceeded the threshold are captured in TICR[TE_SITE(0:15)].
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 TICR[SITE(0:15)].
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 TICR[CSITE(0:15)].
3–31 -	This field is reserved. Reserved

29.2.6 TMU interrupt site capture register (TMU_TISCR)

The TMU interrupt site capture register holds information about the temperature sensor site associated with a detected interrupt event.

.

Address: F_0000h base + 28h offset = F_0028h

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	ISITE																ASITE															
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

TMU_TISCR field descriptions

Field	Description
0–15 ISITE	Temperature sensor site associated with the setting of TIDR[ITTE]. This field has the same bit representation as TMR[MSITE(0:15)]. Software should clear this field after handling the detected interrupt events ITTE.
16–31 ASITE	Temperature sensor site associated with the setting of TIDR[ATTE] . This field has the same bit representation as TMR[MSITE(0:15)]. Software should clear this field after handling the detected interrupt event ATTE.

29.2.7 TMU interrupt critical site capture register (TMU_TICSCR)

The TMU interrupt critical site capture register holds information about the temperature sensor site associated with a detected critical interrupt event.

Address: F_0000h base + 2Ch offset = F_002Ch

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																CASITE															
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

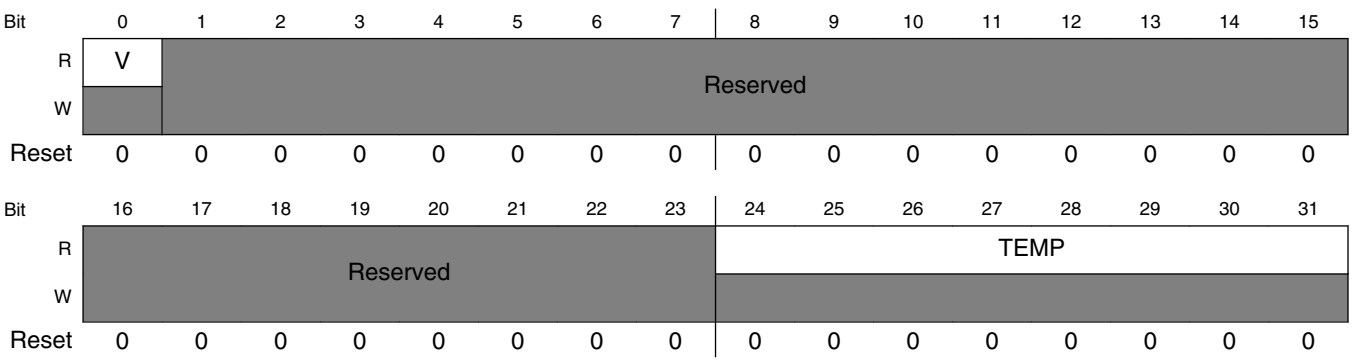
TMU_TICSCR field descriptions

Field	Description
0–15 -	This field is reserved. Reserved
16–31 CASITE	Temperature sensor site associated with the setting of TIDR[ATCTE] . This field has the same bit representation as TMR[MSITE(0:15)]. Software should clear this field after handling the detected critical interrupt event ATCTE.

29.2.8 TMU monitor high temperature capture register (TMU_TMHTCRH)

This TMU monitor register captures and records the highest temperature reached for any enabled monitored site within the temperature sensor range.

Address: F_0000h base + 40h offset = F_0040h



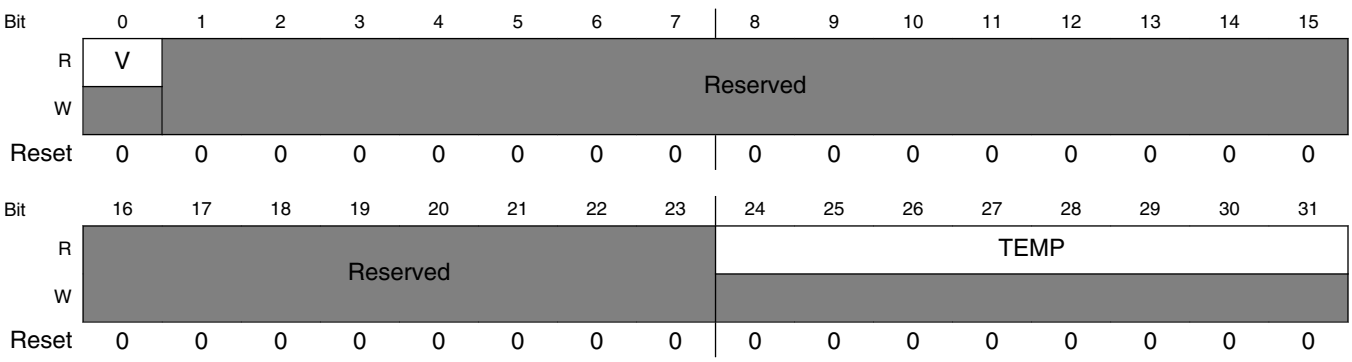
TMU_TMHTCRH field descriptions

Field	Description
0 V	Valid reading. (Re-)enabling the TMU will automatically clear this bit and start a new search. 0 Temperature reading is not valid due to no measured temperature within the sensor range of 0-125C for an enabled monitored site. 1 Temperature reading is valid.
1–23 -	This field is reserved. Reserved
24–31 TEMP	Highest temperature recorded in degrees Celsius by any enabled monitored site. Valid when V=1. 0-125C Sensor range 126-255C Reserved

29.2.9 TMU monitor low temperature capture register (TMU_TMHTCRL)

This TMU monitor register captures and record the lowest temperature reached for any one enabled monitored site within temperature sensor range.

Address: F_0000h base + 44h offset = F_0044h



TMU_TMHTCRL field descriptions

Field	Description
0 V	Valid reading. (Re-)enabling the TMU will automatically clear this bit and start a new search. 0 Temperature reading is not valid due to no measured temperature within the sensor range of 0-125C for an enabled monitored site. 1 Temperature reading is valid.
1–23 -	This field is reserved. Reserved
24–31 TEMP	Lowest temperature recorded in degrees Celsius by any enabled monitored site. Valid when V=1. (Note that these values are in decimal.) 0-125C Sensor range 126-255C Reserved

29.2.10 TMU monitor high temperature immediate threshold register (TMU_TMHTITR)

This TMU monitor register determines the high current temperature threshold for generating the TIDR[ITTE] event.

Address: F_0000h base + 50h offset = F_0050h

Bit	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

Bit	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

TMU_TMHTITR field descriptions

Field	Description
0 EN	Enable threshold. 0 Disabled. 1 Threshold enabled.
1–23 -	This field is reserved. 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. 0-125C Sensor range 126-255C Reserved

29.2.11 TMU monitor high temperature average threshold register (TMU_TMHTATR)

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.

Address: F_0000h base + 54h offset = F_0054h

Bit	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

Bit	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

TMU_TMHTATR field descriptions

Field	Description
0 EN	Enable threshold. 0 Disabled. 1 Threshold enabled.
1–23 -	This field is reserved. Reserved
24–31 TEMP	High temperature average threshold value. Determines the average upper temperature threshold, for any enabled monitored site, that if exceeded will cause TIDR[ATTE] to be set when EN=1. 0-125C Sensor range 126-255C Reserved

29.2.12 TMU monitor high temperature average critical threshold register (TMU_TMHTACTR)

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.

Address: F_0000h base + 58h offset = F_0058h

Bit	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

Bit	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

TMU_TMHTACTR field descriptions

Field	Description
0 EN	Enable threshold. 0 Disabled. 1 Threshold enabled.
1–23 -	This field is reserved. 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. 0-125C Sensor range 126-255C Reserved

29.2.13 TMU temperature configuration register (TMU_TTCFGR)

The TMU temperature configuration register, in conjunction with the sensor configuration register, is used to initialize the internal sensor translation table used during monitoring. Freescale provides the data required.

The internal sensor translation table must be programmed with the sensor reading.

Address: F_0000h base + 80h offset = F_0080h

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	DATA																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

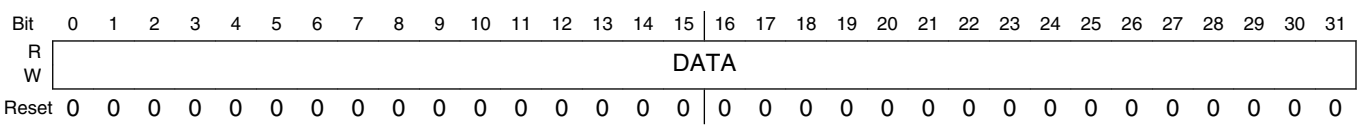
TMU_TTCFGR field descriptions

Field	Description
0–31 DATA	Sensor data.

29.2.14 TMU sensor configuration register (TMU_TSCFGR)

The TMU sensor configuration register, in conjunction with the temperature configuration register, is used to initialize the internal sensor translation table used during monitoring. Reading this register will return the data from the translation table as defined by TTCFGR. Freescale provides the data required.

Address: F_0000h base + 84h offset = F_0084h



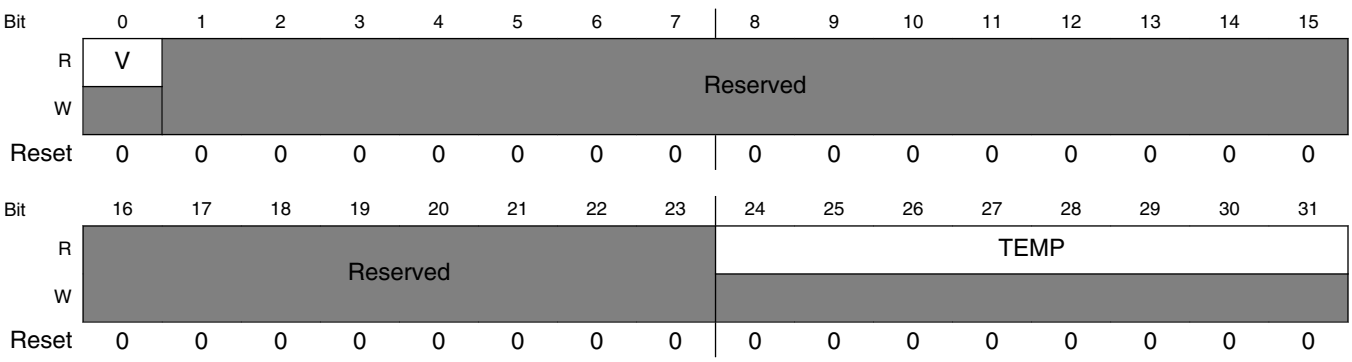
TMU_TSCFGR field descriptions

Field	Description
0–31 DATA	Sensor data.

29.2.15 TMU report immediate temperature site register n (TMU_TRITSR)

This TMU report register returns the last measured temperature at site *n*. The site must be part of the list of enabled monitored sites as defined by TMR[MSITE(0:15)].

Address: F_0000h base + 100h offset = F_0100h



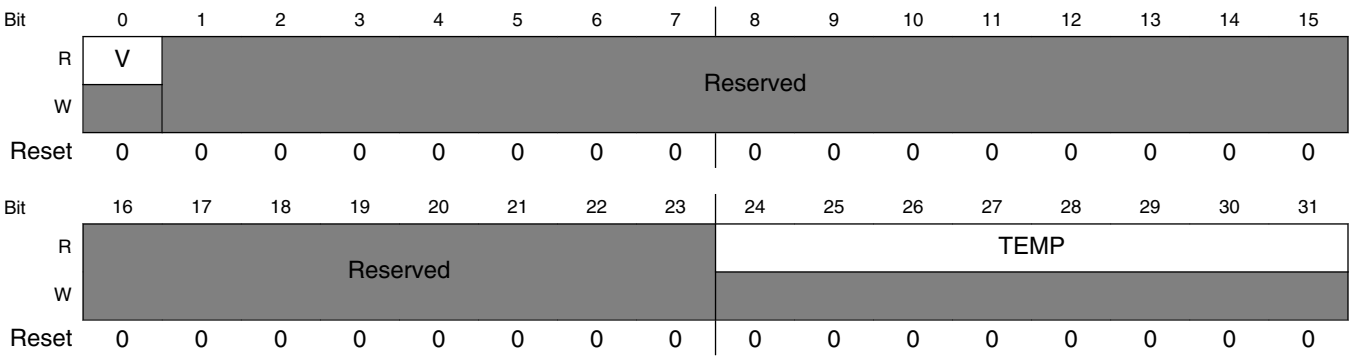
TMU_TRITSR field descriptions

Field	Description
0 V	Valid measured temperature. 0 Not valid. Temperature out of sensor range or first measurement still pending. 1 Valid.
1–23 -	This field is reserved. Reserved
24–31 TEMP	Last temperature reading at site *n* when V=1.

29.2.16 TMU report average temperature site register n (TMU_TRATSR)

This TMU report register returns the average measured temperature at site *n*. The site must be part of the list of enabled monitored sites as defined by TMR[MSITE(0:15)].

Address: F_0000h base + 104h offset = F_0104h



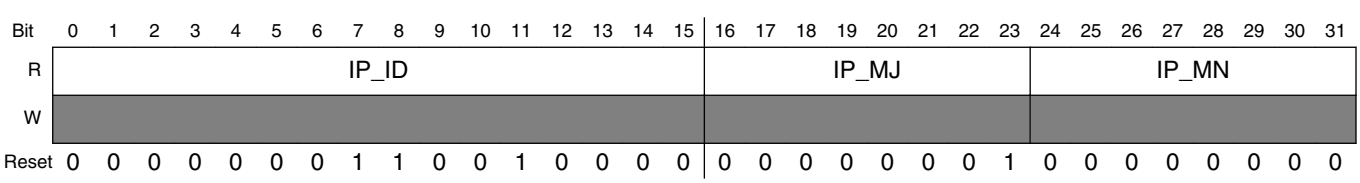
TMU_TRATSR field descriptions

Field	Description
0 V	Valid measured temperature. 0 Not valid. Temperature out of sensor range or first measurement still pending. 1 Valid.
1–23 -	This field is reserved. Reserved
24–31 TEMP	Average temperature reading at site n when V=1.

29.2.17 IP block revision register 0 (TMU_IPBRR0)

The IP block revision register 0 (IPBRR0) is used to identify the TMU block ID and revision.

Address: F_0000h base + BF8h offset = F_0BF8h



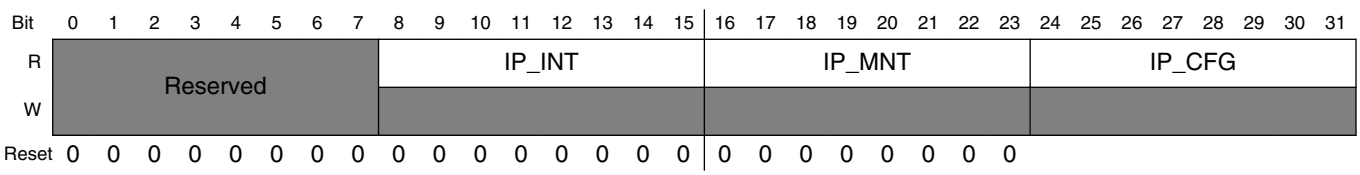
TMU_IPBRR0 field descriptions

Field	Description
0–15 IP_ID	IP block ID
16–23 IP_MJ	Major revision
24–31 IP_MN	Minor revision

29.2.18 IP block revision register 1 (TMU_IPBRR1)

The IP block revision register 1 (IPBRR1) is used to identify TMU block configuration.

Address: F_0000h base + BFCh offset = F_0BFCh



TMU_IPBRR1 field descriptions

Field	Description
0–7 -	This field is reserved. Reserved
8–15 IP_INT	IP block integration options
16–23 IP_MNT	IP block maintenance version
24–31 IP_CFG	IP block configuration options. Represents the number of temperature sensor sites. The value at bits 24-27 is the number of sites minus one. The value at bits 28-31 is always 0.

29.3 Functional Description

The following sections describe the functionality of the TMU in details.

29.3.1 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 2. 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 the interrupt capture register, TICR.
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.

29.3.2 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 0-125 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 30
Run Control and Power Management (RCPM)

30.1 Introduction

This chapter provides the specification and programming model for the RCPM.

30.1.1 Overview

The Run Control and Power Management (RCPM) Unit performs all device-level tasks associated with device run control and power management.

The RCPM unit communicates with embedded cores and coherency modules extensively to achieve both device run control and power management function.

This document focuses solely on the mechanisms of the RCPM unit that control the power managment aspects of the device and each of the processor clusters/cores/threads within the device.

Dynamic power management minimizes power consumption when a block is idle. Hypervisor software running on a master core can access RCPM memory-mapped registers (TPH10SETR, PCPH15SETR, PCPH20SETR, and PCPH30SETR, and POWMGTCSR) to put specific thread/core in PH10/PH15/PH20/PH30 state or put the device in LPM10/LPM20 mode. The RCPM unit supports several wake up sources through internal timers and internal and external interrupts.

NOTE

In this chapter, the mapping of the referenced RCPM core n to the T2080 e6500 core threads is as follows:

Table 30-1. T2080 RCPM Core n Mapping

RCPM Core n	T2080 Core/Thread
0	e6500 core 0, thread 0
1	e6500 core 0, thread 1
2	e6500 core 1, thread 0
3	e6500 core 1, thread 1
4	e6500 core 2, thread 0
5	e6500 core 2, thread 1
6	e6500 core 3, thread 0
7	e6500 core 3, thread 1

References to other RCPM cores should be disregarded.

30.1.2 Power Management Features

30.1.2.1 Power Management Features

- Software controlled power management, which minimizes power consumption of blocks when they are idle.
- Software-controlled thread PH10 mode, physical core PH15/PH20/PH30 mode, cluster PCL10 mode, and device LPM10/LPM20 mode.
- Supports OpenPIC 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 mode through core or external master (for example, PCI Express).
- Thread Power Management:
  - Independent Power Management Control of each thread
  - PH10 State where:
    - Instruction fetching is suspended
    - All previously fetched instructions are completed
    - Thread is halted but clocks remain active
  - Entry into Thread Power Management states via:
    - RCPM TPH10SETR for request to enter the thread's PH10 State
    - Event Processing Unit (EPU) triggering to enter the thread's PH10 mode
    - Device pin assertion to enter the thread's PH10 mode
  - Independent wake up from per thread:
    - Set RCPM TPH10CLRR to clear the thread PH10 request.
    - Unmasked interrupt request
    - Unmasked critical interrupt
    - Unmasked external machine check
    - NMI
    - Core wake up event
    - Debug interrupt
    - Core hard reset
    - Core debug halt request
- Core Power Management:
  - Independent power management control of each core
  - PH15 State where:

  - Entry into PH15 core power management state via RCPM PCPH15SETR for request to enter the core's PH15 state. PH15 is also triggered when MPIC requests to place core in warm reset or core is disabled by COREDISR register.
  - Entered from both threads of the core in PH10 state.
  - PH15 state wake up source is captured in Table 30-4
  - Core caches (for the core has PH15 request pending) must be explicitly flushed before entering PH15 to maintain cache coherence.
  - Clocks are gated off
- PH20 State where:
  - Entry into PH20 core power management state via RCPM PCPH20SETR for request to enter the core's PH20 state.
- Independent wake up from per thread :
  - Unmasked interrupt request
  - Unmasked critical interrupt
  - Debug interrupt
- PH30 State where:
  - Entry into PH30 core power management state via RCPM PCPH30SETR for request to enter the core's PH30 state.
  - Interrupt is ignored for core in PH30 state.
  - No state is retained.
  - The only wake up method is through core warm reset.

- Cluster Power Management:
  - PCL10 State where:
    - Clock distribution is inhibited to cluster functional unit.
    - Prior to enter PCL10 state, all cores in cluster should be placed in PH20 mode.
    - The L2 cache no longer continues to participate in snooping activities, software should always flush, then invalidate the L2 cache prior to intiating PCL10 state to ensure that any modified data is written out to backing store.
  - The entry and exit mode of cluster power management is captured in Table 30-5
- Device Power Management:
  - LPM10 State where:
    - Not all cores are in Run (PH00) state.
  - LPM20 State where:
    - All cores are in PH20 state.
    - Cluster is in PCL10 state.
    - Platform clock is disabled.
    - Entry into LPM20 via setting POWMGTCSR[LPM20_RQ]
    - LPM20 state wake up source is captured in Table 30-6

— Independent Device wake up from:
  – Unmasked interrupt
  – Unmasked critical interrupt
  – Unmasked machine check
  – Unmasked NMI

30.1.3 Modes of Operation

30.1.3.1 Mode Summary - Core & Device Levels

Table 30-2 summarizes the core RCPM modes.

Table 30-2. Core RCPM Mode Summary for Power Management

Device Mode Name	Type	Resumable	Services Snoops	Device Clocks	Core Clocks	State Retained	Book E Core State Entry	Description
Power Management Modes								
PH10	PM	Y	Y	Y	Y	All	Seq: • Full On -> Core Halted	The *core* referred to is currently in the PH10 state.
PH15		Y	N	Y	N	Latches, GPR, FPR, VPR, L1/L2 TLB, BTB, L1 Icache and Dcache tags and locks	Seq: • Full On -> Core Halted • Core Halted -> Core Stopped	The core referred to is currently in the PH15 state.
PH20		Y	N	Y	N	Latches, GPR, FPR, VPR, L1/L2 TLB, BTB, L1 Icache and Dcache tags and locks		
PH30		N	N	Y	N	None		PH 30 mode can be exited by asserting HRESET_B. PH30 mode can also be exited via core warm reset by writing to MPIC PIR.

Table 30-3 summarizes the device RCPM modes.

Table 30-3. Device RCPM Mode Summary for Power Management

Device Mode Name	Type	Resumable	Services Snoops	Device Clocks	Core Clocks	Book E Core State Entry	Description
Power Management Modes							
LPM10	PM	Y	N/A	Y	N/A	Seq: • Full On -> Core Halted • Core Halted -> Core Stopped	
LPM20	PM	Y	N	N	N	Seq: • Full On -> Core Halted • Core Halted -> Core Stopped	Platform clock to all IP are gated off. Core Timebase is turned-off.

30.1.4 Modes Entry & Exit for Power Management

Table 30-4 summarizes the entry and exit for the core power management modes.

Table 30-4. Core RCPM Mode: Entry and Exit for Power Management

Device Mode Name	Type	Entry via ...	Exit via ...	Book E Core State Entry
Power Management Modes				

PH10	PM	• Hypervisor set corresponding bit in RCPM/TPH10SETR[T*] to 1	• Set RCPM/TPH10CLRR[T*]
			• Interrupt without RCPM/TPMIMR masking.
			• Critical interrupt without RCPM/TPMCIMR masking.
			• External machine check without RCPM/TPMMCMR masking.
			• NMI without RCPM/TPMNMIMR masking.
			• The following core wake up events will unconditionally wake up core from power management state via core_wakeup_req signal.:
			  • Decrementer exception (qualified by MSR[EE] | MSR[GS])
			  • FIT exception (qualified by MSR[EE] | MSR[GS])
			  • Watchdog exception (qualified by MSR[CE] | MSR[GS])
			  • Performance interrupt (qualified by MSR[EE] | MSR[GS])
			  • Machine check asynchronous exception (exclude external initiate machine check and NMI) (qualified by MSR[ME] | MSR[GS]). This includes core unrecoverable failure during snoop processing.
			  • Doorbell interrupt (qualified by MSR[EE] | MSR[GS]).
			  • Doorbell critical interrupt (qualified by MSR[CE] | MSR[GS]).
			  • Guest doorbell (qualified by MSR[EE] | MSR[GS]).
			  • Guest doorbell critical (qualified by MSR[CE] & MSR[GS]).
			  • Guest doorbell machine check event (qualified by MSR[ME] & MSR[GS]).
			• Core timebase interrupt if enable via PCTBENR.
			• Core Hard reset event (The power management entry need to be cleared during reset to avoid the power management event re-posted)
			• Core debug halt request [1]
			• Debug interrupt without masking
			• Wake on LAN - Magic packet, if programmed
			• Wake on USB - plug/unplug, if programmed
			• Wake on eSDHC - Card detect, if programmed
			• Wake on GPIO if programmed
				Seq:
				• Full On -> Core Halted

Table 30-4. Core RCPM Mode: Entry and Exit for Power Management (continued)

Device Mode Name	Type	Entry via ...	Exit via ...	Book E Core State Entry

PH15	PM	• Hypervisor set corresponding bit in RCPM PCPH15SETR to 1.
		• MPIC core warm reset request assertion[2]
		• Core disable request by setting COREDISR[CORE] = 1
			• Hypervisor set corresponding bit in RCPM PCPH15CLRR to 1.
			• Interrupt without RCPM/TPMIMR masking.
			• Critical interrupt without RCPM/TPMCIMR masking.
			• External machine check without RCPM/TPMMCMR masking.
			• NMI without RCPM/TPMNMIMR masking.
			• The following core wake up events will unconditionally wake up core from power management state via core_wakeup_req signal.:
			  • Decrementer exception (qualified by MSR[EE] | MSR[GS])
			  • FIT exception (qualified by MSR[EE] | MSR[GS])
			  • Watchdog exception (qualified by MSR[CE] | MSR[GS])
			• Core timebase interrupt if enable via PCTBENR.
			• MPIC core warm reset request deassertion
			• Core debug halt request[1]
			• Debug interrupt without masking
			• Wake on LAN - Magic packet, if programmed
			• Wake on USB - plug/unplug, if programmed
			• Wake on eSDHC - Card detect, if programmed
			• Wake on GPIO if programmed
			• A core disabled by COREDISR can only exit by device hard reset or power on reset.
				Seq:
				• Full On -> Thread Halted
				• Thread Halted -> Core Stopped

Table 30-4. Core RCPM Mode: Entry and Exit for Power Management (continued)

Device Mode Name	Type	Entry via ...	Exit via ...	Book E Core State Entry

PH20	PM	• Hypervisor set corresponding bit in RCPM PCPH20SETR to 1	• Hypervisor set corresponding bit in RCPM PCPH20CLRR to 1
			• Interrupt without RCPM/TPMIMR masking.
			• Critical interrupt without RCPM/TPMCIMR masking.
			• External machine check without RCPM/TPMMCMR masking.
			• NMI without RCPM/TPMNMIMR masking.
			• The following core wake up events will unconditionally wake up core from power management state via core_wakeup_req signal.:
			  • Decrementer exception (qualified by MSR[EE] | MSR[GS])
			  • FIT exception (qualified by MSR[EE] | MSR[GS])
			  • Watchdog exception (qualified by MSR[CE] | MSR[GS])
			• Core timebase interrupt if enable via PCTBENR.
			• MPIC core warm reset request deassertion
			• Core debug halt request[1]
			• Debug interrupt without masking
			• Wake on LAN - Magic packet, if programmed
			• Wake on USB - plug/unplug, if programmed
			• Wake on eSDHC - Card detect, if programmed
			• Wake on GPIO if programmed
				Seq:
				• Full On -> Thread Halted
				• Thread Halted -> Core Stopped
				• Core Stopped -> Core PH20

PH30		• Hypervisor set corresponding bit in RCPM PCPH30SETR to 1	• Hypervisor set corresponding bit in RCPM PCPH30CLRR to 1 • Core hard reset. • Interrupt is ignored.	

1 For a core debut halt request to cause an exit from power management mode, the core must be operating at a frequency greater than the platform frequency.

2 A warm reset request is serviced by placing the core in the PH15 state prior to reset assertion in order for the CCM to be able to properly handle the core being reset.

Table 30-5 summarizes the entry and exit for the cluster power management mode.

Table 30-5. Cluster RCPM Mode: Entry and Exit for Power Management

Device Mode Name	Type	Entry via ...	Exit via ...	Book E Core State Entry
Power Management Modes				
PCL10	PM	• Hypervisor set corresponding bit in RCPM CLPCL10SETR register to 1.	• Hypervisor set corresponding bit in RCPM CLPCL10CLRR register to 1 to clear the power management request. (The exit from PCL10 is not recoverable in first implementation of e6500 cluster since it requires all cores being placed in PH30 prior to PCL10 state) • Any PH20 wakeup mechanism if some cores are in PH20. • Core hard reset. • Device hard reset	Seq: • Full On -> Thread Halted • Thread Halted -> Core Stopped • Core Stopped -> Core PH20 • Core PH20 -> Core

Table 30-6 summarizes the entry and exit for the device power management modes.

Table 30-6. Device RCPM Mode: Entry and Exit for Power Management

Device Mode Name	Type	Entry via ...	Exit via ...	Book E Core State Entry
Power Management Modes				

LPM20	PM	• Set RCPM POWMGTCSR[LPM20_RQ] to 1	• Clear RCPM POWMGTCSR[LPM20_RQ]
			• Core interrupt without RCPM/CPMIMR masking.
			• Core critical interrupt without RCPM/CPMCIMR masking.
			• Core external machine check without RCPM/CPMMCMR masking.
			• Core NMI without RCPM/CPMNMIMR masking.
			• The following core wake up events will unconditionally wake up core from power management state via core_wakeup_req signal.:
			  • Decrementer exception (qualified by MSR[EE] | MSR[GS])
			  • FIT exception (qualified by MSR[EE] | MSR[GS])
			  • Watchdog exception (qualified by MSR[CE] | MSR[GS])
			• Debug interrupt without masking
			• Wake on LAN - Magic packet, if programmed
			• Wake on USB - plug/unplug, if programmed
			• Wake on eSDHC - Card detect, if programmed
			• Wake on GPIO if programmed
				Seq:
				• Full On -> Core Halted
				• Core Halted -> Core Stopped

The Device RCPM Unit supports the following Power Management modes of operation:

- Reset Modes

- Power Management Modes
  - Core(s) Full On Mode
  - Thread(s) in PH10 state
  - Core(s) in PH15 state
  - Core(s) in PH20 state
  - Core(s) in PH30 (off) state
  - Cluster(s) in PCL10 state
  - Device Full On Mode
  - Device in LPM10 state
  - Device in LPM20 state

The different modes available are summarized in Table 30-3 through Table 30-6.

30.1.5 Reset Modes

30.1.5.1 Power-On Reset State

During Power-on Reset, the following hardware structures are reset:

- All CCSR registers
- SOC power management state machine
- Cluster power management state machine
- Core power management state machine
- Thread power management state machine

30.1.5.2 Hard Reset State

During Hard Reset, the following are affected:

- All CCSR registers
- SOC power management state machine
- Cluster power management state machine
- Core power management state machine
- Thread power management state machine

30.1.5.3 Power Management Modes

Hypervisor software can place the device in one of the following power modes:

- thread PH10
- core PH15
- core PH20
- core PH30
- cluster PCL10

by writing to RCPM TPH10SETR, PCPH15SETR, PCPH20SETR, PCPH30SETR or CLPCL10SETR registers respectively. In addition, Hypervisor or external masters can write to memory-mapped POWMGTCSR in RCPM to cause the device to enter one of the following device power modes:

- LPM20

30.1.5.3.1 PH10 State

In PH10 mode, the thread suspends instruction execution, significantly reducing the power consumption of the core. Snooping of the L1/L2 cache are still supported and thus the data in the data cache is kept coherent. Interrupts directed to the thread are monitored by the device and cause RCPM to use the defined handshake mechanism to exit the thread from PH10 mode to allow the thread to recognize and process the interrupt.

The core's timer facilities are still enabled during thread PH10 mode, and core timebase interrupts can be generated. All device logic external to the core remains fully operational in thread PH10 mode.

30.1.5.3.2 PH15 State

In PH15 mode, all clocks internal to the core are turned off except for its timer facilities clock (the core timebase). The L1/L2 caches do not respond to snoops in PH15 mode, so if coherency is required, the L1/L2 cache must be flushed before entering PH15 mode.

All device logic external to core remains fully operational in PH15 mode.

It is not possible for the device to place the core into PH15 mode. This is because the device cannot initiate a cache flush of the core. Since this is not possible, if the device did place a core into PH15 mode, then coherency would be lost because snooping does not occur into PH15 mode.

30.1.5.3.3 PH20 State

Core PH20 mode is core power gating with state retention.

30.1.5.3.4 PH30 State

Core PH30 mode is core power gating without state retention. Interrupt is ignored for core being placed in PH30 state.

30.1.5.3.5 PCL10 State

PCL10 is a cluster power management state which cluster clock is gated off. Due to the limitation in the first revision of the e6500 cluster implementation (required core being placed in PH30 prior to cluster PCL10), PCL10 state is not recoverable.

For cluster which is placed in PCL10 and core PH30 is a requirement for PCL10 (first revision of the e6500 cluster implementation)), interrupt to any core within PCL10 cluster should be ignored. Issue core warm reset to core within the cluster when cluster in PCL10 is illegal condition.

For cluster which is placed in PCL10 and core PH20 is a requirement for PCL10, interrupt to any core within PCL10 cluster will wake up cluster from PCL10 state to PCL00 state.

30.1.5.3.6 LPM10 State

LPM10 mode is a device state which at least one core is not in PH00 (full on state).

30.1.5.3.7 LPM20 State

In LPM20 mode, all clocks internal to the core are turned off as well as the clock in device logic so that only the modules which are required to wake up the device will still have a running clock. Core timebase is turned-off.

The modules which can be used as a wake up source are internal timers, internal and external interrupts.

After the core and I/O interfaces have shut down, ASLEEP pin is asserted.

30.2 External Signal Description

Table 30-7. RCPM Detailed Signal Descriptions

Signal	I/O	Description	
ASLEEP	O	Asleep. After negation of $\overline{\text{PORESET}}$, 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 mode. (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.
RTC	I	Real Time Clock. Please refer to the hardware specification of device for timing specification. The signal can be optionally use to clock the global timers in the programmable interrupt controller.	
		Timing	Assertion/Negation - See the hardware specification of device for specific timing information for this signal.

30.3 Memory Map and Register Definition

This section identifies Power Management resources that are not included as part of a processor core or platform IP.

Table 30-8. Device RCPM Unit Memory Map

Offset or Address	Register	Access	Reset Value	Section/Page
Thread Power Management Control/Status Registers				
0x000	Reserved:	R	—	—
0x004	Reserved:	R	—	—
0x008	Reserved:	R	—	—
0x00C	TPH10SR0 - Thread PH10 Status Register	R	0x0000_0000	30.3.1.1.1/30-17
0x010	Reserved:	R/W1S	—	—
0x014	Reserved:	R/W1S	—	—
0x018	Reserved:	R/W1S	—	—
0x01C	TPH10SETR0 - Thread PH10 Set Control Register	R/W1S	0x0000_0000	30.3.1.1.2/30-18
0x020	Reserved:	R/W1C	—	—
0x024	Reserved:	R/W1C	—	—
0x028	Reserved:	R/W1C	—	—
0x02C	TPH10CLRR - Thread PH10 Clear Control Register	R/W1C	0x0000_0000	30.3.1.1.3/30-18
0x030	Reserved:	R/W1C	—	—
0x034	Reserved:	R/W1C	—	—
0x038	Reserved:	R/W1C	—	—
0x03C	TPH10PSR0 - Thread PH10 Previous Status Register	R/W1C	0x0000_0000	30.3.1.1.4/30-19
0x040	Reserved:	R	—	—
0x044	Reserved:	R	—	—
0x048	Reserved:	R	—	—
0x04C	TWAITSR0 - Thread Wait Status Register	R	0x0000_0000	30.3.1.1.5/30-20
Physical Core Power Management Control/Status Registers				
0x0B0	PCPH15SR - Physical Core PH15 Status Register	R	0x0000_0000	30.3.1.2.1/30-22
0x0B4	PCPH15SETR - Physical Core PH15 Set Control Register	R/W1S	0x0000_0000	30.3.1.2.2/30-22
0x0B8	PCPH15CLRR - Physical Core PH15 Clear Control Register	R/W1C	0x0000_0000	30.3.1.2.3/30-23

Table 30-8. Device RCPM Unit Memory Map

Offset or Address	Register	Access	Reset Value	Section/Page
0x0BC	PCPH15PSR - Physical Core PH15 Previous Status Register	R	0x0000_0000	30.3.1.2.4/30-24
0x0D0	PCPH20SR - Physical Core PH20 Status Register	R	0x0000_0000	30.3.1.2.5/30-25
0x0D4	PCPH20SETR - Physical Core PH20 Set Control Register	R/W1S	0x0000_0000	30.3.1.2.6/30-26
0x0D8	PCPH20CLRR - Physical Core PH20 Clear Control Register	R/W1C	0x0000_0000	30.3.1.2.7/30-26
0x0DC	PCPH20PSR - Physical Core PH20 Previous Status Register	R	0x0000_0000	30.3.1.2.8/30-27
0x0E0	PCPW20SR - Physical Core PW20 Status Register	R	0x0000_0000	30.3.1.2.9/30-28
0x0F0	PCPH30SR - Physical Core PH30 Status Register	R	0x0000_0000	/30-28
0x0F4	PCPH30SETR - Physical Core PH30 Set Control Register	R/W1S	0x0000_0000	30.3.1.2.11/30-29
0x0F8	PCPH30CLRR - Physical Core PH30 Clear Control Register	R/W1C	0x0000_0000	30.3.1.2.12/30-30
0x0FC	PCPH30PSR - Physical Core PH30 Previous Status Register	R	0x0000_0000	30.3.1.2.13/30-30
SoC Power Management Control/Status Registers				
0x130	POWMGTCSR — Power Management Control and Status Register	Mixed	0x0000_0000	30.3.1.3.1/30-31
0x140	IPPDEXPCRn - IP Powerdown Exception Control Register n	R/W	0x0000_0000	30.3.1.3.2/30-33
0x144 -0x14C	Reserved: IPPDEXPCRn	R/W	—	—
Thread Interrupt Masking Registers				
0x150	Reserved:	R/W	—	—
0x154	Reserved:	R/W	—	—
0x158	Reserved:	R/W	—	—
0x15C	TPMIMR0 - Thread Power Management Interrupt Masking Register 0	R/W	0x0000_0000	30.3.1.4.1/30-34
0x160	Reserved:	R/W	—	—
0x164	Reserved:	R/W	—	—
0x168	Reserved:	R/W	—	—
0x16C	TPMCIMR0 - Thread Power Management Critical Interrupt Masking Register 0	R/W	0x0000_0000	30.3.1.4.2/30-35

Table 30-8. Device RCPM Unit Memory Map

Offset or Address	Register	Access	Reset Value	Section/Page
0x170	Reserved:	R/W	—	—
0x174	Reserved:	R/W	—	—
0x178	Reserved:	R/W	—	—
0x17C	TPMMCMR0 - Thread Power Management Machine Check Masking Register 0	R/W	0x0000_0000	30.3.1.4.3/30-36
0x180	Reserved:	R/W	—	—
0x184	Reserved:	R/W	—	—
0x188	Reserved:	R/W	—	—
0x18C	TPMNMIMR0 - Thread Power Management NMI Masking Register 0	R/W	0x0000_0000	30.3.1.4.4/30-37
0x190	Reserved:	R/W	—	—
0x194	Reserved:	R/W	—	—
0x198	Reserved:	R/W	—	—
0x19C	TMCPMASKCR0 - Thread Machine Check Mask Control Register 0	R/W	0x0000_0000	30.3.1.4.5/30-38
Physical Core Time Base Registers				
0x1A0	PCTBENR — Physical Core Time Base Enable Register	R/W	0x0000_0000	30.3.1.5.1/30-40
0x1A4	PCTBCKSELR — Physical Core Time Base Clock Select Register	R/W	0x0000_0000	30.3.1.5.2/30-40
0x1A8	TBCLKDIVR - Time Base Clock Divider Register	R/W	0xnn00_0000	30.3.1.5.3/30-41
0x1B0	Reserved:	R/W	—	—
0x1B4	Reserved:	R/W	—	—
0x1B8	Reserved:	R/W	—	—
0x1BC	TTBHLTCR0 — Thread Time Base Halt Control Register 0	R/W	0x0000_0000	30.3.1.5.4/30-42
Cluster Power Management Control/Status Registers				
0x1C0	CLPCL10SR - Cluster PCL10 Status Register	R	0x0000_0000	30.3.1.6.1/30-43
0x1C4	CLPCL10SETR - Cluster PCL10 Set Control Register	R/W1S	0x0000_0000	30.3.1.6.2/30-43
0x1C8	CLPCL10CLRR - Cluster PCL10 Clear Control Register	R/W1C	0x0000_0000	30.3.1.6.3/30-44
0x1CC	CLPCL10PSR - Cluster PCL10 Previous Status Register	R	0x0000_0000	30.3.1.6.4/30-45

30.3.1 Register Descriptions (CCSR)

30.3.1.1 Thread Power Management Control/Status Registers

30.3.1.1.1 Thread PH10 Status Register (TPH10SR0-TPH10SR3)

This is one of a set of thread power management registers. This register is used for reporting *PH10 status* per thread.

Offset TPH10SR3: 0x000; TPH10SR2: 0x004; TPH10SR1: 0x008; TPH10SR0: 0x00C — Access: Read Only

	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	—																								T7	T6	T5	T4	T3	T2	T1	T0
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

Figure 30-1. Thread PH10 Status Register n (TPH10SRn)

The table below describes this register's bit settings.

Table 30-9. TPH10SRn Field Descriptions

Bits	Name	Description
0-23	—	Reserved
24	T7	Thread 7 PH10 Status. 0 Thread 7 is not in the PH10 mode 1 Thread 7 is in the PH10 mode
25	T6	Thread 6 PH10 Status. 0 Thread 6 is not in the PH10 mode 1 Thread 6 is in the PH10 mode
26	T5	Thread 5 PH10 Status. 0 Thread 5 is not in the PH10 mode 1 Thread 5 is in the PH10 mode
27	T4	Thread 4 PH10 Status. 0 Thread 4 is not in the PH10 mode 1 Thread 4 is in the PH10 mode
28	T3	Thread 3 PH10 Status. 0 Thread 3 is not in the PH10 mode 1 Thread 3 is in the PH10 mode
29	T2	Thread 2 PH10 Status. 0 Thread 2 is not in the PH10 mode 1 Thread 2 is in the PH10 mode

Table 30-9. TPH10SRn Field Descriptions (continued)

Bits	Name	Description
30	T1	Thread 1 PH10 Status. 0 Thread 1 is not in the PH10 mode 1 Thread 1 is in the PH10 mode
31	T0	Thread 0 PH10 Status. 0 Thread 0 is not in the PH10 mode 1 Thread 0 is in the PH10 mode

30.3.1.1.2 Thread PH10 Set Control Register (TPH10SETR0-TPH10SETR3)

This is one of a set of Thread Power Management Registers. This register is used for requesting that a thread be placed in the *PH10* state. The true value read from the register means there is a pending thread PH10 request.

Address TPH10SETR3: 0x010, TPH10SETR2: 0x014, TPH10SET1: 0x018, TPH10SETR0: 0x01C — Access: Read / Write -one-set

	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	T_PH10_REQ																															
W	w1s																															
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

Figure 30-2. Thread PH10 Set Control Register (TPH10SETRn)

The table below describes this register's bit settings.

Table 30-10. TPH10SETRn Field Descriptions

Bits	Name	Description
0-31	T_PH10_REQ	Write one to trigger thread PH10 request. This bit is clear by interrupt event or write-one event on TPH10CLRRn register. xxxxxxxx_xxxxxxxx_xxxxxxxx_xxxxxxx1 Request to put thread 0 in PH10 state. xxxxxxxx_xxxxxxxx_xxxxxxxx_xxxxxx1x Request to put thread 1 in PH10 state. xxxxxxxx_xxxxxxxx_xxxxxxxx_xxxxx1xx Request to put thread 2 in PH10 state. xxxxxxxx_xxxxxxxx_xxxxxxxx_xxxx1xxx Request to put thread 3 in PH10 state. xxxxxxxx_xxxxxxxx_xxxxxxxx_xxx1xxxx Request to put thread 4 in PH10 state. xxxxxxxx_xxxxxxxx_xxxxxxxx_xx1xxxxx Request to put thread 5 in PH10 state. xxxxxxxx_xxxxxxxx_xxxxxxxx_x1xxxxxx Request to put thread 6 in PH10 state. xxxxxxxx_xxxxxxxx_xxxxxxxx_1xxxxxxx Request to put thread 7 in PH10 state.

30.3.1.1.3 Thread PH10 Clear Control Register (TPH10CLRR)

This is one of a set of Thread Power Management Registers. This register is used for waking up the thread be placed in the *PH10* state. The true value read from the register only means there is a pending thread PH10 request.

Address TPH10CLRR3: 0x020, TPH10CLRR2: 0x024, TPH10CLRR1: 0x028, TPH10CLRR0: 0x02C — Access: Read / Write -one-clear

	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	T_PH10_REQ																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 30-3. Thread PH10 Clear Control Register (TPH10CLRR)

The table below describes this register's bit settings.

Table 30-11. TPH10CLRR Field Descriptions

Bits	Name	Description
0-31	T_PH10_REQ	Write one to cancel thread PH10 request. The bit is set by write-one-event to TPH10SETR register. This bit is also clear by interrupt event. The read true value of the register bit means there is pending thread PH10 request for that thread. xxxxxxxx_xxxxxxxx_xxxxxxxx_xxxxxxx1 thread0 is receiving a PH10 request from TPH10SETR xxxxxxxx_xxxxxxxx_xxxxxxxx_xxxxxx1x thread1 is receiving a PH10 request from TPH10SETR xxxxxxxx_xxxxxxxx_xxxxxxxx_xxxxx1xx thread2 is receiving a PH10 request from TPH10SETR xxxxxxxx_xxxxxxxx_xxxxxxxx_xxxx1xxx thread3 is receiving a PH10 request from TPH10SETR xxxxxxxx_xxxxxxxx_xxxxxxxx_xxx1xxxx thread4 is receiving a PH10 request from TPH10SETR xxxxxxxx_xxxxxxxx_xxxxxxxx_xx1xxxxx thread5 is receiving a PH10 request from TPH10SETR xxxxxxxx_xxxxxxxx_xxxxxxxx_x1xxxxxx thread6 is receiving a PH10 request from TPH10SETR xxxxxxxx_xxxxxxxx_xxxxxxxx_1xxxxxxx thread7 is receiving a PH10 request from TPH10SETR

30.3.1.1.4 Thread PH10 Previous Status Register (TPH10PSR)

This is one of a set of Thread Power Management Registers. This register is used for reporting previous *PH10 status* per thread. It is used by software to know which power saving state it was in before wake up by interrupt.

Address TPH10PSR3: 0x030,TPH10PSR2: 0x034, TPH10PSR1: 0x038, TPH10PSR0: 0x03C — Access: Read Only

	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	—																								T_P_PH10_7	T_P_PH10_6	T_P_PH10_5	T_P_PH10_4	T_P_PH10_3	T_P_PH10_2	T_P_PH10_1	T_P_PH10_0
W																									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	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 30-4. Thread PH10 Previous Status Register (TPH10PSR)

The table below describes this register's bit settings.

Table 30-12. TPH10PSR Field Description

Bits	Name	Description
0–23	—	Reserved
24	T_P_PH10_7	Thread PH10 previous status. The bit is set when corresponding TPH10SR bit has 1 to 0 transition and it is caused by interrupt. The bit is w1c. 0 Thread was not in the PH10 mode before wake up by interrupt. 1 Thread was in the PH10 mode before wake up by interrupt.
25	T_P_PH10_6	Thread PH10 previous status. The bit is set when corresponding TPH10SR bit has 1 to 0 transition and it is caused by interrupt. The bit is w1c. 0 Thread was not in the PH10 mode before wake up by interrupt. 1 Thread was in the PH10 mode before wake up by interrupt.
26	T_P_PH10_5	Thread PH10 previous status. The bit is set when corresponding TPH10SR bit has 1 to 0 transition and it is caused by interrupt. The bit is w1c. 0 Thread was not in the PH10 mode before wake up by interrupt. 1 Thread was in the PH10 mode before wake up by interrupt.
27	T_P_PH10_4	Thread PH10 previous status. The bit is set when corresponding TPH10SR bit has 1 to 0 transition and it is caused by interrupt. The bit is w1c. 0 Thread was not in the PH10 mode before wake up by interrupt. 1 Thread was in the PH10 mode before wake up by interrupt.
28	T_P_PH10_3	Thread PH10 previous status. The bit is set when corresponding TPH10SR bit has 1 to 0 transition and it is caused by interrupt. The bit is w1c. 0 Thread was not in the PH10 mode before wake up by interrupt. 1 Thread was in the PH10 mode before wake up by interrupt.
29	T_P_PH10_2	Thread PH10 previous status. The bit is set when corresponding TPH10SR bit has 1 to 0 transition and it is caused by interrupt. The bit is w1c. 0 Thread was not in the PH10 mode before wake up by interrupt. 1 Thread was in the PH10 mode before wake up by interrupt.
30	T_P_PH10_1	Thread PH10 previous status. The bit is set when corresponding TPH10SR bit has 1 to 0 transition and it is caused by interrupt. The bit is w1c. 0 Thread was not in the PH10 mode before wake up by interrupt. 1 Thread was in the PH10 mode before wake up by interrupt.
31	T_P_PH10_0	Thread PH10 previous status. The bit is set when corresponding TPH10SR bit has 1 to 0 transition and it is caused by interrupt. The bit is w1c. 0 Thread was not in the PH10 mode before wake up by interrupt. 1 Thread was in the PH10 mode before wake up by interrupt.

30.3.1.1.5 Thread Wait Status Register (TWAITSR)

This is one of a set of Thread Power Management Registers. This register is used for reporting *wait status* per thread.

Address TWAITSR3:0x040, TWAITSR2: 0x044, TWAITSR1: 0x048, TWAITSR0: 0x04C Access: Read Only

	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	—																								T_WAITING7	T_WAITING6	T_WAITING5	T_WAITING4	T_WAITING3	T_WAITING2	T_WAITING1	T_WAITING0
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

Figure 30-5. Thread Wait Status Register (TWAITSR)

NOTE

The table below describes this register's bit settings.

Table 30-13. TWAITSR Field Descriptions

Bits	Name	Description
0–23	—	Reserved
24	T_WAITING7	Thread Wait Status. 0 Thread is not in its Wait mode 1 Thread is in its Wait mode
25	T_WAITING6	Thread Wait Status. 0 Thread is not in its Wait mode 1 Thread is in its Wait mode
26	T_WAITING5	Thread Wait Status. 0 Thread is not in its Wait mode 1 Thread is in its Wait mode
27	T_WAITING4	Thread Wait Status. 0 Thread is not in its Wait mode 1 Thread is in its Wait mode
28	T_WAITING3	Thread Wait Status. 0 Thread is not in its Wait mode 1 Thread is in its Wait mode
29	T_WAITING2	Thread Wait Status. 0 Thread is not in its Wait mode 1 Thread is in its Wait mode
30	T_WAITING1	Thread Wait Status. 0 Thread is not in its Wait mode 1 Thread is in its Wait mode
31	T_WAITING0	Thread Wait Status. 0 Thread is not in its Wait mode 1 Thread is in its Wait mode

30.3.1.2 Physical Core Power Management Control/Status Registers

30.3.1.2.1 Physical Core PH15 Status Register (PCPH15SR)

This is one of a set of Physical Core Power Management Registers. This register is used for reporting *PH15 status* per physical core.

Address 0x0B0 Access: Read Only

	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																													PC3	PC2	PC1	PC0
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

Figure 30-6. Physical Core PH15 Status Register Lower (PCPH15SR)

The table below describes this register's bit settings.

Table 30-14. PCPH15SR Field Descriptions

Bits	Name	Description
0–27	—	Reserved
28	PC3	Physical Core PH15 Status. 0 Physical Core is not in its PH15 mode 1 Physical Core is in its PH15 mode
29	PC2	Physical Core PH15 Status. 0 Physical Core is not in its PH15 mode 1 Physical Core is in its PH15 mode
30	PC1	Physical Core PH15 Status. 0 Physical Core is not in its PH15 mode 1 Physical Core is in its PH15 mode
31	PC0	Physical Core PH15 Status. 0 Physical Core is not in its PH15 mode 1 Physical Core is in its PH15 mode

30.3.1.2.2 Physical Core PH15 Set Control Register (PCPH15SETR)

This is one of a set of Physical Core Power Management Registers. This register is used for requesting that a physical core be placed in the *PH15* state. The true value read from the register means there is a pending physical core PH15 request.

Address 0x0B4 Access: Read / Write -one-set

	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	PC_PH15_REQ																															
W	w1s																															
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

Figure 30-7. Physical Core PH15 Set Control Register (PCPH15SETRn)

The table below describes this register's bit settings.

Table 30-15. PCPH15SETRn Field Descriptions

Bits	Name	Description
0-31	PC_PH15_RQ	Write one to trigger physical core PH15 request. This bit is clear by interrupt event or write-one event on PCPH15CLRRn register. xxxxxxxx_xxxxxxxx_xxxxxxxx_xxxxxxx1 Request to put physical core 0 in PH15 state. xxxxxxxx_xxxxxxxx_xxxxxxxx_xxxxxx1x Request to put physical core 1 in PH15 state. xxxxxxxx_xxxxxxxx_xxxxxxxx_xxxxx1xx Request to put physical core 2 in PH15 state. xxxxxxxx_xxxxxxxx_xxxxxxxx_xxxx1xxx Request to put physical core 3 in PH15 state.

30.3.1.2.3 Physical Core PH15 Clear Control Register (PCPH15CLRR)

This is one of a set of Physical Core Power Management Registers. This register is used for waking up the thread be placed in the *PH15* state. The true value read from the register only means there is a pending physical core PH15 request.

Address 0x0B8 Access: Read / Write -one-clear

	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	CP_PH15_REQ																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 30-8. Physical Core PH15 Clear Control Register (PCPH15CLRR)

The table below describes this register's bit settings.

Table 30-16. PCPH15CLRR Field Descriptions

Bits	Name	Description
0-31	CP_PH15_RQ	Write one to cancel physical core PH15 request. The bit is set by write-one-event to PCPH15SETR register. This bit is also clear by interrupt event. The read true value of the register bit means there is pending Physical Core PH15 request for that core. xxxxxxxx_xxxxxxxx_xxxxxxxx_xxxxxxx1 core0 is receiving a PH15 request from PCPH15SETR xxxxxxxx_xxxxxxxx_xxxxxxxx_xxxxxx1x core1 is receiving a PH15 request from PCPH15SETR xxxxxxxx_xxxxxxxx_xxxxxxxx_xxxxx1xx core2 is receiving a PH15 request from PCPH15SETR xxxxxxxx_xxxxxxxx_xxxxxxxx_xxxx1xxx core3 is receiving a PH15 request from PCPH15SETR

30.3.1.2.4 Physical Core PH15 Previous Status Register (PCPH15PSR)

This is one of a set of Physical Core Power Management Registers. This register is used for reporting previous *PH15 status* per physical core. It is used by software to know which power saving state it was in before wake up by interrupt.

Address 0x0BC Access: Read Only

	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																													PC_P_PH15_3	PC_P_PH15_2	PC_P_PH15_1	PC_P_PH15_0
W																													W1C	W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 30-9. Physical Core PH15 Previous Status Register (PCPH15PSR)

The table below describes this register's bit settings.

Table 30-17. PCPH15PSR Field Description

Bits	Name	Description
0–27	—	Reserved
28	PC_P_PH15_3	Physical core PH15 previous status. The bit is set when corresponding PCPH15SR bit has 1 to 0 transition and it is caused by interrupt. The bit is w1c. 0 Physical core was not in the PH15 mode before wake up by interrupt. 1 Physical core was in the PH15 mode before wake up by interrupt.
29	PC_P_PH15_2	Physical core PH15 previous status. The bit is set when corresponding PCPH15SR bit has 1 to 0 transition and it is caused by interrupt. The bit is w1c. 0 Physical core was not in the PH15 mode before wake up by interrupt. 1 Physical core was in the PH15 mode before wake up by interrupt.

Table 30-17. PCPH15PSR Field Description

Bits	Name	Description
30	PC_P_PH15_1	Physical core PH15 previous status. The bit is set when corresponding PCPH15SR bit has 1 to 0 transition and it is caused by interrupt. The bit is w1c. 0 Physical core was not in the PH15 mode before wake up by interrupt. 1 Physical core was in the PH15 mode before wake up by interrupt.
31	PC_P_PH15_0	Physical core PH15 previous status. The bit is set when corresponding PCPH15SR bit has 1 to 0 transition and it is caused by interrupt. The bit is w1c. 0 Physical core was not in the PH15 mode before wake up by interrupt. 1 Physical core was in the PH15 mode before wake up by interrupt.

30.3.1.2.5 Physical Core PH20 Status Register (PCPH20SR)

This is one of a set of Physical Core Power Management Registers. This register is used for reporting *PH20 status* per physical core.

Address 0x0D0 Access: Read Only

	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	—																												PC3	PC2	PC1	PC0
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

Figure 30-10. Physical Core PH20 Status Register Lower (PCPH20SR)

The table below describes this register's bit settings.

Table 30-18. PCPH20SR Field Descriptions

Bits	Name	Description
0–27	—	Reserved
28	PC3	Physical Core PH20 Status. 0 Physical Core is not in its PH20 mode 1 Physical Core is in its PH20 mode
29	PC2	Physical Core PH20 Status. 0 Physical Core is not in its PH20 mode 1 Physical Core is in its PH20 mode
30	PC1	Physical Core PH20 Status. 0 Physical Core is not in its PH20 mode 1 Physical Core is in its PH20 mode
31	PC0	Physical Core PH20 Status. 0 Physical Core is not in its PH20 mode 1 Physical Core is in its PH20 mode

30.3.1.2.6 Physical Core PH20 Set Control Register (PCPH20SETR)

This is one of a set of Physical Core Power Management Registers. This register is used for requesting that a physical core be placed in the *PH20* state. The true value read from the register means there is a pending physical core PH20 request.

Address 0x0D4 Access: Read / Write -one-set

	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	PC_PH20_REQ																															
W	w1s																															
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

Figure 30-11. Physical Core PH20 Set Control Register (PCPH20SETRn)

The table below describes this register's bit settings.

Table 30-19. PCPH20SETRn Field Descriptions

Bits	Name	Description
0-31	PC_PH20_RQ	Write one to trigger physical core PH20 request. This bit is clear by interrupt event or write-one event on PCPH20CLRRn register. xxxxxxxx_xxxxxxxx_xxxxxxxx_xxxxxxx1 Request to put physical core 0 in PH20 state. xxxxxxxx_xxxxxxxx_xxxxxxxx_xxxxxx1x Request to put physical core 1 in PH20 state. xxxxxxxx_xxxxxxxx_xxxxxxxx_xxxxx1xx Request to put physical core 2 in PH20 state. xxxxxxxx_xxxxxxxx_xxxxxxxx_xxxx1xxx Request to put physical core 3 in PH20 state.

30.3.1.2.7 Physical Core PH20 Clear Control Register (PCPH20CLRR)

This is one of a set of Physical Core Power Management Registers. This register is used for waking up the thread be placed in the *PH20* state. The true value read from the register only means there is a pending physical core PH20 request.

Address 0x0D8 Access: Read / Write -one-clear

	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	CP_PH20_REQ																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 30-12. Physical Core PH20 Clear Control Register (PCPH20CLRR)

The table below describes this register's bit settings.

Table 30-20. PCPH20CLRR Field Descriptions

Bits	Name	Description
0-31	CP_PH20_RQ	Write one to cancel physical core PH20 request. The bit is set by write-one-event to PCPH20SETR register. This bit is also clear by interrupt event. The read true value of the register bit means there is pending Physical Core PH20 request for that core. xxxxxxxx_xxxxxxxx_xxxxxxxx_xxxxxxx1 core0 is receiving a PH20 request from PCPH20SETR xxxxxxxx_xxxxxxxx_xxxxxxxx_xxxxxx1x core1 is receiving a PH20 request from PCPH20SETRxxxxxxxx_xxxxxxxx_xxxxxxxx_xxxxx1xx core2 is receiving a PH20 request from PCPH20SETR xxxxxxxx_xxxxxxxx_xxxxxxxx_xxxx1xxx core3 is receiving a PH20 request from PCPH20SETR

30.3.1.2.8 Physical Core PH20 Previous Status Register (PCPH20PSR)

This is one of a set of Physical Core Power Management Registers. This register is used for reporting previous *PH20 status* per physical core. It is used by software to know which power saving state it was in before wake up by interrupt.

Address 0x0DC Access: Read Only

	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	—																												PC_P_PH20_3	PC_P_PH20_2	PC_P_PH20_1	PC_P_PH20_0
W																													W1C	W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 30-13. Physical Core PH20 Previous Status Register (PCPH20PSR)

The table below describes this register's bit settings.

Table 30-21. PCPH20PSR Field Description

Bits	Name	Description
0–27	—	Reserved
28	PC_P_PH20_3	Physical core PH20 previous status. The bit is set when corresponding PCPH20SR bit has 1 to 0 transition and it is caused by interrupt. The bit is w1c. 0 Physical core was not in the PH20 mode before wake up by interrupt. 1 Physical core was in the PH20 mode before wake up by interrupt.
29	PC_P_PH20_2	Physical core PH20 previous status. The bit is set when corresponding PCPH20SR bit has 1 to 0 transition and it is caused by interrupt. The bit is w1c. 0 Physical core was not in the PH20 mode before wake up by interrupt. 1 Physical core was in the PH20 mode before wake up by interrupt.

Table 30-21. PCPH20PSR Field Description

Bits	Name	Description
30	PC_P_PH20_1	Physical core PH20 previous status. The bit is set when corresponding PCPH20SR bit has 1 to 0 transition and it is caused by interrupt. The bit is w1c. 0 Physical core was not in the PH20 mode before wake up by interrupt. 1 Physical core was in the PH20 mode before wake up by interrupt.
31	PC_P_PH20_0	Physical core PH20 previous status. The bit is set when corresponding PCPH20SR bit has 1 to 0 transition and it is caused by interrupt. The bit is w1c. 0 Physical core was not in the PH20 mode before wake up by interrupt. 1 Physical core was in the PH20 mode before wake up by interrupt.

30.3.1.2.9 Physical Core PW20 Status Register (PCPW20SR)

This is one of a set of Physical Core Power Management Registers. This register is used for reporting *PW20 status* per physical core.

Address 0x0E0 Access: Read Only

	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	—																												PC3	PC2	PC1	PC0
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

Figure 30-14. Physical Core PW20 Status Register Lower (PCPW20SR)

The table below describes this register's bit settings.

Table 30-22. PCPW20SR Field Descriptions

Bits	Name	Description
0–27	—	Reserved
28	PC3	Physical Core PW20 Status. 0 Physical Core is not in its PW20 mode 1 Physical Core is in its PW20 mode
29	PC2	Physical Core PW20 Status. 0 Physical Core is not in its PW20 mode 1 Physical Core is in its PW20 mode
30	PC1	Physical Core PW20 Status. 0 Physical Core is not in its PW20 mode 1 Physical Core is in its PW20 mode
31	PC0	Physical Core PW20 Status. 0 Physical Core is not in its PW20 mode 1 Physical Core is in its PH30 mode

30.3.1.2.10 Physical Core PH30 Status Register (PCPH30SR)

This is one of a set of Physical Core Power Management Registers. This register is used for reporting *PH30 status* per physical core.

Address 0x0F Access: Read Only

	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	—																												PC3	PC2	PC1	PC0
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

Figure 30-15. Physical Core PH30 Status Register Lower (PCPH30SR)

The table below describes this register's bit settings.

Table 30-23. PCPH30SR Field Descriptions

Bits	Name	Description
0–27	—	Reserved
28	PC3	Physical Core PH30 Status. 0 Physical Core is not in its PH30 mode 1 Physical Core is in its PH30 mode
29	PC2	Physical Core PH30 Status. 0 Physical Core is not in its PH30 mode 1 Physical Core is in its PH30 mode
30	PC1	Physical Core PH30 Status. 0 Physical Core is not in its PH30 mode 1 Physical Core is in its PH30 mode
31	PC0	Physical Core PH30 Status. 0 Physical Core is not in its PH30 mode 1 Physical Core is in its PH30 mode

30.3.1.2.11 Physical Core PH30 Set Control Register (PCPH30SETR)

This is one of a set of Physical Core Power Management Registers. This register is used for requesting that a physical core be placed in the *PH30* state. The true value read from the register means there is a pending physical core PH30 request.

Address 0x0F4 Access: Read / Write -one-set

	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	PC_PH30_REQ																															
W	w1s																															
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

Figure 30-16. Physical Core PH30 Set Control Register (PCPH30SETRn)

The table below describes this register's bit settings.

Table 30-24. PCPH30SETRn Field Descriptions

Bits	Name	Description
0-31	PC_PH30_RQ	Write one to trigger physical core PH30 request. This bit is clear by interrupt event or write-one event on PCPH30CLRRn register. xxxxxxxx_xxxxxxxx_xxxxxxxx_xxxxxxx1 Request to put physical core 0 in PH30 state. xxxxxxxx_xxxxxxxx_xxxxxxxx_xxxxxx1x Request to put physical core 1 in PH30 state. xxxxxxxx_xxxxxxxx_xxxxxxxx_xxxxx1xx Request to put physical core 2 in PH30 state. xxxxxxxx_xxxxxxxx_xxxxxxxx_xxxx1xxx Request to put physical core 3 in PH30 state

30.3.1.2.12 Physical Core PH30 Clear Control Register (PCPH30CLRR)

This is one of a set of Physical Core Power Management Registers. This register is used for waking up the thread be placed in the *PH30* state. The true value read from the register only means there is a pending physical core PH30 request.

Address 0x0F8 Access: Read / Write -one-clear

	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	CP_PH30_REQ																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 30-17. Physical Core PH30 Clear Control Register (PCPH30CLRR)

The table below describes this register's bit settings.

Table 30-25. PCPH30CLRR Field Descriptions

Bits	Name	Description
0-31	CP_PH30_RQ	Write one to cancel physical core PH30 request. The bit is set by write-one-event to PCPH30SETR register. This bit is also clear by interrupt event. The read true value of the register bit means there is pending Physical Core PH30 request for that core. xxxxxxxx_xxxxxxxx_xxxxxxxx_xxxxxxx1 core0 is receiving a PH30 request from PCPH30SETR xxxxxxxx_xxxxxxxx_xxxxxxxx_xxxxxx1x core1 is receiving a PH30 request from PCPH30SETRxxxxxxxx_xxxxxxxx_xxxxxxxx_xxxxx1xx core2 is receiving a PH30 request from PCPH30SETR xxxxxxxx_xxxxxxxx_xxxxxxxx_xxxx1xxx core3 is receiving a PH30 request from PCPH30SETR

30.3.1.2.13 Physical Core PH30 Previous Status Register (PCPH30PSR)

This is one of a set of Physical Core Power Management Registers. This register is used for reporting previous *PH30 status* per physical core. It is used by software to know which power saving state it was in before wake up by interrupt.

Address 0x0FC Access: Read Only

	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																													PC_P_PH30_3	PC_P_PH30_2	PC_P_PH30_1	PC_P_PH30_0
W																													W1C	W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 30-18. Physical Core PH30 Previous Status Register (PCPH30PSR)

The table below describes this register's bit settings.

Table 30-26. PCPH30PSR Field Description

Bits	Name	Description
0–27	—	Reserved
28	PC_P_PH30_3	Physical core PH30 previous status. The bit is set when corresponding PCPH30SR bit has 1 to 0 transition and it is caused by interrupt. The bit is w1c. 0 Physical core was not in the PH30 mode before wake up by interrupt. 1 Physical core was in the PH30 mode before wake up by interrupt.
29	PC_P_PH30_2	Physical core PH30 previous status. The bit is set when corresponding PCPH30SR bit has 1 to 0 transition and it is caused by interrupt. The bit is w1c. 0 Physical core was not in the PH30 mode before wake up by interrupt. 1 Physical core was in the PH30 mode before wake up by interrupt.
30	PC_P_PH30_1	Physical core PH30 previous status. The bit is set when corresponding PCPH30SR bit has 1 to 0 transition and it is caused by interrupt. The bit is w1c. 0 Physical core was not in the PH30 mode before wake up by interrupt. 1 Physical core was in the PH30 mode before wake up by interrupt.
31	PC_P_PH30_0	Physical core PH30 previous status. The bit is set when corresponding PCPH30SR bit has 1 to 0 transition and it is caused by interrupt. The bit is w1c. 0 Physical core was not in the PH30 mode before wake up by interrupt. 1 Physical core was in the PH30 mode before wake up by interrupt.

30.3.1.3 SoC Power Management Control / Status Registers

30.3.1.3.1 POWMGTCSR (Power Management Control and Status Register)

This is for performing Power Management at the device level. This register is used for entering the device's LPM state. In addition, it provides status indicating successful entry into the correspinding power management state.

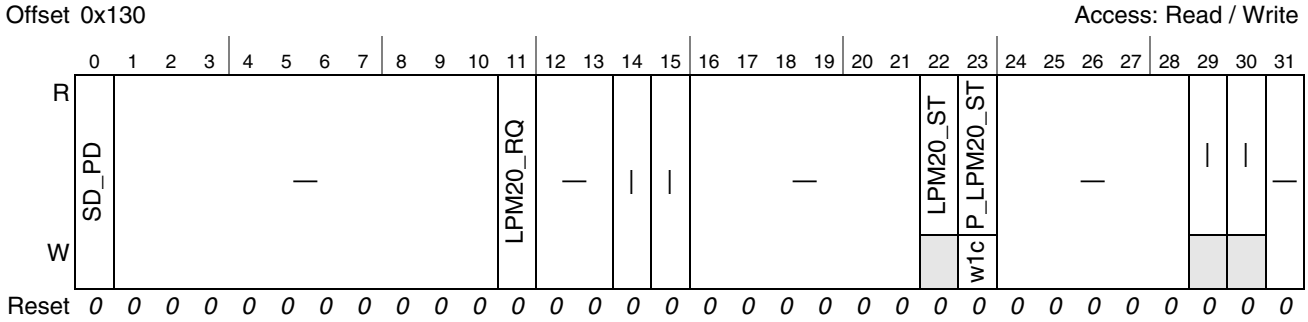


Figure 30-19. Power Management Control and Status Register n (POWMGTCSR)

NOTE

If debugger wants to access memory content in LPM20 , it can put the device back to debug quiesced state. In quiesced state, the debugger can issue both coherent and non-coherent memory access and the content in the memory space should be the same as in LPM20 mode.

The table below describes this register's bit settings.

Table 30-27. POWMGTCSR Field Description

Bits	Name	Description
0	SD_PD	SerDes and Protocol Converter Powerdown Control 0 SerDes and Protocol Converter are not powerdown in LPM state. 1 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_RQ	LPM20 Mode Request. This bit is clear by interrupt event. 0 No request to put device in LPM20 mode 1 Request to place device in LPM20 mode.
12-13	—	Reserved
14	—	Reserved
15	—	Reserved
16-21	—	Reserved
22	LPM20_ST	LPM20 Status 0 Device is not attempting to reach LPM20 mode 1 The device is attempting to enter LPM20 mode because POWMGTCSR[LPM20_RQ] is set.
23	P_LPM20_ST	Previous LPM20 Status. It is used by software to know which state the device was in before being wake up. Before software set POWMGTCSR[LPM20_RQ], we expect software W1C [P_LPM20_ST] Bit. 0 Device was not in LPM20 mode before being wake up by interrupt. 1 Device was in LPM20 mode before being wake up by interrupt. The bit is set when LPM20_ST has 1 to 0 transition and it is caused by interrupt. The bit is w1c.

Table 30-27. POWMGTCSR Field Description

Bits	Name	Description
24-28	—	Reserved
29	—	Reserved
30	—	Reserved
31	—	Reserved

30.3.1.3.2 IP Powerdown Exception Control Register n (IPPDEXPCRn)

IPPDEXPCRn, shown in Figure 30-20, provides a mechanism for un-powerdown certain IP during device LPM20 in order to make certain IP a wake-up source. For example,

- Wake on LAN (magic packet) - Requires ethernet controller un-powerdown during device LPM20
- Wake on USB (unplug/plug event) - Requires USB controller un-powerdown during device LPM20
- Wake on eSDHC (card detect event) - Requires eSDHC controller un-powerdown during device LPM20
- Wake on GPIO - Requires GPIO controller un-powerdown during device LPM20

Offset 0x140-0x14C Access: Read / Write

	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	MAC1_1	MAC1_2	MAC1_3	MAC1_4	MAC1_5	MAC1_6	—		MAC1_9	MAC1_10	—														eSDHC	GPIO	USB1	USB2	—			
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

Figure 30-20. IP Powerdown Exception Control Register n (IPPDEXPCRn)

The table below describes this register's bit settings.

Table 30-28. IPPDEXPCRn Field Descriptions

Bits	Name	Description
0	MAC1_1	Frame Manager 1 MAC1 powerdown exception 0 Frame Manager 1 MAC1 powerdown during device LPM20 1 Frame Manager 1 MAC1 is not powerdown during device LPM20
1	MAC1_2	Frame Manager 1 MAC2 powerdown exception 0 Frame Manager 1 MAC2 powerdown during device LPM20 1 Frame Manager 1 MAC2 is not powerdown during device LPM20
2	MAC1_3	Frame Manager 1 MAC3 powerdown exception 0 Frame Manager 1 MAC3 powerdown during device LPM20 1 Frame Manager 1 MAC3 is not powerdown during device LPM20
3	MAC1_4	Frame Manager 1 MAC4 powerdown exception 0 Frame Manager 1 MAC4 powerdown during device LPM20 1 Frame Manager 1 MAC4 is not powerdown during device LPM20

Table 30-28. IPPDEXPCRn Field Descriptions (continued)

Bits	Name	Description
4	MAC1_5	Frame Manager 1 MAC5 powerdown exception 0 Frame Manager 1 MAC5 powerdown during device LPM20 1 Frame Manager 1 MAC5 is not powerdown during device LPM20
6-7	—	Reserved
8	MAC1_9	Frame Manager 1 MAC9 powerdown exception 0 Frame Manager 1 MAC9 powerdown during device LPM20 1 Frame Manager 1 MAC9 is not powerdown during device LPM20
9	MAC1_10	Frame Manager 1 MAC10 powerdown exception 0 Frame Manager 1 MAC10 powerdown during device LPM20 1 Frame Manager 1 MAC10 is not powerdown during device LPM20
10-11	—	Reserved
12-23	—	Reserved
24	eSDHC	eSDHC powerdown exception 0 eSDHC powerdown during device LPM20 1 eSDHC is not powerdown during device LPM20
25	GPIO	GPIO powerdown exception 0 GPIO powerdown during device LPM20 1 GPIO is not powerdown during device LPM20
26	USB1	USB1 powerdown exception 0 USB Controller 1 powerdown during device LPM20 1 USB Controller 1 is not powerdown during device LPM20
27	USB2	USB2 powerdown exception 0 USB Controller 2 powerdown during device LPM20 1 USB Controller 2 is not powerdown during device LPM20
28-31	—	Reserved

30.3.1.4 Interrupt Mask Registers

30.3.1.4.1 Thread Power Management Interrupt Mask Register (TPMIMR0-TPMIMR3)

This register is used for masking pending thread interrupt signalled to the RCPM from MPIC as a means for waking up from a lower power state.

Address TPMIMR3:0x150,TPMIMR2:0x154,TPMIMR1:0x158,TPMIMR0:0x15C Access: User read/write

	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	—																								TIM7	TIM6	TIM5	TIM4	TIM3	TIM2	TIM1	TIM0
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

Figure 30-21. Thread Power Management Interrupt Mask Register (TPMIMR)

The table below describes this register's bit settings.

Table 30-29. TPMIMR Field Descriptions

Bits	Name	Description
0–23	—	Reserved
24	TIM7	Thread Interrupt Power Management Wake Up Mask. 0 Pending thread interrupt cause wake up from a low power mode 1 Pending thread interrupt are masked as a wake up condition
25	TIM6	Thread Interrupt Power Management Wake Up Mask. 0 Pending thread interrupt cause wake up from a low power mode 1 Pending thread interrupt are masked as a wake up condition
26	TIM5	Thread Interrupt Power Management Wake Up Mask. 0 Pending thread interrupt cause wake up from a low power mode 1 Pending thread interrupt are masked as a wake up condition
27	TIM4	Thread Interrupt Power Management Wake Up Mask. 0 Pending thread interrupt cause wake up from a low power mode 1 Pending thread interrupt are masked as a wake up condition
28	TIM3	Thread Interrupt Power Management Wake Up Mask. 0 Pending thread interrupt cause wake up from a low power mode 1 Pending thread interrupt are masked as a wake up condition
29	TIM2	Thread Interrupt Power Management Wake Up Mask. 0 Pending thread interrupt cause wake up from a low power mode 1 Pending thread interrupt are masked as a wake up condition
30	TIM1	Thread Interrupt Power Management Wake Up Mask. 0 Pending thread interrupt cause wake up from a low power mode 1 Pending thread interrupt are masked as a wake up condition
31	TIM0	Thread Interrupt Power Management Wake Up Mask. 0 Pending thread interrupt cause wake up from a low power mode 1 Pending thread interrupt are masked as a wake up condition

30.3.1.4.2 Thread Power Management Critical Interrupt Mask Register (TPMCIMR0-TPMCIMR3)

This register is used for masking pending thread critical interrupt signalled to the RCPM from MPIC as a means for waking up from a lower power state.

Address TPMCIMR3:0x160,TPMCIMR2:0x164,TPMCIMR1:0x168, TPMCIMR0:0x16C — Access: User read/write

	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	—																								TCIM7	TCIM6	TCIM5	TCIM4	TCIM3	TCIM2	TCIM1	TCIM0
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

Figure 30-22. Thread Power Management Critical Interrupt Mask Register (TPMCIMR)

The table below describes this register's bit settings.

Table 30-30. TPMCIMR Field Descriptions

Bits	Name	Description
0–23	—	Reserved
24	TCIM7	Thread Critical Interrupt Power Management Wake Up Mask. 0 Pending thread critical interrupt cause wake up from a low power mode 1 Pending thread critcal interrupt are masked as a wake up condition
25	TCIM6	Thread Critical Interrupt Power Management Wake Up Mask. 0 Pending thread critical interrupt cause wake up from a low power mode 1 Pending thread critcal interrupt are masked as a wake up condition
26	TCIM5	Thread Critical Interrupt Power Management Wake Up Mask. 0 Pending thread critical interrupt cause wake up from a low power mode 1 Pending thread critcal interrupt are masked as a wake up condition
27	TCIM4	Thread Critical Interrupt Power Management Wake Up Mask. 0 Pending thread critical interrupt cause wake up from a low power mode 1 Pending thread critcal interrupt are masked as a wake up condition
28	TCIM3	Thread Critical Interrupt Power Management Wake Up Mask. 0 Pending thread critical interrupt cause wake up from a low power mode 1 Pending thread critcal interrupt are masked as a wake up condition
29	TCIM2	Thread Critical Interrupt Power Management Wake Up Mask. 0 Pending thread critical interrupt cause wake up from a low power mode 1 Pending thread critcal interrupt are masked as a wake up condition
30	TCIM1	Thread Critical Interrupt Power Management Wake Up Mask. 0 Pending thread critical interrupt cause wake up from a low power mode 1 Pending thread critcal interrupt are masked as a wake up condition
31	TCIM0	Thread Critical Interrupt Power Management Wake Up Mask. 0 Pending thread critical interrupt cause wake up from a low power mode 1 Pending thread critcal interrupt are masked as a wake up condition

30.3.1.4.3 Thread Power Management Machine Check Mask Register (TPMMCMR0-TPMMCMR3)

This register is used for masking pending core machine check signalled to the PQ4 COP from MPIC as a means for waking up from a lower power state.

Address TPMMCMR3:0x170,TPMMCMR2:0x174,TPMMCM1:0x178, TPMMCMR0:0x17C — Access: User read/write

	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	—																								TMCM7	TMCM6	TMCM5	TMCM4	TMCM3	TMCM2	TMCM1	TMCM0
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

Figure 30-23. Thread Power Management Machine Check Mask Register (TPMMCMR)

The table below describes this register's bit settings.

Table 30-31. TPMMCMR Field Descriptions

Bits	Name	Description
0–23	—	Reserved
24	TMCM7	Thread Machine Check Power Management Wake Up Mask. 0 Pending thread machine check cause wake up from a low power mode 1 Pending thread machine check are masked as a wake up condition
25	TMCM6	Thread Machine Check Power Management Wake Up Mask. 0 Pending thread machine check cause wake up from a low power mode 1 Pending thread machine check are masked as a wake up condition
26	TMCM5	Thread Machine Check Power Management Wake Up Mask. 0 Pending thread machine check cause wake up from a low power mode 1 Pending thread machine check are masked as a wake up condition
27	TMCM4	Thread Machine Check Power Management Wake Up Mask. 0 Pending thread machine check cause wake up from a low power mode 1 Pending thread machine check are masked as a wake up condition
28	TMCM3	Thread Machine Check Power Management Wake Up Mask. 0 Pending thread machine check cause wake up from a low power mode 1 Pending thread machine check are masked as a wake up condition
29	TMCM2	Thread Machine Check Power Management Wake Up Mask. 0 Pending thread machine check cause wake up from a low power mode 1 Pending thread machine check are masked as a wake up condition
30	TMCM1	Thread Machine Check Power Management Wake Up Mask. 0 Pending thread machine check cause wake up from a low power mode 1 Pending thread machine check are masked as a wake up condition
31	TMCM0	Thread Machine Check Power Management Wake Up Mask. 0 Pending thread machine check cause wake up from a low power mode 1 Pending thread machine check are masked as a wake up condition

30.3.1.4.4 Thread Power Management NMI Mask Register (TPMNMIMR0-TPMNMIMR3)

This register is used for masking pending thread NMI signalled to the RCPM from MPIC as a means for waking up from a lower power state.

Address TPMNMIMR3:0x180,TPMNMIMR2:0x184,TPMNMIMR1:0x188, TPMNMIMR0:0x18C — Access: User read/write

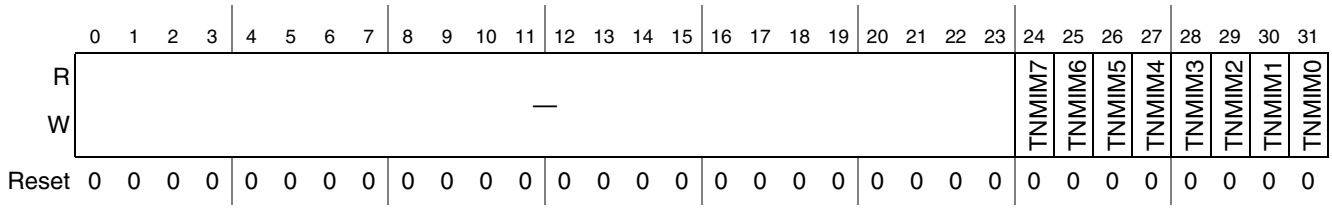


Figure 30-24. Thread Power Management NMI Mask Register (TPMNMIMR)

The table below describes this register's bit settings.

Table 30-32. TPMNMIMR Field Descriptions

Bits	Name	Description
0–23	—	Reserved
24	TNMIM7	Thread NMI Power Management Wake Up Mask. 0 Pending thread NMI cause wake up from a low power mode 1 Pending thread NMI are masked as a wake up condition
25	TNMIM6	Thread NMI Power Management Wake Up Mask. 0 Pending thread NMI cause wake up from a low power mode 1 Pending thread NMI are masked as a wake up condition
26	TNMIM5	Thread NMI Power Management Wake Up Mask. 0 Pending thread NMI cause wake up from a low power mode 1 Pending thread NMI are masked as a wake up condition
27	TNMIM4	Thread NMI Power Management Wake Up Mask. 0 Pending thread NMI cause wake up from a low power mode 1 Pending thread NMI are masked as a wake up condition
28	TNMIM3	Thread NMI Power Management Wake Up Mask. 0 Pending thread NMI cause wake up from a low power mode 1 Pending thread NMI are masked as a wake up condition
29	TNMIM2	Thread NMI Power Management Wake Up Mask. 0 Pending thread NMI cause wake up from a low power mode 1 Pending thread NMI are masked as a wake up condition
30	TNMIM1	Thread NMI Power Management Wake Up Mask. 0 Pending thread NMI cause wake up from a low power mode 1 Pending thread NMI are masked as a wake up condition
31	TNMIM0	Thread NMI Power Management Wake Up Mask. 0 Pending thread NMI cause wake up from a low power mode 1 Pending thread NMI are masked as a wake up condition

30.3.1.4.5 Thread Machine Check Mask Control Register (TMCPMASKCR)

TMCPMASKCR, shown in Figure 30-25, controls the bahavior if thread machine check output is sent to thread concentrator.

Address TMCPMASKCR3:0x190,TMCPMASKCR2: 0x194,
TMCPMASKCR1: 0x198, TMCPMASKCR0: 0x19C

Access: User read/write

	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																									TMCPMASK7	TMCPMASK6	TMCPMASK5	TMCPMASK4	TMCPMASK3	TMCPMASK2	TMCPMASK1	TMCPMASK0
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

Figure 30-25. Thread Machine Check Mask Control Register (TMCPMASKCR)

The table below describes this register's bit settings.

Table 30-33. TMCPMASKCR Field Descriptions

Bits	Name	Description
0–23	—	Reserved
24	TMCPMASK7	Mask thread machine check output 0 Thread machine check output sent to thread concentrator. 1 Thread machine check output is not sent to thread concentrator.
25	TMCPMASK6	Mask thread machine check output 0 Thread machine check output sent to thread concentrator. 1 Thread machine check output is not sent to thread concentrator.
26	TMCPMASK5	Mask thread machine check output 0 Thread machine check output sent to thread concentrator. 1 Thread machine check output is not sent to thread concentrator.
27	TMCPMASK4	Mask thread machine check output 0 Thread machine check output sent to thread concentrator. 1 Thread machine check output is not sent to thread concentrator.
28	TMCPMASK3	Mask thread machine check output 0 Thread machine check output sent to thread concentrator. 1 Thread machine check output is not sent to thread concentrator.
29	TMCPMASK2	Mask thread machine check output 0 Thread machine check output sent to thread concentrator. 1 Thread machine check output is not sent to thread concentrator.
30	TMCPMASK1	Mask thread machine check output 0 Thread machine check output sent to thread concentrator. 1 Thread machine check output is not sent to thread concentrator.
31	TMCPMASK0	Mask thread machine check output 0 Thread machine check output sent to thread concentrator. 1 Thread machine check output is not sent to thread concentrator.

30.3.1.5 Physical Core Time Base Register

30.3.1.5.1 Physical Core Timebase Enable Register (PCTBENR0-PCTBENR1)

PCTBENR, shown in Figure 30-26, provides a mechanism for enabling clocks to any core timebases on the device.

Address 0x1A0 Access: User read/write

	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																													PCTBEN3	PCTBEN2	PCTBEN1	PCTBEN0
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

Figure 30-26. Physical Core Timebase Enable Register (PCTBENR)

The table below describes this register's bit settings.

Table 30-34. PCTBENR Field Descriptions

Bits	Name	Description
0–27	—	Reserved
28	PCTBEN3	Core Timebase Enable. 0 Selected Core Timebase is disabled 1 Selected Core Timebase is enabled
29	PCTBEN2	Core Timebase Enable. 0 Selected Core Timebase is disabled 1 Selected Core Timebase is enabled
30	PCTBEN1	Core Timebase Enable. 0 Selected Core Timebase is disabled 1 Selected Core Timebase is enabled
31	PCTBEN0	Core Timebase Enable. 0 Selected Core Timebase is disabled 1 Selected Core Timebase is enabled

30.3.1.5.2 Physical Core Timebase Clock Select Register (PCTBCKSELR0-PCTBCKSELR1)

PCTBCKSELR, shown in Figure 30-27, selects the clock source for each core's timebase.

Address 0x1A4 Access: User read/write

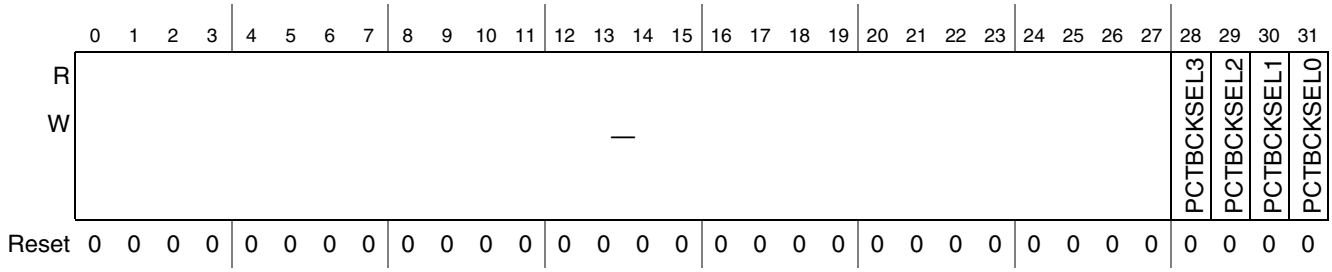


Figure 30-27. Physical Core Timebase Clock Select Register (PCTBCKSELR)

The table below describes this register's bit settings.

Table 30-35. PCTBCKSELRL Field Descriptions

Bits	Name	Description
0–27	—	Reserved
28	PCTBCKSEL3	Core Timebase Input Clock Select. 0 Core Timebase is clocked withPlatform Clock /16 1 Core Timebase is clocked with the RTC signal
29	PCTBCKSEL2	Core Timebase Input Clock Select. 0 Core Timebase is clocked withPlatform Clock /16 1 Core Timebase is clocked with the RTC signal
30	PCTBCKSEL1	Core Timebase Input Clock Select. 0 Core Timebase is clocked with Platform Clock /16 1 Core Timebase is clocked with the RTC signal
31	PCTBCKSEL0	Core Timebase Input Clock Select. 0 Core Timebase is clocked withPlatform Clock /16 1 Core Timebase is clocked with the RTC signal

30.3.1.5.3 Timebase Clock Divider Register (TBCLKDIVR)

TBCLKDIVR, shown in Figure 30-27, selects the clock divider for each core's timebase.

Address 0x1A8 Access: User read/write

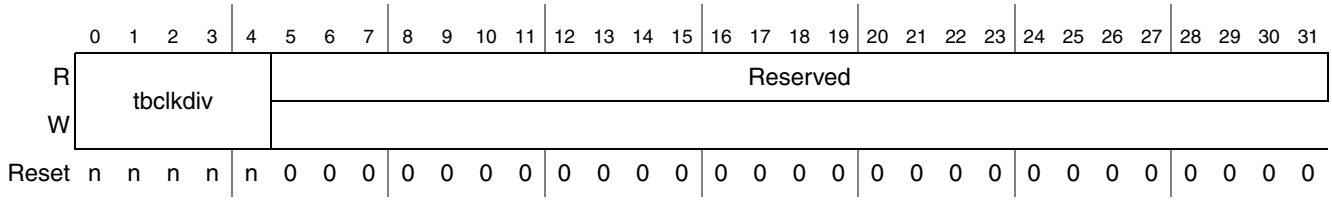


Figure 30-28. Timebase Clock Divider Register (TBCLKDIVR)

The table below describes this register's bit settings.

Table 30-36. TBCLKDIVR Field Descriptions

Bits	Name	Description
0:4	tbclkdiv	Timebase clock divider configuration 0x00 : 1/16 0x02 : 1/8 0x06 : 1/24 0x08 : 1/32
5:31	—	Reserved

30.3.1.5.4 Thread Timebase Halt Control Register (TTBHLTCR0)

TTBHLTCR, shown in Figure 30-27, defines the time based behavior when thread is in halted state. The feature provides user more flexible timebase control in multi-thread programming debug.

NOTE

Proper use of TTBHLTCR requires that all threads within a physical core must be programmed with the same value.

Address TTBHLTCR0:0x1BC Access: User read/write

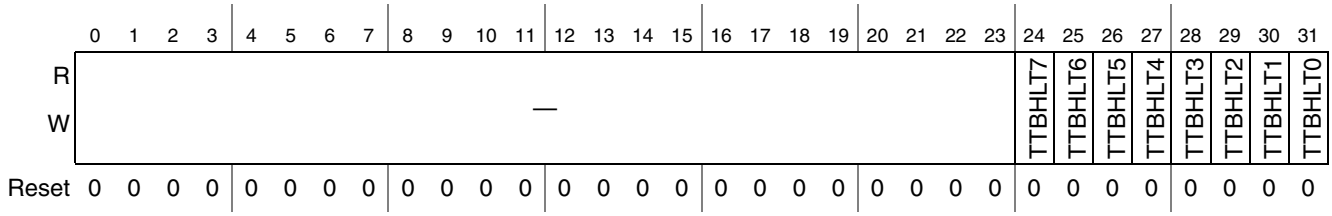


Figure 30-29. Thread Timebase Halt Control Register (TTBHLTCR)

The table below describes this register's bit settings.

Table 30-37. TTBHLTCR Field Descriptions

Bits	Name	Description
0–23	—	Reserved
24	TTBHLT7	Core Timebase Behavior during thread halted 0 Core Timebase is stopped when thread is halted 1 Core Timebase is not stopped when thread is halted
25	TTBHLT6	Core Timebase Behavior during thread halted 0 Core Timebase is stopped when thread is halted 1 Core Timebase is not stopped when thread is halted
26	TTBHLT5	Core Timebase Behavior during thread halted 0 Core Timebase is stopped when thread is halted 1 Core Timebase is not stopped when thread is halted
27	TTBHLT4	Core Timebase Behavior during thread halted 0 Core Timebase is stopped when thread is halted 1 Core Timebase is not stopped when thread is halted

Table 30-37. TTBHLTCR Field Descriptions (continued)

Bits	Name	Description
28	TTBHLT3	Core Timebase Behavior during thread halted 0 Core Timebase is stopped when thread is halted 1 Core Timebase is not stopped when thread is halted
29	TTBHLT2	Core Timebase Behavior during thread halted 0 Core Timebase is stopped when thread is halted 1 Core Timebase is not stopped when thread is halted
30	TTBHLT1	Core Timebase Behavior during thread halted 0 Core Timebase is stopped when thread is halted 1 Core Timebase is not stopped when thread is halted
31	TTBHLT0	Core Timebase Behavior during thread halted 0 Core Timebase is stopped when thread is halted 1 Core Timebase is not stopped when thread is halted

30.3.1.6 Cluster Power Managment Control/Status Register

30.3.1.6.1 Cluster PCL10 Status Register (CLPCL10SR)

This is one of a set of cluster power management registers. This register is used for reporting *PCL10 status* per cluster.

Offset 0x1C0 Access: Read Only

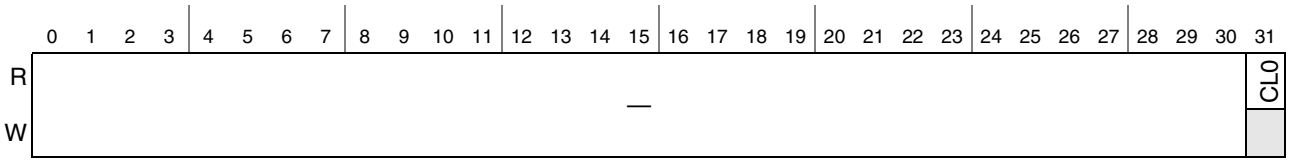


Figure 30-30. Cluster PCL10 Status Register (CLPCL10SR)

The table below describes this register's bit settings.

Table 30-38. CLPCL10SR Field Descriptions

Bits	Name	Description
0–30	—	Reserved
31	CL0	Cluster 0 PCL10 Status. 0 Cluster 0 is not in the PCL10 mode 1 Cluster 0 is in the PCL10 mode

30.3.1.6.2 Cluster PCL10 Set Control Register (CLPCL10SETR0-CLPCL10SETR3)

This is one of a set of Cluster Power Management Registers. This register is used for requesting that a cluster be placed in the *PCL10* state. The true value read from the register means there is a pending cluster PCL10 request.

Address 0x1C4 Access: Read / Write -one-set

	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	CL_PCL10_REQ																															
W	w1s																															
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

Figure 30-31. Cluster PCL10 Set Control Register (CLPCL10SETRn)

The table below describes this register's bit settings.

Table 30-39. CLPCL10SETRn Field Descriptions

Bits	Name	Description
0-31	CL_PCL10_RQ	Write one to trigger cluster PCL10 request. This bit is clear by interrupt event or write-one event on CLPCL10CLRRn register. xxxxxxxx_xxxxxxxx_xxxxxxxx_xxxxxxx1 Request to put thread 0 in PH30 state. xxxxxxxx_xxxxxxxx_xxxxxxxx_xxxxxx1x Request to put thread 1 in PH30 state. xxxxxxxx_xxxxxxxx_xxxxxxxx_xxxxx1xx Request to put thread 2 in PH30 state. xxxxxxxx_xxxxxxxx_xxxxxxxx_xxxx1xxx Request to put thread 3 in PH30 state. xxxxxxxx_xxxxxxxx_xxxxxxxx_xxx1xxxx Request to put thread 4 in PH30 state. xxxxxxxx_xxxxxxxx_xxxxxxxx_xx1xxxxx Request to put thread 5 in PH30 state. xxxxxxxx_xxxxxxxx_xxxxxxxx_x1xxxxxx Request to put thread 6 in PH30 state. xxxxxxxx_xxxxxxxx_xxxxxxxx_1xxxxxxx Request to put thread 7 in PH30 state.

30.3.1.6.3 Cluster PCL10 Clear Control Register (CLPCL10CLRR)

This is one of a set of Cluster Power Management Registers. This register is used for waking up the cluster be placed in the *PCL10* state. The true value read from the register only means there is a pending cluster PCL10 request.

Address 0x1C8 Access: Read / Write -one-clear

	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	CL_PCL10_REQ																															
W	w1c																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 30-32. Cluster PCL10 Clear Control Register (CLPCL10CLRR)

The table below describes this register's bit settings.

Table 30-40. CLPCL10CLRR Field Descriptions

Bits	Name	Description
0-31	CL_PCL10_RQ	Write one to cancel cluster PCL10 request. The bit is set by write-one-event to CLPCL10SETR register. This bit is also clear by interrupt event. The read true value of the register bit means there is pending cluster PCL10 request for that cluster. xxxxxxxx_xxxxxxxx_xxxxxxxx_xxxxxxx1 cluster 0 is receiving a CPL10 request from CLPCL10SETR

30.3.1.6.4 Cluster PCL10 Previous Status Register (CLPCL10PSR)

This is one of a set of Cluster Power Management Registers. This register is used for reporting previous *PCL10 status* per thread. It is used by software to know which power saving state it was in before wake up by interrupt.

Address 0x1CC Access: Read Only

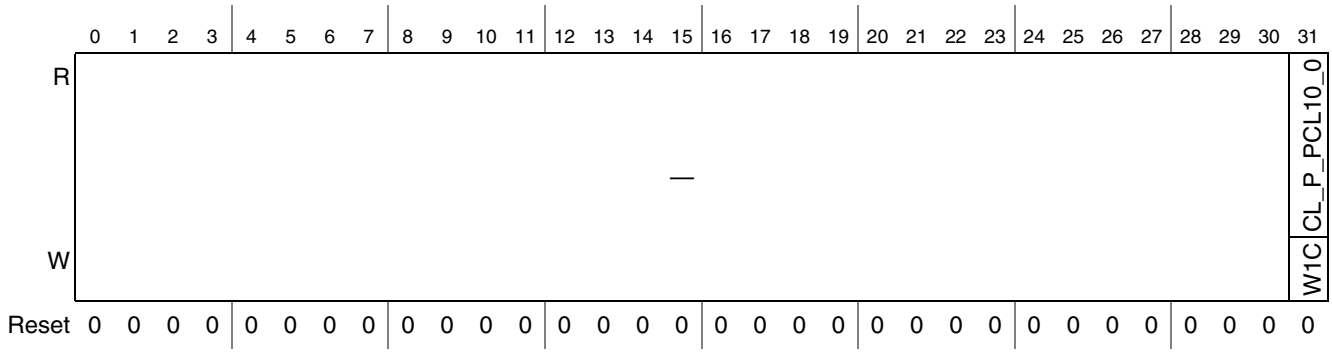


Figure 30-33. Cluster PCL10 Previous Status Register (CLPCL10PSR)

The table below describes this register's bit settings.

Table 30-41. CLPCL10PSR Field Description

Bits	Name	Description
0–30	—	Reserved
31	CL_P_PCL10_0	Cluster PCL10 previous status. The bit is set when corresponding CLPCL10SR bit has 1 to 0 transition and it is caused by interrupt. The bit is w1c. 0 Cluster was not in the PCL10 mode before wake up by interrupt. 1 Cluster was in the PCL10 mode before wake up by interrupt.

30.4 Functional Description

30.4.1 Thread/Core Power Management

30.4.1.1 Overview

The RCPM supports minimizing the power consumption through software controlled power management states—PH10 and PH15/PH20/PH30.

The RCPM handshakes with thread to halt the instruction sequence and handshakes with CCF to gracefully stop the internal CoreNet network.

The RCPM allows several wake-up event sources to exit low power mode (internal timer, internal and external interrupts). The wake-up events are mapped to OpenPIC interrupts to generate a wake-up interrupt to the RCPM.

30.4.1.2 Thread PH10 Operation

30.4.1.2.1 Thread PH10 Operation Entry

PH10 operation can be triggered by the following mechanisms:

- EPU event trigger via CGACRE.
- Device event trigger via CGACRD.
- Setting corresponding bit(s) in RCPM TPH10SETR via software.

30.4.1.2.2 Thread PH10 Operation Exit

Exit from PH10 state (wake up) is triggered by following:

- Corresponding bit(s) in RCPM TPH10CLRR are set.
- Thread IRQ interrupt without RCPM TPMIMR masking.
- Thread critical interrupt without RCPM TPMCIMR masking.
- Thread debug interrupt
- Core time based interrupt
- Wake on LAN - magic packet. If IP is present and programmed
- Wake on USB - unplug/plug. If IP is present and programmed
- Wake on eSDHC - card detect. If IP is present and programmed
- Wake on GPIO. If IP is present and programmed
- Core unrecoverable failure
- Core Hard reset event

30.4.1.2.3 Thread PH10 Operation Sequence

The sequence of PH10 operation is described below:

- The RCPM core power management control logic takes PH10 request, and asks the core to halt thread (by asserting thrd_pm_halt signal). The power management halt request is recorded in core SBSR0[PM_HALT].
- When core receives power management halt request, it will stop decoding instruction, unless there is a *lmw* or *stmw* in progress, in which case it will stop after all of the micro-ops have been decoded. Next, core will wait for the completion buffer to drain. In order to make sure that all load/store operation can been seen globally, the sequencer logic will jam an *msync* instruction and wait for it to deallocate.
- After that, core declares its halted state and asserts thrd_halted signal to RCPM. At this point, thread is in PH10 state.

Core in PH10 state can still handle snoop requests.

30.4.1.3 PH15 Operation

30.4.1.3.1 PH15 Operation Entry

PH15 operation can be triggered by the following mechanism:

- Setting corresponding bit(s) in RCPM PCPH15SETR via software.

Setting the following two programming registers will cause hardware trigger PH15 like operation to bring the core in stopped state gracefully. In these two cases, PH15 mode is used to gracefully stop core to prevent CoreNet hang.

- Setting corresponding bit(s) in DCFG COREDISR to disable core(s).
- Setting corresponding bit(s) in MPIC PIR to initiate core warm reset.

30.4.1.3.2 PH15 Operation Exit

Exit from PH15 state (wake up) is triggered by the following:

- Corresponding bit(s) in RCPM PCPH15CLRR are set.
- Thread IRQ interrupt without RCPM TPMIMR masking.
- Thread critical interrupt without RCPM TPMCIMR masking.
- Core time based interrupt
- Wake on LAN - magic packet. If IP is present and programmed
- Wake on USB - unplug/plug. If IP is present and programmed
- Wake on eSDHC - card detect. If IP is present and programmed
- Wake on GPIO. If IP is present and programmed
- Core unrecoverable failure
- Core Hard reset event
- A core disabled by COREDISR can only exit by device hard reset or power on reset.

30.4.1.3.3 PH15 Operation Sequence

The sequence of PH15 operation is described below:

- The RCPM core power management control logic takes PH15 request and start PH15 sequence. The first part of PH15 sequence is the same as PH10 to bring the core in halted state.
- After core is halted, RCPM sends core-specific stop request to CCM (ccm_core_stop). It is to ask CCM start to drain the snoop transaction on the fly and to reach a transaction boundary which we can stop.
- CCM puts the processor (port) into “No Snoop” mode and stops sending any new snoops to the processor. Note that there might be snoops (AIns) already enroute to the processor which are not stopped.
- After all the AIns have been delivered to the Async FIFO, CCM drives the stop signal with proper core IDs to the AIn FIFO.
- After all the AIns have been picked up by the core, the AIn FIFO delivers the stop signal to the processor (via core_stop signal).
- The processor completes all activities related to all transactions snooped by the core. After all such activity is complete, the core drives core_stopped signal out to AOut Async FIFO.
- At this time, the AOut FIFO should be empty. The Async FIFO forwards the stopped signal to CCM.
- The CCM ensures that all activity from the core related to snoops delivered to the core has been received into CCM, CCM forwards the ccm_core_stopped signal to RCPM. This ensures that CCM has extracted all response transactions from the various FIFOs that may have been written to by the core. At this point, core is in PH15 state.

There is no need to handshake between core power management control logic and SOC power management control logic in PH15 operation since it is assumed that software flushes core L1/L2 cache data before triggering PH15 operation. For debug event, debugger need to do the necessary work to make sure the coherence state is still preserved. The power management stop request is recorded in core SBSR0[PM_STOP].

30.4.1.4 Core PH20 (PGSR) Operation

30.4.1.4.1 Core PH20 (PGSR) Operation Entry

Core PH20 (PGSR) operation can be triggered by the following mechanism:

- Setting corresponding bit(s) in RCPM PCPH20SETR via software.

The corresponding bit(s) in RCPM PCPH20SETR should be set when core is out of reset as indicated by CRSTSR status register in DCFG. If PCPH20SETR is set while the core(s) are still in reset then the corresponding core would enter PH20 after reaching PH15 in the core hreset sequence, see section Section 30.4.1.3.1, “PH15 Operation Entry on core PH15 operation entry.

30.4.1.4.2 Core PH20 (PGSR) Operation Exit

Core napping can be waked up in the following cases:

- Corresponding bit(s) in RCPM PCPH20CLRR are set.
- Core time based interrupt

- Wake on LAN - magic packet. If IP is present and programmed
- Wake on USB - unplug/plug. If IP is present and programmed
- Wake on eSDHC - card detect. If IP is present and programmed
- Wake on GPIO. If IP is present and programmed
- Core unrecoverable failure
- Core Hard reset event

30.4.1.4.3 Core PH20 (PGSR) Operation Sequence

The sequence of core PH20 (PGSR) operation is described below:

- The RCPM core power management control logic takes core PH20 (PGSR) request and start core PH20 (PGSR) sequence. In the first part of core PH20 (PGSR) sequence, RCPM requests thread to enter PH10 (doze) halted state.
- After core is halted and thread is in PH10, RCPM then requests core to enter PH15 (nap) stopped state.
- After core is stopped, RCPM asserts core_pg_sr and requests the core to initiate power gating with state retention sequence.
- Once core is power gated with state retention, core declares its PGSR state and asserts core_ph20 signal to RCPM.

30.4.1.5 Core PH30 (PG) Operation

30.4.1.5.1 Core PH30 (PG) Operation Entry

Core PH30 (PG) operation can be triggered by the following mechanism:

- Setting corresponding bit(s) in RCPM PCPH30SETR via software.

The corresponding bit(s) in RCPM PCPH30SETR should be set when core is out of reset as indicated by CRSTSR status register in DCFG. If PCPH30SETR is set while the core(s) are still in reset then the corresponding core would enter PH20 the PH30 after reaching PH15 in the core hreset sequence, see section Section 30.4.1.3.1, “PH15 Operation Entry on core PH15 operation entry.

30.4.1.5.2 Core PH30 (PG) Operation Exit

Core napping can be waked up in the following cases:

Corresponding bit(s) in RCPM PCPH30CLRR are set if the core has not yet reached the PH30 state.

- Core Hard reset event

30.4.1.5.3 Core PH30 (PG) Operation Sequence

The sequence of core PH30 (PG) operation is described below:

- The RCPM core power management control logic takes core PH30 (PG) request and start core PH30 (PG) sequence. In the first part of core PH30 (PG) sequence, RCPM requests thread to enter PH10 (doze) halted state.

- After core is halted and thread is in PH10, RCPM then requests core to enter PH15 (nap) stopped state.
- After core is stopped, RCPM asserts core_pg_sr and requests the core to initiate power gating with state retention sequence.
- Once core is power gated with state retention, core declares its PGSR state and asserts core_ph20 signal. RCPM then asserts core_off and requests the core to initiate full power gating sequence.

30.4.2 Device Power Management

30.4.2.1 Overview

RCPM supports minimizing the power consumption through software controlled device power management state (LPM20).

RCPM can handshake with various IP at SOC level to gracefully stop the IP traffic and direct the memory controller to put DDR into self-refresh mode (if enabled).

RCPM allows several wake-up events source to exit low power mode (core timebase, internal and external interrupts). The wake-up events are mapped to OpenPIC interrupts to generate a wake-up interrupt to the RCPM.

30.4.2.2 LPM20 Operation

LPM20 and device debug softstop are two different manners for gating off clocks to the entire chip. Both are recoverable but they are used for different purposes. LPM20 is a chip low power state. Device debug softstop is a debug state which we can utilize to do LSRL debug scan. Prior entering LPM20 , software is expected to quiesce all external devices/internal IPs through configuration. RCPM hardware is designed to implement sanity check for IPs idle status. Core timebase is turned off when the chip is in LPM20 state.

30.4.2.2.1 LPM20 Operation Entry

LPM20 operation can be triggered by the following mechanism:

- Setting RCPM POWMGTCSR[LPM20_RQ] to 1 via software.

30.4.2.2.2 LPM20 Operation Exit

Exit from LPM20 mode (wake up) can be triggered by the following:

- Core time based interrupt
- Wake on LAN - magic packet. If IP programmed
- Wake on USB - unplug/plug (via MPIC). If IP programmed
- Wake on eSDHC - card detect (via MPIC). If IP and programmed
- Wake on GPIO. If IP programmed
- Core unrecoverable failure
- Core Hard reset event

30.4.2.2.3 LPM20 Operation Sequence

The sequence of device LPM20 operation is described below:

- The RCPM device power management control logic takes device LPM20 request and start device LPM20 sequence.
- RCPM watches the IP idle status and confirms that IPs reports idle.
- RCPM asserts stop request to all master IPs in device, and waits for the stop acknowledgement from all master IPs. After master IP receives stop request, it will stop mastering any new transaction. After its all transaction tenures are completed (request, response, data tenure), it will assert stop acknowledgement back to RCPM.
- RCPM asserts stop request to the IPs which can be served as both master and target device, and wait for the stop acknowledgement from all master/target IPs. During this process, the deadlock case could happen. If deadlock case has been detected by RCPM, RCPM will de-assert the stop request to these IPs and let the transaction goes through to resolve the deadlock.
- RCPM asserts stop request to all target IPs in device, and wait for the stop acknowledgement from all target IPs.
- At this point, the chip is in the LPM20 state. RCPM asserts the power down bus to clock control unit to clock gate all IPs in device except for necessary IPs for wake up. For example, MPIC and USB (if programmed).

30.5 Initialization Information

This section describe RCPM initialization information. RCPM uses both POR reset and Hard reset for hardware initialization. Interrupt sampling, RUNN counter, and Core Group logic are initialized only through POR reset. SOC power management module and Core power management module are initialized through Hard reset. RCPM CCSR/DCSR are reset either by POR reset or Hard reset depending on their functionality. For details, please refer to Section 30.3, “Memory Map and Register Definition”.

30.6 Application Information

30.6.1 Putting the Chip into LPM20 Mode

This section captures the software sequence to place the chip into LPM20 mode.

- Frame manager needs to be initialized before software issues a LPM20 request. Please refer to frame manager document for the initialization procedure.
- Write SNVS.LPPGDR (SM_LP Power Glitch Detector Register) with value 32’h41736166.
- Write SNVS.LPSR (SM_LP Status Register) with value 32’h00000008. It is to clear power supply glitch detection.

30.6.2 Magic Packet Wake Up

To enable magic packet wake up feature in LPM20 mode, the frame manager module associated with the mEMAC receiving magic packet needs to be enabled.

Appendix A
Appendix T2081

A.1 Introduction

The T2081 QorIQ advanced, multicore processor combines four, dual-threaded e6500 Power Architecture® processor cores with high-performance datapath acceleration logic and network and peripheral bus interfaces required for networking, telecom/datacom, wireless infrastructure, and mil/aerospace applications.

This figure shows the major functional units within the chip.

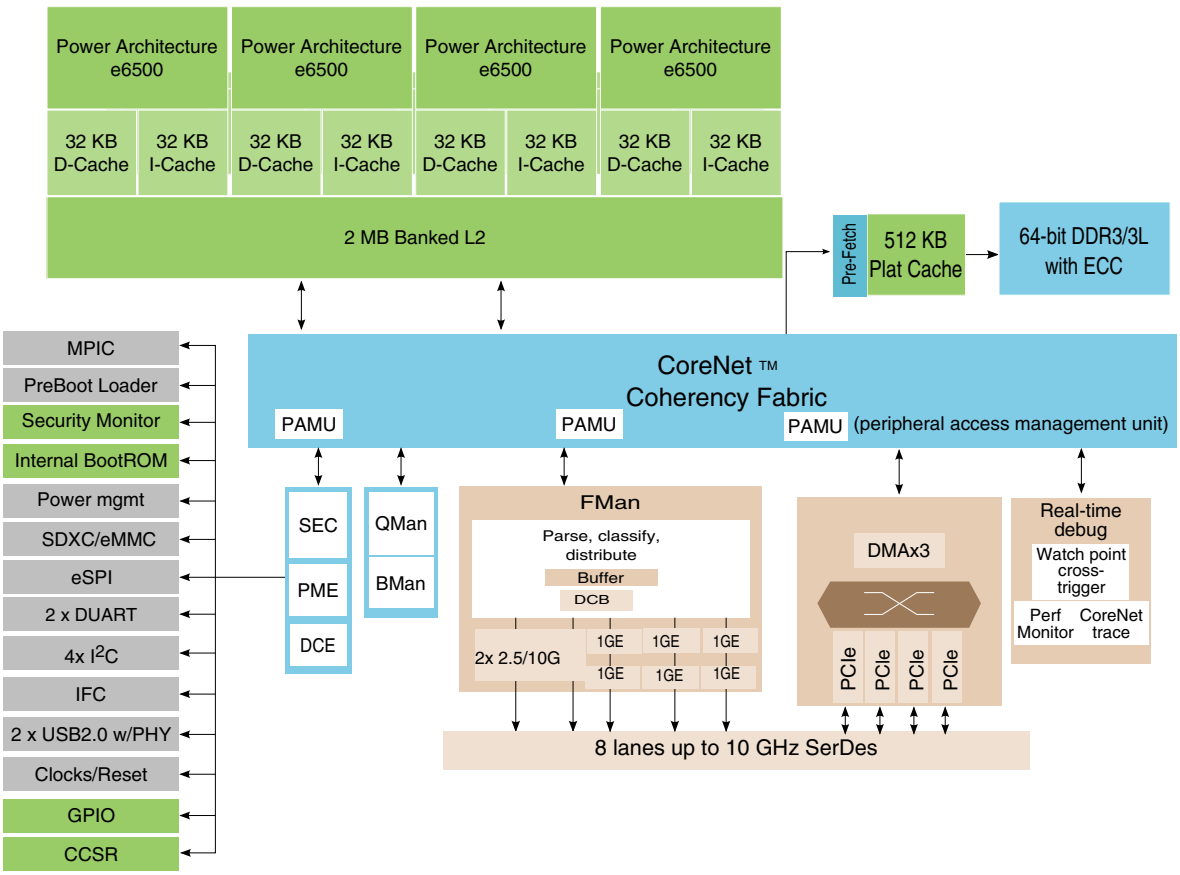


Figure A-1. T2081 block diagram

A.2 Overview of Differences

Table A-1. Comparison between T2080 and T2081

Feature	T2080	T2081
Peripherals		
10G Ethernet Controllers	Up to four with XFI, 10GBase-KR, 10GBase-KX, XAUI, HiGig and HiGig2	Up to two XFI or 10GBase-KR, 10GBase-KX
1G Ethernet Controllers	Up to eight	Up to six
SerDes and Pinout		
Total number of SerDes lanes	16	8
High Speed Serial IO		
SRIO Controller and RapidIO Message Manager	2 + RMan	not supported
SATA Controller	2	not supported
Aurora	supported	not supported
Package	25 x 25mm, 896 pins, 0.8mm pitch	23 x 23mm, 780 pins, 0.8mm pitch, pin compatible with T1042

A.3 T2081 Registers

This section points out the deviation of registers from T2080

Table A-2. Unavailable register bits

Register Name	Bit Number	Description
Device Disable Register 1 (DCFG_DEVDISR1)	24 (RMan)	Set to disable
Device Disable Register 1 (DCFG_DEVDISR1)	16-17 (SATA)	Set to disable
Device Disable Register 3 (DCFG_DEVDISR3)	4-5 (SRIO)	Set to disable
Device Disable Register 5 (DCFG_DEVDISR5)	11 (NAL)	Set to disable

Table A-3. SVR, PCI and RapidIO Device IDs, JTAG ID

	SVR	PCI and RapidIO Device IDs	JTAG ID
T2081 with security	0x 8539_0010	0x0838	018E601D
T2081 without security	0x 8531_0010	0x0839	018E601D

A.4 SerDes Assignments

The following notation conventions are used in the table:

- XFIm indicates XFI (1 lane @10.3125 Gbps), “m” indicates MAC on the Frame Manager. For example, “XFI9” indicates XFI using MAC 9.
- SGMII notation :
  - sgm means SGMII @ 1.25 Gbps where “m” indicates which MAC on the Frame Manager. For example, “SG3” indicates SGMII for MAC 3 at 1.25 Gbps.
  - *sgm* means SGMII @3.125Gbps where “m” indicates which MAC on the Frame Manager. For example, “*SG3*” indicates SGMII for MAC 3 at 3.25 Gbps.
- PCIe notation :
  - PCIem is PCIe @ 5/2.5 Gbps, m indicates the PCIe controller number.
  - *PCIem* is PCIe @ 8/5/2.5 Gbps, m indicates the PCIe controller number.
- Per lane PLL mapping: 1-PLL1, 2-PLL2

SerDes Networking Options:

Table A-4. SerDes

SRDS_PRTCL_S1	A	B	C	D	E	F	G	H	Per lane PLL mapping
6E	XFI9	XFI10	SG1	SG2	PEX4		SG5	SG6	11222222
AA	PEX3				*PEX4*				11111111
BC	PEX3		SG1	SG2	*PEX4*				11111111
C8	PEX3	SG10	*SG1*	*SG2*	PEX4		SG5	SG6	11221111
CA	PEX3	SG10	*SG1*	*SG2*	PEX4	SG4	SG5	SG6	11221111
D6	PEX3	SG10	SG1	SG2	*PEX4*		SG5	SG6	11112211
DE	PEX3				*PEX4*	PEX1	PEX2	SG6	11111111
E0	PEX3				*PEX4*	PEX1	SG5	SG6	11111111
F2	PEX3	SG10	SG1	SG2	*PEX4*	PEX1	PEX2	SG6	11111111
F8	PEX3	SG10	SG1	SG2	PEX4	PEX1	PEX2	SG6	11221111
FA	PEX3	SG10	SG1	SG2	PEX4	PEX1	SG5	SG6	11221111
6C	XFI9	XFI10	SG1	SG2	PEX4				11222222
70	XFI9	XFI10	SG1	SG2	PEX4	SG4	SG5	SG6	11222222

Appendix B
Revision History

B.1 Substantive changes from revision A.2 to revision C.1

Substantive changes from revision A.2 to revision C.1 are as follows:

B.1.1 Overview

Reference	Description
Features summary	Changed to "Four PCI Express controllers (two support PCIe 2.0 and two support PCIe 3.0)".
High-speed peripheral interface complex (HSSI)	One Gen3.0 PCI express controller with SRIOV, One Gen3.0 PCI express controller without SRIOV, and two PXI express Gen3.0 controllers

B.1.2 Memory Map Revision History

Reference	Description

B.1.3 Signals Revision History

No substantive changes

B.1.4 Reset Revision History

Reference	Description
External Signal Descriptions	Removed the duplicate row for CLK_OUT signal.
PLL cluster n general status register (Clocking_PLLC*n*GSR)DDR PLL general status register (Clocking_PLLDGSR)	Added bit 0 KILL and updated the bit width of CFG bits from 25-30 to 23-30
System Control Signals	Added row for LP_TMP_DETECT_B.

B.1.5 Pre-Boot Loader Revision History

Reference	Description
Software Restrictions	Added bullet stating that the Flush command is restricted to CCSR space.
PBL Modes of Operation	Added "Load Pre-Boot initialization words only if default RCW is to be used."
Reserved Address Space Used as Internal PBL Commands	Added "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."

B.1.6 Secure Boot and Trust Architecture Revision History

Reference	Description
System security monitor	Replaced security monitor state diagram with a simplified version.
State definitions	Changed "Start" to "Init" In the table, removed transition from Trusted to Secure. Revised description for the Soft Fail state
Instruction Register (SFP_INGR)	Moved ERR bit from bit 24 to bit 23.
SFP Version Register (SFP_VERSION)	Added new VERSION register.
SM_LP Tamper Detectors Configuration Register (SECMON_LPTDCR)	Added new bit, ET1_EN (bit 22), that controls external low power tamper detection.
Fuse programming	Changed supply voltage to 1.8 V
Debug Challenge Value Register n (SFP_DCVR*n*)	Changed to Write to DCVR 1: c0c0c0c0 Write to DCVR 0: d0d0d0d0 from Write to DCVR 0: c0c0c0c0 Write to DCVR 1: d0d0d0d0 changed the bitfield description of DCVR*n* bits to DCVR1 is the upper part (bits 0-31) and DCVR0 is the lower part (bits 32-63) of the 64-bit debug challenge value.

Table continues on the next page...

Reference	Description
SFP Configuration Register (SFP_SFPCR)	Changed bit SFPWD to R/W.
SM_LP Status Register (SECMON_LPSR)	Added bit ET1D
Signals	Updated the description of TMP_DETECT_B
Signals	Added LP_TMP_DETECT_B
One Time Programmable Master Key n (SFP_OTPMKR*n*)	Removed "The security monitor also reports OTPMK Hamming errors in the SM_HP Status Register (SECMON_HPSR)."
OEM Scratch Pad Fuse Register n (SFP_OSPFR*n*)	Changed "These registers are protected by the OEM write protect bits" to "SFP_OSFPR0 is protected by the OEM write protect bits".
Security Monitor functional description	Removed "The Security Monitor is also responsible for checking the OTPMK's hamming code and passing it to the SEC when the hamming code passes and the state machine is in a Trusted or Secure state"
SM_HP Security Violation Status Register (SECMON_HPSVSR)	Added new bit definition (LP_SEC_VIO) at bit 0.
SM_HP Version ID Register 1 (SECMON_HPVIDR1) and SM_HP Version ID Register 2 (SECMON_HPVIDR2)	Changed register reset values to n. Added IP_ERA bitfield to HPVIDR2.

B.1.7 e6500 Core Integration Revision History

No substantive changes

B.1.8 CoreNet Platform Cache (CPC) Revision History

Reference	Description
Enabling the CPC after Power-On Reset	Revised the steps for enabling CPC after Power-On Reset.
CPC partition allocation register 0 (CPC_CPCPAR0), CPC partition allocation register n (CPC_CPCPAR*n*)	Removed LOADEC from DRDALLOC, DRDPEALLOC, and DRDFEALLOC allocation policies
CPC partition allocation register 0 (CPC_CPCPAR0), CPC partition allocation register n (CPC_CPCPAR*n*)	Changed bits 22, 23 and 25 to reserved. Changed "Data RWITM, WBI,CTM and CTT transactions that miss in the CPC" to "Transactions resulting from a Store operation that missed in a core cache" in the description of DSTALLOC bits. Replaced WCO with "Writes carrying cast-out data from core caches" in the description of WCOALLOC bits." Replaced ReadU transactions by "Decorated Load operations" in the description of READUALLOC bits. Replaced WriteU/notify transaction by "Decorated Store operations" in the description of WRITEUALLOC bit description. Added link to "Decorated Storage". Replaced CoreNet by "Explicitly" in the description of CNSTASHEN bit description."

Table continues on the next page...

Reference	Description
Error CaptureCoreNet Platform Cache (CPC) Memory Map	Removed CPC error attribute register (CPCx_CPCERRATTR)
CPC partition ID register 0 (CPC_CPCPIR0), CPC partition ID register n (CPC_CPCPIR*n*)	Added link to LAW registers

B.1.9 Prefetch Manager (PMAN) Revision History

Reference	Description
PMAN Control register 3 (PMAN_PC3)	Replaced the term “requester” with “source” and “RID” with “SID” in the bitfield description of bits 0-7 and 8-15. Changed the bit name from RIDTE to SIDTE. Replaced the term”Master” by “Source” in the bitfield description of bits 28-29 changed the name of the bit to STTE from MTTE. Changed bits 17-22 to reserved.
PMAN Memory Map/Register Definition	Removed PMAN Capabilities register.
Allocation policy	Modified the description to say "A CPC miss that does not match any of the existing singleton buffers and the operation type is either a Read or a Stash from a core or I/O. No allocation will occur on a cache inhibited Read from a core, nor on a Write or Store type operation from a core or I/O. "

B.1.10 CoreNet Coherency Fabric (CCF) Revision History

Reference	Description
CSDID n Port Mapping Register (CCF_CIDMR*n*)	Corrected the offsets for CCF_CIDMR1-CCF_CIDMR31.

B.1.11 Peripheral Access Management Unit (PAMU) Revision History

No substantive changes

B.1.12 DDR Revision History

Reference	Description
DDR SDRAM timing configuration 5 (DDR_TIMING_CFG_5)	Changed the bitfield description of WODT_OFF to "Write to ODT off for DRAM. This field specifies the number of cycles that the relevant

Table continues on the next page...

Reference	Description
	ODT signal(s) remain asserted for each write transaction. Select 4 clock cycles. The memory controller automatically adds 2 clock cycles when an 8-beat burst is used. Therefore, regardless of burst type selection, fixed 8-beat, fixed 4-beat burst chop, or on-the-fly burst chop mode value of 4 clock cycles should be selected for this field. 3'b100 for any burst type selected."
Memory data path read capture ECC (DDR_CAPTURE_ECC)	Updated description of field ECE. Widened bitfield from 16 to 32 bits.
DDR SDRAM timing configuration 6 (DDR_TIMING_CFG_6)	Added register
DDR Memory Test Pattern n Register (DDR_DDR_MTP)	Added a note to register description: "Memory test read compares data that is written in this register."
DDR Memory Test Control Register (DDR_DDR_MTCR)	Updated MT_EN bitfield note from "The memory test may not be used if memory controller interleaving has been enabled via CS0_CONFIG[INTLV_EN]." to "The memory test may not be used if memory controller interleaving has been enabled via CS0_CONFIG[INTLV_EN] and DDR_MTCR[MT_TRNARND]=0000."
DDR SDRAM timing configuration 0 (DDR_TIMING_CFG_0)	Updated EXT_PRE_PD_EXIT bitfield settings 01 and 10 from "01 (32 clocks), 10 (16 clocks)" to "01 (16 clocks), 10 (32 clocks)".
Memory Interface Signals	Updated MAPAR_ERR_B timing for DDR3 registered DIMMs: "Driven by the registered DDR3 DIMMs 3 DRAM cycles after the parity bit has been driven by the memory controller."
Table 12-93	Added table.
Memory Interface Signals	In MDICn description, changed resistor value from 240-ohm to 237-ohm.
DDR training initialization address (DDR_DDR_INIT_ADDR)	Added to register description: "When the default value is used (that is, address 0x0), all chip selects are considered for the training. If DDR_INIT_ADDR is set to any value other than the default value of address zero, then only the first chip select will be trained. When multiple chip selects are used and DQS/DQ skew is not common between chip selects/ranks, then the default address value of 0x0 is be recommended to obtain the best timing margins."
DDR write leveling control 2 (DDR_DDR_WRLVL_CNTL_2), DDR write leveling control 3 (DDR_DDR_WRLVL_CNTL_3)	Added to register description: **NOTE:** For each field WRLVL_START_n, a setting of 0000 is not recommended; it disables the per-byte write level start time selection. It is recommended to select proper individual delay for each byte lane based on board skews.
DDR Control Driver Register 1 (DDR_DDRCDR_1) and DDR Control Driver Register 2 (DDR_DDRCDR_2)	Changed bitfield settings 010, 100, 110 of bits ODT to reserved. Changed the value of setting 111 from 47 ohms to 46 ohms.
DDR Control Driver Register 1 (DDR_DDRCDR_1)	Updated register description.

B.1.13 Integrated Flash Controller (IFC) Revision History

Reference	Description
NAND asynchronous mode calculating read data window width	In bullet EDO mode NAND, corrected formula from 'tREH - (tREA + tRP) + tRHOH' to 'tREH–(tREA – tRP)+ tRLOH'.
Chip-Select Option Register - GPCM (IFC_CSOR*n*_GPCM)	Updated description of bits [WGETA] and [RGETA].
GPCM event and error status register (IFC_GPCM_EVTER_STAT)	Updated description of bits [TOER] and [ABER].
SRAM buffer	Updated "SRAM buffer layout" figures in subsections.
NOR flash features	Removed the following feature: Burst size in NOR is governed by AXI size and length
Table 13-251	Updated 'Initial Contents' of NAND_FIR0 from 2545_4000h to 2608_4000h .
Table 13-210	Updated tWB as TWBE, tWHR as TWHRE.
Switching to the asynchronous interface	Removed topic "Switching to the asynchronous interface".
Table 13-210	Updated occurences of time-out as time-out (depends uponIFC_NANDCR[FTOCNT]).
Clock control register (IFC_CCR)	Marked bit CCR[FB_IFC_CLK_SEL] as reserved.
General ASIC program operation	Added 1, 2, 3, and 4 in figure to show 4-wait states.
NAND flash features	Changed 72 (8 + 64) to 136(8 + 128) bytes in "At least 72 (8 + 64) bytes of spare region is required for bad block information and ECC bytes, 4-bit BCH" Changed 136(8 + 128) to 264(8 + 256) bytes in "at least 136 (8 +128) bytes of spare region required for bad block information and ECC bytes, 4-bit or 8-bit BCH"
NAND flash features	Changed 2048 to 512 pages in "Configurable block-size constraints to a multiple of 32 pages, up to 2048 pages"
IFC functional description	Removed "It has inherently low performance, because it does not support bursting."
Table 13-2	Updated the descriptions of IFC_WE_B and IFC_WP_B and added alternate signal names and descriptions used in IFC to describe IFC signals in the Signals Overview.

B.1.14 I2C Revision History

No substantive changes

B.1.15 Enhanced Serial Peripheral Interface Revision History

No substantive changes

B.1.16 eSDHC Revision History

Reference	Description
Transfer type register (eSDHC_XFERTYP)	In the Relationship Between Parameters and the Name of the Response Type table, updated the fourth row to add R7 as the response type.
Protocol control register (eSDHC_PROCTL)	Changed bit 28 D3CD to reserved.
SD monitor	Added note "Do not use DAT3 pin as a CD pin."
Card insertion and removal detection	Removed references to DAT3 pin being used for card detection
eSDHC features summary	Removed "and SD_DAT3 pin depending on PROCTL[D3CD]"
Data transfer with Auto CMD12 Enable	Added section.
Soft reset for data not allowed when SD clock is disabled	Added SYSCTL[RSTC] to the restriction.
Host controller capabilities register 2 (eSDHC_HOSTCAPBLT2)	Changed the access for the supported bits to read only.
Host controller capabilities register 2 (eSDHC_HOSTCAPBLT2)	Changed the access for AIS and 64BSBS bits to read only.
System control register when ESDHCCTL[CRS=0] (eSDHC_SYSCTL)	Removed "NOTE: The Base clock frequency is platform_clk/ 2" from SDCLKFS bit description.
System control register when ESDHCCTL[CRS=0] (eSDHC_SYSCTL)System Control Register when ESDHCCTL[CRS=1] (eSDHC_SYSCTL_ESDHCCTL_CRS_1)	Added note to RSTA bit description " 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."
Present state register (eSDHC_PRSSTAT)	Changed "By default, the read value of this bit field after reset is 8'b11110111, when DAT[3] is pulled down and the other lines are pulled up" to By default, the read value of this bit field after reset is 8'b11111111 in DLSL[0:7] description. Changed the access to read only.
External signals overview	Added "SDHC_CMD_DIR is an output signal for CMD line direction control SDHC_DAT0_DIR is an output signal DAT0 line direction control SDHC_DAT123_DIR is an output signal for DAT1-DAT3 lines direction control
External signals overview	Added "SD_VS is an output signal that controls the voltage of external SD Bus Supply to be high voltage (around 3.0V) or low voltage (around 1.8V). ‘0’ stands for high voltage range Optional output"
Auto CMD error status register/ System control 2 (eSDHC_AUTOCERR)	Removed "12" from the bitfield names of AC12TOE, AC12IE, AC12CE, AC12EBE.
Auto CMD error status register/ System control 2 (eSDHC_AUTOCERR)	Added "[0:2]" to UHSM bit name.

Table continues on the next page...

Substantive changes from revision A.2 to revision C.1

Reference	Description
Host controller capabilities register 2 (eSDHC_HOSTCAPBLT2)	Changed bits 29- DDR50 and 30- SDR104 to reserved.
Tuning block (SDICU)Tuning block procedure	Removed references to SDR104 and HS200.
Interrupt status register (eSDHC_IRQSTAT)	Added "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" to the description of CRM, CINS bit.
Host controller capabilities register 2 (eSDHC_HOSTCAPBLT2)	Changed bits RTM to read only. Changed bit width of TCRT bits to 20-23 and bit 24 to reserved.

B.1.17 Universal Serial Bus Interfaces Revision History

Reference	Description
Executing a transfer descriptor	In point 7 and 8, updated "If status bit read in (3) ..." as "If status bit read in (4)...".
Frame adjust register	Removed 'for 16-bit physical interfaces' from last sentence.
Table 17-147	From bit 0, [T], removed "(in combination with the CTRLDSSEGMENT register if applicable)".
Periodic frame list	Removed the second paragraph,:“Split transaction interrupt, bulk and control are also managed using queue heads and queue element transfer descriptors.” Modified the third para starting from “The periodic frame list is...”
Periodic frame span traversal node (FSTN)	Removed the following sentence starting from “Software must not use...” in the first para: “Software must not use the FSTN feature with a host controller whose HCIVERSION register indicates a revision implementation under 0x0096. Note that FSTNs were not defined for EHCI implementations before Revision 0.96 of the EHCI Specification and their use may yield undefined results. Add the following note: “The host controller performs only read operations to the FSTN data structure.”
Periodic isochronous-do-start-split	Replace the sentence "The preferred method is to.....zero-length data payload" with ""Setting the Active bit to zero depends on siTD.TP being 00 or 11, and siTD. Total Bytes decrements to 0.”
USB status (USB_USBSTS)	In bit description of [SLI], replaced "The device controller clears the bit upon exiting from a suspend state." with "This bit is cleared via software by writing 1 to it. "
Config1Config2Status1	Added registers
USB_BURSTSIZE	Removed register.
Host controller initialization and Device controller initialization	Removed bullet "Optionally modify the BURSTSIZE register."
Host TT transmit pre-buffer packet tuning (USB_TXFILLTUNING)USB device mode (USB_USBMODE)Port status/control (USB_PORTSC)	Updated Reset values.

Table continues on the next page...

Reference	Description
CONTROL	Updated description of the [USB_EN] bit.
USBPHY	Renamed register and bit name. Updated description.

B.1.18 DUART Revision History

Reference	Description
DUART Features Summary	Moved the UART block diagram to Overview section.

B.1.19 SerDes Module Revision History

Table B-1. T2081 Appendix

Reference	Description
SRDSx XAUIn Protocol Control Register 1 (SRDSxXAUInCR1)	Added register
SRDSx Lane m General Control Register 0 (SRDSxLNmGCR0)	Added TTRM_VM_SEL bitfield.
SerDes Lanes Assignments and Multiplexing	Added options 67-1F, AB-02, DB-1F, DB-2D, 66-16.

B.1.20 PCI Express Interface Controller Revision History

Reference	Description
PCI Express Inbound Window Attributes Register (PEXIWARn) PCI Express VF Inbound Window Attributes Registers (PEXVFIWARn)—EP Mode Only PCI Express Expansion ROM Inbound Window Attributes Registers (PEXEPROMIWAR) P CI Express MSI Inbound Window Attributes Registers (PEXMSIIWAR)—RC Mode Only	Added TRGT mapping 0000 PCI Express 1 0001 PCI Express 2 0010 PCI Express 3 1000 Serial RapidIO 1 1001 Serial RapidIO 2 Updated the reset value of IWS bits from nnnnn to 010111 for window 0
PCI Express ATMU Outbound Messages	Added note indicating that Set Slot Power Limit messages are not supported via ATMU, but are supported via PCI Express configuration register.
Gen 3 Link Equalization	Added section explaining that Gen3 link equalization must be disabled in loopback mode.

B.1.21 Serial RapidIO Interface Revision History

Reference	Description
Port 1 RapidIO inbound window attributes register 0 (SRIO_P1RIWAR0)Port 2 RapidIO inbound window attributes register 0 (SRIO_P2RIWAR0)	Changed the access for EN bit to read only.
Port 1 Accept-all configuration register (SRIO_P1AACR)Port 2 Accept-all configuration register (SRIO_P2AACR)	Changed the reset value of AA bits to 1
Port 1 RapidIO inbound window attributes register 0 (SRIO_P1RIWAR0)Port 2 RapidIO inbound window attributes register 0 (SRIO_P2RIWAR0)Port 1 RapidIO inbound window attributes register n (SRIO_P1RIWAR*n*)Port 2 RapidIO inbound window attributes register n (SRIO_P2RIWAR*n*)	Added TGINT mapping
Port 1 Accept-all configuration register (SRIO_P1AACR)Port 2 Accept-all configuration register (SRIO_P2AACR)	Updated the reset value to 8000_0001
Port 1 RapidIO Inbound window base address register n (SRIO_P1RIWBAR*n*)Port 2 RapidIO Inbound window base address register n (SRIO_P2RIWBAR*n*)	Removed registers P1RIWBAR0 and P2RIWBAR1

B.1.22 SATA Controller Revision History

Reference	Description
Vendor-specific BIST operation	Added BIST Operations.

B.1.23 DMA Controller Revision History

Reference	Description
DMA signal descriptionsSignal overview	Added note "External Control is supported on channel 0 only".Removed duplicate figures.
DMA signal descriptions	Replaced the link to the Global Source and Target IDs by a link to the Signals table.

Table continues on the next page...

Reference	Description
DMA controller memory map	Removed information about and bitfields associated with ATMU bypass mode from SATRn, SARn, DATRn, and DARn.
DMA errors	Removed last bullet related to ATMU bypass mode.
DMA controller limitations and restrictions	Removed bullets related to ATMU bypass mode.
DMA features summary	Removed bullet regarding support for three priority levels
DMA channel operation	Removed paragraph that started, "The DMA controller is designed to support RapidIO transaction types, including various..."
DMA errors	Removed bullet, "Transfer started with a priority of three"

B.1.24 DPAA Overview Revision History

No substantive changes

B.1.25 Multicore Programmable Interrupt Controller (MPIC) Revision History

Reference	Description
Interprocessor Interrupt Vector/Priority Registers (IPIVPR0–IPIVPR3)	Corrected the register name at offset 0x10D0 IPIVPR0 to IPIVR3 in the figure.

B.1.26 T2080 Interrupt Assignments Revision History

No substantive changes

B.1.27 Device Configuration and Pin Control Revision History

Reference	Description
PAMU Bypass Enable Register (DCFG_CCSR_PAMUBYPENR)	Renamed "PBYP[1-16]" bit field to "PBYP"; also resized size and updated reset value. Added footnote to reset value: "If secure boot is enabled (either by RCW[SB_EN] or the ITS fuse in the SFP block), these bits reset to 0; if secure boot is not enabled, these bits reset to 1".
POR Status Register 1 (DCFG_CCSR_PORSR1)	Added bit 19 XVDD_SEL
Preboot Loader Status Register (DCFG_CCSR_PBLSR)	Removed unused bits in DCFG_PBLSR.
Reset Control Register (DCFG_CCSR_RSTCR)	Changed bit 27 to reserved

Table continues on the next page...

Reference	Description
Core Reset Status Register n (DCFG_CCSR_CRSTSR)	Changed bit 28 to reserved
Reset Request WDT Mask Register (DCFG_CCSR_RSTRQWDTMR), Reset Request WDT Status Register (DCFG_CCSR_RSTRQWDTSR)	In the WDT field table, added rows for "Core 3 Thread 0" and "Core 3 Thread 1".
DDR Clock Disable Register (DCFG_CCSR_DDRCLKDR)	Changed bits 10-13 to D1_MCK[0-3] and bits 0-3 to reserved.

B.1.28 General Purpose I/O (GPIO) Revision History

No substantive changes

B.1.29 Thermal Management Unit Revision History

No substantive changes

B.1.30 Run Control and Power Management Module Revision History

Reference	Description
Timebase Clock Divider Register (TBCLKDIVR)	Updated the bitfield description to 0x00: 1/16 0x02: 1/8 0x06: 1/24 0x08: 1/32
Core PH20 (PGSR) Operation	Added section
Core PH30 (PG) Operation	Added section
Thread PH10 (Doze) Operation	Changed Corresponding bit(s) in RCPM CDOZCR are clear. Core debug interrupt Core time based interrupt to Corresponding bit(s) in RCPM TPH10CLRR are set. Thread interrupt without RCPM TPMNTMR masking. Thread IRQ interrupt without RCPM TPMIMR masking. Thread critical interrupt without RCPM TPMCIMR masking. Thread debug interrupt.
RCPM memory map	Removed PCTBENR1 and PCTBCKSELR1
Power Management Modes	Rearranged section. Added cluster and thread power management states.
Core RCPM Mode Summary for Power Management	Added “PH30 mode can also be exited via core warm reset by writing to MPIC PIR.”

Table continues on the next page...

Reference	Description
Core RCPM Mode: Entry and Exit for Power Management	Added “Core timebase interrupt if enable via PCTBENR" as another exit source” to PH10 Exit Via column.
Core RCPM Mode: Entry and Exit for Power Management	Added “Core timebase interrupt if enable via PCTBENR" as another exit source” to PH15 Exit Via column
Power-On Reset State Hard Reset State	Added Cluster power management state machine Core power management state machine Thread power management state machine

B.1.31 T2081 Appendix

Reference	Description
SerDes Assignments	Added options 6C and 70. Added explanation for the notation used in the table

B.2 Substantive changes from revision A.1 to revision A.2

Substantive changes from revision A.1 to revision A.2 are as follows:

B.2.1 Overview

Reference	Description
	Added note "Further details describing the full debug functionality of this device are beyond the scope of this document. Please contact your preferred third-party tools vendor for additional information and guidance for leveraging debug capabilities of QorIQ products."
Debug support	Removed Figure, "Debug Architecture".
Debug support	Replaced 3 sentences with 6 more descriptive sentences from debug_perfmon_overview chapter.

B.2.2 Memory Map Revision History

Reference	Description
CCSR Address Map	Changed USB1 PHY to dual role from host role and removed offsets for USB1PHY and USB2 PHY as the PHY has continous register space.

Table continues on the next page...

Reference	Description
CCSR Address Map	Removed USB PHY from the CCSR address map

B.2.3 Signals Revision History

Reference	Description
UART and GPIO1 signal multiplexing	Replaced hex RCW values by binary values

B.2.4 Reset Revision History

Reference	Description
DRAM type select	Changed cfg_dram_type_select from IFC_AD[21] to IFC_A[21].
SerDes XVDD voltage select	Changed XVDD voltage for setting 0 to reserved

B.2.5 Pre-Boot Loader Revision History

Reference	Description
PBL Features Summary	Added 1.8V support.
PBL Features Summary	Added "with external voltage translator" to 3.3V.

B.2.6 Secure Boot and Trust Architecture Revision History

Reference	Description
SFP Configuration Register (SFP_SFPCR)	Added SFP configuration register (SFPCR) that controls adjustment of the program pulse width.
Security Monitor functional description	Revised the section to eliminate the bulleted list.
One Time Programmable Master Key n (SFP_OTPMKR*n*)	Added register/key bits mapping in the OTPMKn bitfield description.

Table continues on the next page...

Reference	Description
Transitioning among system security monitor states (Formerly, "Security state machine" and "State definitions"	Combined the two sections into a single section. Revised the security states diagram

B.2.7 e6500 Core Integration Revision History

Reference	Description
Processor Version Register (PVR)	Updated Processor Version Register (PVR) values
Dealing with Guest OS WIMGE Aliasing in a Boundedly Undefined Manner	Added section

B.2.8 CoreNet Platform Cache (CPC) Revision History

Reference	Description
CPC error injection control register (CPC_CPCERRINJCTL)	Removed "CPCERRINJHI[31] and" in the bitfield description of TERRIEN bits to change it to " Tag error injection is determined by CPCERRINJLO[7:31]"
CPC configuration and status register 0 (CPC_CPCCSR0)	Changed bit 23 to reserved
Error Injection	Added "Tag error injection is determined by CPCERRINJLO and CPCERRINJCTL, at allocation."

B.2.9 Prefetch Manager Revision History

Reference	Description
	Removed registers PMAN Operation Error Status 1 and 2

B.2.10 CoreNet Coherency Fabric (CCF) Revision History

Reference	Description
CECAR	Changed the access type for all non-reserved bits in CCF_CECAR to write-1-to-clear (w1c)

B.2.11 Peripheral Access Management Unit (PAMU) Revision History

Reference	Description
PAMU Features SummaryPAMU Address Capabilities Register 1 (PAMU_PAC1)	Replaced 36 by 40 in "Support for a maximum 36-bit system address space" and changed PAMU_PAC1 bitfield setting to 0x0000_00FF Maximum 40-bit system address space

B.2.12 DDR Revision History

Reference	Description
Supported DDR SDRAM Organizations	Added rows for 8 Gbits support
DDR Introduction	Removed x32 as there are no DDR3 memories that support x32 available. Removed x4. Removed "2 or" in "A sub-bank is specified by the 2 or 3 bits on the bank address (MBA) pin"
DDR FeaturesDDR SDRAM interface operation	Removed x32 as there are no DDR3 memories that support x32 available. Removed x4 mode support
DDR Signals Overview	Removed MDQS[9:17], MDQSL[0:8] in x4 mode. Also changed the active low signals to "_B". Deleted duplicate rows for MAPAR_ERR, MAPAR_OUT. Corrected D2_MCK[0:5] to 0:3
Memory Interface Signals	Added _B to the active low signals.
Memory Interface Signals	Removed "Note that the MDQS[9:17] IOs are only used if DDR_SDRAM_CFG_2[X4_EN] is set. Also refer to the MDM signal descriptions for the true data strobes used in x4 mode." Also removed "If DDR_SDRAM_CFG_2[X4_EN] is set, then these pins will act as the true data strobes for the lower nibble of each byte. For example, MDM[0]/MDQS[9] would be associated with DQ[4:7] in this mode. In this mode, the timings will match what is shown for the MDQS[0:8] IO"
DDR SDRAM control configuration 2 (DDR_DDR_SDRAM_CFG_2)	Changed bit 21 x4_EN to reserved
DDR Introduction	Removed "Only x32 DRAMs that use 1 data strobe per data byte are supported."
DDR SDRAM control configuration (DDR_DDR_SDRAM_CFG)	Added note to MEM_EN: The DDRC must be programmed before the corresponding Local Access Windows are programmed and enabled for the DDR target(s).
DDR SDRAM timing configuration 0 (DDR_TIMING_CFG_0)	Changed description of TIMING_CFG_0[ACT_PD_EXIT] from "Active powerdown exit timing (tXARD and tXARDS)" to "Active powerdown exit timing (tXP)".
DDR Memory Test Control Register (DDR_DDR_MTCR)	Added note to DDR_DDR_MTCR[MT_EN]: The memory test may not be used if memory controller interleaving has been enabled via CS0_CONFIG[INTLV_EN].

Table continues on the next page...

Reference	Description
DDR SDRAM timing configuration 3 (DDR_TIMING_CFG_3), DDR SDRAM timing configuration 1 (DDR_TIMING_CFG_1)	Changed description of TIMING_CFG_3[EXT_REFREC] from "This field is concatenated with TIMING_CFG_1[REFREC] to obtain an 8-bit value" to "This field is concatenated with TIMING_CFG_1[REFREC] to obtain an 9-bit value...." Changed description of TIMING_CFG_1[REFREC] from "This field is concatenated with TIMING_CFG_3[EXTREFREC] to obtain a 7-bit value" to "This field is concatenated with TIMING_CFG_3[EXTREFREC] to obtain a 9-bit value".
DDR Signals Overview	Removed references to MDVAL and MSRCIDn.
DDR Functional Description	In Figure 12-73, removed references to MDVAL and MSRCIDn.

B.2.13 Integrated Flash Controller (IFC) Revision History

Reference	Description
Generic ASIC read data sampling	Added last three paragraphs.
Mode 0 pin muxing (CSOR*n*[ADM_SHFT_MODE] = 0)	Updated description and example figures.
Mode 1 pin muxing (CSOR*n*[ADM_SHFT_MODE] = 1)	Updated description and example figures.
NAND configuration register (IFC_NCFGR)	Updated description of IFC_NCFGR[SRAM_INIT_EN] field.
Flash Timing Register 1 for CS*n* - NOR Flash Mode (IFC_FTIM1_CS*n*_NOR)	Changed TSEQRAD_NOR bits from 26-31 to 24-31, TRAD_NOR bits from 18-23 to 16-23.
Extended Chip Select Option Register - NAND Flash Mode (IFC_CSOR*n*_EXT)	Added register.
Clock control register (IFC_CCR)	Updated bit 20 as 'FB_IFC_CLK_SEL'.
NAND chip-select register (IFC_NAND_CSEL)	Updated bit settings of [CSEL].
NAND transfer error attributes register 0 (IFC_NAND_ERATTR0)	Updated bit [ERCS] from 4-5 to 3-5.
NOR transfer error attributes register 0 (IFC_NOR_ERATTR0)	Added bit 0-7[ERSRCID] and 12-19 [ERAID] and added '...' in bit settings of [ERCS] to show range..
GPCM transfer error attributes register 0 (IFC_GPCM_ERATTR0)	Added bit 0-7[ERSRCID] and 12-19 [ERAID] and updated bit ERCS as 23-25.
Address mask register (IFC_AMASK*n*)	Added paragraph "IFC has only 32 address pins at..."
Chip-Select Option Register - NAND Flash Mode (IFC_CSOR*n*_NAND)	Updated IFC_CSORn_NAND[SPRZ] bit description.
Flash Timing Register 0 for Chip-Select *n* - NAND Flash Mode (IFC_FTIM0_CS*n*_NAND)	Added programming guidelines for timing registers in the register description.
Internal connectivity of the write protect and ready/busy signals	Added section.
Ready busy status for each chip-select (IFC_RB_STAT)	Changed the bitfield description of bits RB0, RB2-RB7 to 0: At least one of the device connected at CS0, CS2-CS7 is busy 1: All devices connected to CS0, CS2-CS7 are ready.

Table continues on the next page...

Substantive changes from revision A.1 to revision A.2

Reference	Description
BCH encoding	Added example and figure.
Table 13-243	Added table.
SRAM buffer initialization requirement	Removed the SRAM initialization steps involving reading a block/page from the NAND flash, as this is irrelevant with the addition of the NCFGR[SRAM_INIT_EN] bit.
NAND asynchronous mode calculating read data window width	Added note at the end of the section where in EDO mode, the value of TRAD should always be less than t_{RP} + t_{REH}. See section for complete note.
NAND asynchronous mode boot process	Added the section at the end regarding hard-coded FIR sequences used during auto-boot operation, including Table 13-248 and the following note.
Normal GPCM read operation	Removed dependency of normal GPCM read operation on CSORn[WEGTA]; it is only dependent on CSORn[REGTA].
Figure 13-262	Updated figure "Burst write cycle timing".
Normal GPCM internal counter-based read operation	Added information about normal GPCM burst mode.
Normal GPCM internal counter-based read operation	In first paragraph, added relevant of FTIM3_CSn_GPCM[TAAD].
IFC revision control register (IFC_REV)	Changed the reset value of REV_MIN bits to nnnn
GPCM transfer error attributes register 2 (IFC_GPCM_ERATTR2)	Bit description of [PERR_BYTE] updated from "(bit 8 represents byte 0, the most significant byte lane)" to "(bit 24 represents byte 0, the most significant byte lane)".
GPCM event and error enable register (IFC_GPCM_EVTER_EN)	Updated the reset value from 0x0500_0000 to 0x0540_0000.
Mode 1 pin muxing (CSOR*n*[ADM_SHFT_MODE] = 1)	Reworded the second and third paragraphs for clarity.
Throughout chapter	Replaced MSB and LSB references with msb and lsb, respectively.
Transceiver enable during boot	Replaced references to por_cfg_te signal with cfg_ifc_te.
Figure 13-234	On the ADM NOR, corrected addressing from 26 bits (A0:A25) to 27 bits (A0:A26).

B.2.14 I2C Revision History

No substantive changes

B.2.15 Enhanced Serial Peripheral Interface Revision History

Reference	Description
eSPI CS0 mode register (ESPI_SPMODE0), eSPI CS1 mode register (ESPI_SPMODE1), eSPI CS2 mode register (ESPI_SPMODE2), and eSPI CS3 mode register (ESPI_SPMODE3)	In SPMODEn[LENn] bitfield description, added text "Supports a range from 1-bit to 16-bit data characters."

B.2.16 eSDHC Revision History

Reference	Description
eSDHC signal descriptions	Added SDHC_CMD_DIR , SDHC_DAT0_DIR , SDHC_DAT123_DIR.
eSDHC signal descriptions	Added SDHC_VS.
System control register when ESDHCCTL[CRS=0] (eSDHC_SYSCTL)	Added note "The Base clock frequency is platform_clk/2." to bits SDCLKFS.
Protocol control register (eSDHC_PROCTL)	Updated bit description and bit settings value of VOLT_SEL.
System control register when ESDHCCTL[CRS=0] (eSDHC_SYSCTL)	In bit description of [SDCLKFS], added note and updated text "According to the SD Physical Specification Version 2.0 and the SDIO Card Specification Version 2.0 , the maximum SD clock frequency is 50 MHz and should never exceed this limit." as "The programmed SD Clock frequency....". Added note "Both DVS and SDCLKFS ...." in bits [SDCLKFS][DVS].
System Control Register when ESDHCCTL[CRS=1] (eSDHC_SYSCTL_ESDHCCTL_CRS_1)	In bit description of [SDCLKFS], updated bit setting 0x00 from Base Clock to reserved. Also removed "Setting 000h bypasses the frequency divisor of the SD Clock."
	Updated reset value of eSDHC_HOSTCAPBLT.
Host controller capabilities register 2 (eSDHC_HOSTCAPBLT2)	Added bit 20-23 as TCRT. Updated reset value.
Tuning control register (eSDHC_TCR)	Added note "Write or read to reserved fields of this register doesn't guarantee specific value to be written or read."
eSDHC control register (eSDHC_ESDHCCTL)	Renamed register eSDHC control register.
Tuning control register (eSDHC_TCR)	Added new register.
External signals overview, eSDHC signal descriptions	Removed SDHC_CLK_SYNC_OUT and SDHC_CLK_SYNC_IN.
eSDHC signal descriptions	Changed "0 stands for low voltage" to "stands for high voltage" in the description of SDHC_VS.
eSDHC features summary	Added "Supports SD UHS-1 speed modes: SDR12, SDR25, SDR50"
DDR	Added seven new steps in DDR mode configuration sequence. Added note 2.
Interfacing Card	Added new section.

B.2.17 Universal Serial Bus Interfaces Revision History

Reference	Description
USB Overview	Removed redundant paragraph describing what signalling the module supports.
USB_PHY Memory Map/Register Definition	Corrected the location and bit definitions for the USB PHY registers.
Memory Map	Removed USB PHY registers

B.2.18 DUART Revision History

Reference	Description
DUART External Signal Descriptions	Added "UART1_CTS is not available when UART3 is configured for SIN/SOUT mode and UART2_CTS is not available when UART4 is configured for SIN/SOUT mode. The DUARTs are set during reset to be in either 2-pin or 4-pin mode (or disabled if the GPIO functionality is selected). For 2-pin mode, the CTS_B inputs are asserted internally."

B.2.19 SerDes Module Revision History

Reference	Description
Throughout	Replaced HiGig+ with HiGig2
	RemovedT2080 tag from “XFI/10GBASE-R” and Serdes2 in Section 19.1.1, Features.
throughout	Removed QSGMII references as it is not supported on T2080.
PLLnCR1	• Corrected default value of PLLnCR1 • More updates to MDIO registers, including completed description of 1000Base-KX PCS MMD, and half of AN MMD
	• Documented CPRI and JESD initialization • Updated SRDSxCPRInCR0/1 RX_DLY and TX_DLY • Corrected 1GKX settings for TECR0[AMP_RED] • Corrected SGMII settings for REIDL_ET_SEL • Make PLLRST_B and SDRST_B visible, since they are used in PLL reconfiguration sequence • Update PLL reset and reconfiguration sequence • Corrected AN expansion default value and NP_ABLE description • Correction SGMII Revision -> Design Revision and default value • Corrected AN control default value • Corrected XNP transmit and BP ethernet status default value • Corrected 1000Base-KX AN LP Base Page Ability Register 2 name typo • Completed documenting 1000Base-KX registers • Completed documenting XFI/10GBase-KR registers • Corrected SGMII and 1000Base-KX initialization sequences • Added initialization sequence for 2.5G SGMII • Added XFI • Added EEE bits to SGMII ability registers • Corrected SGMII ability register descriptions • Added EEE status bits to QSGMIInCR3 • Replaced PEX with PCIe
	• Added SATA 6G settings in SSCR1 • Added OSETOVD_EN, OSETOVD, BAS_WAND

B.2.20 PCI Express Interface Controller Revision History

Reference	Description
PCI Express Outbound ATMU Registers	Replaced “IWS” by “OES” in “Note that the size of a particular BAR for all the VFs in a PF is determined by PEXVFOWARn[OWS]. The total BAR memory size is determined by PEXVFOWARn[OWS] times the .....” as the register does not have IWS field in reference to T2080RM Rev.A.1 special review comment by Toby Foster
Modes of Operation	Added “and operational generation possibilities (for example 1.0, 2.0, 3.0)”
External Signal Descriptions	Removed the “_P” and “_N” in the signal names and added “_B” for negative signals.
PCI Express Correctable Error Status Register—0x110	Added bit 13 ADVNFE in reference to SR#1-858965228
PCI Express Inbound Window Attributes Registers (PEXIWARn)	Replaced nnnn in TRGT reset value to 1110 as it defaults to CCSR BAR for window 0
PCI Express Memory Mapped Registers	Changed the reset value from 0xnnnn_nnnn to 0x0000_0000 in the memory map.
PCI Express Configuration Address Register (PEX_CONFIG_ADDR)	Updated PFN field descriptions for EP with SR-IOV to indicate that PF's 2-15 are reserved
PCI Express SR-IOV Extended Capability ID Register (SR-IOV-only)—0x150	Updated register offset from 0x154 to 0x150
PCI Express Expansion ROM Inbound Window Attributes Registers (PEXEPROMIWAR)	Corrected the register offset in the register description to 0xCF0

B.2.21 Serial RapidIO Interface Revision History

Reference	Description
Inbound LIODN TranslationLIODN Lookup tableAccessing the LIODN Lookup tableTranslation	Added sections
Port 1 RapidIO outbound window attributes register 0 (SRIO_P1ROWAR0)Port 2 RapidIO outbound window attributes register 0 (SRIO_P2ROWAR0)Port 1 RapidIO outbound window attributes register n (SRIO_P1ROWAR*n*)Port 2 RapidIO outbound window attributes register n (SRIO_P2ROWAR*n*)	Updated description of [TFLOWLV] field to be more clear.
IP Block Revision Register 1 (SRIO_IPBRR1)	Updated IPMN and reset value of the register
Data streaming information capability register (SRIO_DSICAR)	Updated reset value to 0x 0000_0018
Logical/Transport layer error detect command and status register (SRIO_LTLEDCSR)	Changed bits 10-14 to reserved and added bit 22 ILE

Table continues on the next page...

Substantive changes from revision A.1 to revision A.2

Reference	Description
Logical/Transport layer error enable command and status register (SRIO_LTLEECSR)	Added bit 22 ILE
Logical layer configuration register (SRIO_LLCR)	Changed bit 3 to reserved. Added "or send Type 9 Flow Control Packets" to "This bit controls whether or not a device is allowed to issue requests into the system. If M = 0, the device may only respond to requests or send Type 9 Flow Control Packets."
Logical Layer RapidIO Errors Detected	Added LIODN entries to the tables
Port 1 RapidIO outbound window attributes register 0 (SRIO_P1ROWAR0)Port 2 RapidIO outbound window attributes register 0 (SRIO_P2ROWAR0)Port 1 RapidIO outbound window attributes register n (SRIO_P1ROWAR*n*)Port 2 RapidIO outbound window attributes register n (SRIO_P2ROWAR*n*)	Updated the reset values
Data streaming logic layer command and status register (SRIO_DSLLCSR)	Changed the bitfield access of MTU bits from Read only to RW
Port 1 Control command and status register (SRIO_P1CCSR)Port 2 Control command and status register (SRIO_P2CCSR)	Updated reset value of PnCCSR registers from D0C0_0001 to D0C2_0001

B.2.22 SATA Controller Revision History

Reference	Description
Error processing	In the fourth bullet under fatal error tasks, updated the second sentence from "This cause san out-of-bounds..." to "This will initiate...".

B.2.23 DMA Controller Revision History

Reference	Description
DMA features summaryDMA signal descriptions	Added "These signals are available on DMA1 and DMA2. They are not supported by DMA3. For more details see Global Source and Target IDs"
Basic chaining, single-write start modeExtended chaining, single-write start mode	Replaced 36 by 40 in "Note that ECLNDARn must be written first so that the full 36-bit descriptor....." and in "Note that ECLSDARn must be written first so that the full 36-bit descriptor....."
DMA mode register (DMA_MR*n*) and DMA mode register (DMA_MR*n*)	Changed the bitfield description of DMAx_MRn[XFE] to: 0 Disable striding feature in direct mode or disable list chaining feature in chaining mode. 1 Enable striding feature in direct mode or enable list chaining feature in chaining mode.

Table continues on the next page...

Reference	Description
DMA next link descriptor address register (DMA_NLNDAR*n*)DMA current list descriptor address register (DMA_CLSDAR*n*)DMA current list descriptor address register (DMA_CLSDAR*n*)DMA next link descriptor address register (DMA_NLNDAR*n*)DMA next list descriptor address register (DMA_NLSDAR*n*)DMA next list descriptor address register (DMA_NLSDAR*n*)	Added "Contains the bits 32-5 of the 40-bit....." to the description of bits 0-26.

B.2.24 DPAA Overview Revision History

No substantive changes

B.2.25 Multicore Programmable Interrupt Controller (MPIC) Revision History

No substantive changes

B.2.26 Interrupt Assignments Revision History

No substantive changes

B.2.27 Device Configuration and Pin Control Revision History

Reference	Description
Core Cluster n Topology Register (DCFG_CCSR_TP_CLUSTER*n*)	Updated offset to 0x840.
Fuse Status Register (DCFG_CCSR_FUSESR)	Changed the encodings for DA_V bitfield setting 00000 -1.0000 V to unused (default) and 10000 -unused to 1.0000 V
System Version Register (DCFG_CCSR_SVR)	Updated SVR
IFC_RR	Added more details for interpretation of DCFG_RSTRQSR1[IFC_RR]
POR Status Register 1 (DCFG_CCSR_PORSR1)	
Processor Version Register (DCFG_CCSR_PVR)	Updated PVR
QMBM Warm Reset Control Register (DCFG_CCSR_QMBM_WARMRST)	Added **NOTE:** Before initiating warm reset software must:

Table continues on the next page...

Reference	Description
	Shut down software portal interfaces (stop de-queueing new work from QMAN and stops enqueueing new work to QMAN). Quiesce FMAN **NOTE:** After setting the warm reset bits software must: Poll QMBM_WARMRST to know when warm reset has completed. Re-initialize QMAN/BMAN. This may include clearing out any history it has of QMAN portal ring state as well as forgetting about buffers it may not have previously released. Restart FMAN. For more details please refer the DPAA Appendices.
IFC Clock Disable Register (DCFG_CCSR_IFCCLKDR)	Removed "The output is tri-stated, when disabled.". Added "The output is not driven and in a high impedance state, when disabled. " to IFC_CLK2_DIS bitfield description.
IFC Clock Disable Register (DCFG_CCSR_IFCCLKDR)	Changed bit 31 to reserved

B.2.28 General Purpose I/O (GPIO) Revision History

Reference	Description
GPIO signal descriptions	Added signals in the first column in the table

B.2.29 Thermal Management Unit Revision History

Reference	Description
Local Temperature Sensor Placement	Added topic
TMU monitor temperature measurement interval register (TMU_TMTMIR)	Removed 800 MhZ platform clock frequency column from the table in TMU monitor temperature measurement interval register description.
TMU mode register (TMU_TMR)	Updated the description of MSITE
Thermal Monitoring Unit (TMU)	Changed the number of sites for T2080 from 4 to 3
Modes of Operation	Changed the number of modes supported to one. Modified the table rows to Monitoring mode enabled and disabled
TMU interrupt detect register (TMU_TIDR)	Added "This includes an out-of-range measured temperature above 125C" to the description of the bits ITTE

B.2.30 Run Control and Power Management Module Revision History

Reference	Description
Cluster PCL10 Set Control Register (CLPCL10SETR0-CLPCL10SETR3)	Replaced PH10 by PH30 in the bitfield description.
IP Powerdown Exception Control Register n (IPPDEXPCRn)	Added IP assignments

Appendix C
Terminology, Conventions, and Resources

C.1 About this content

The primary objective of this document is to define the functionality of the T2080 QorIQ Advanced Multiprocessing Processor. The T2080 provides integration of processing power for networking and communications peripherals, resulting in higher device performance. The T2080 contains four dual-threaded processor cores built on Power Architecture® technology. The e6500 processor core is a low-power implementation of the family of reduced instruction set computing (RISC) embedded processors that implement the embedded category features of the Power Architecture technology. This document is intended as a companion to the following:

- The *EREF 2.0: A Programmer's Reference Manual for Freescale Power Architecture® Processors* describes the instruction, register, and interrupt models, as well as other functionality at the architecture level.
- The *e6500 Reference Manual* describes functionality that is specific to the e6500 and that is not defined by the architecture, such as instructions, registers, and register fields. It also provides e6500-specific details about how the e6500 implements functionality that is defined by the architecture.
- The *AltiVec Technology Programming Environments Manual for Power ISA™ Processors* describes how AltiVec technology is defined for Freescale processors that implement Power ISA.

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.

C.2 Acronyms and abbreviations

This table describes commonly-used acronyms and abbreviations used in this document.

Table C-1. Acronyms and abbreviations

Acronym/ Abbreviation	Meaning
8b/10b	8-bit/10-bit encoding
AMC	Advanced mezzanine card
ATMU	Address translation and mapping unit
BD	Buffer descriptor
BTB	Branch target buffer
BUID	Bus unit ID
CAM	Content-addressable memory
CCF	CoreNet coherence fabric
CCSR	Configuration control and status register
CLASS	Chip-level arbitration and switching system
CRC	Cyclic redundancy check
DCSR	Debug configuration and status register
DDR	Double data-rate
DIP	Dual inline package
DPLL	Digital phase-locked loop
DTLB	Data translation lookaside buffer
dTSEC	Data path three-speed Ethernet controller
DUART	Dual universal asynchronous receiver/transmitter
ECC	Error checking and correction
EEST	Enhanced Ethernet serial transceiver
EHCI	Enhanced host controller interface
EPROM	Erasable programmable read-only memory
EEPROM	Electrically-erasable programmable read-only memory
FCS	Frame-check sequence
FIFO	First in, first out
GCI	General circuit interface
GMII	Gigabit media-independent interface
GPCM	General-purpose chip-select machine
GPIO	General-purpose I/O
GPR	General-purpose register
I2C	Inter-integrated circuit
IFC	Intergrated Flash controller
IPG	Interpacket gap

Table continues on the next page...

Table C-1. Acronyms and abbreviations (continued)

Acronym/ Abbreviation	Meaning
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
MAC	Media access control
MDI	Medium-dependent interface
MII	Media independent interface
MMU	Memory management unit
MPIC	Multicore programmable interrupt controller
MSB	Most significant byte
msb	Most significant bit
NMI	Non-maskable interrupt
NMSI	Nonmultiplexed serial interface
No-op	No operation
OCeaN	On-chip network
OCN	
OSI	Open systems interconnection
PCS	Physical coding sublayer
PMA	Physical medium attachment
PMD	Physical medium dependent
PLL	Phase-locked loop
POR	Power-on reset
PRI	Primary rate interface
PWM	Pulse-width modulation
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

Table continues on the next page...

Table C-1. Acronyms and abbreviations (continued)

Acronym/ Abbreviation	Meaning
SCP	Serial control port
SDLC	Synchronous data link control
SD/MMC	Secure digital/multimedia card
SDOS	SmartDSP operating system
SEC	Security Engine
SerDes	Serializer/Deserializer
SFD	Start frame delimiter
SGMII	Serial gigabit media-independent interface
SIU	System interface unit
SMC	Serial management controller
SPI	Serial peripheral interface
SPR	Special-purpose register
SRAM	Static random access memory
SRIO	Serial RapidIO
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

C.3 Audience

It is assumed that the reader understands operating systems, microprocessor system design, and the basic principles of RISC processing.

C.4 Notational conventions

This table shows notational conventions used in this content.

Table C-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, *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
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	
SIGNAL_B or $\overline{\text{OVERBAR}}$	A trailing _B or 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

C.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.freescale.com or contact your local FAE.

Table C-3. Related resources

Resource	Purpose
Reference manual	Provides details about individual implementations
Hardware specifications	Provides specific data regarding bus timing, signal behavior, and AC, DC, and thermal characteristics, as well as other design considerations
Chip errata	Provides additional or corrective information for a particular device mask set. Individual errata items are published cumulatively.
Application note	Addresses specific design issues useful to programmers and engineers working with Freescale processors

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Document Number Document Number T2080RM
Revision C.1, 9/2013